

# LinkedGeoData: A Core for a Web of Spatial Open Data

Claus Stadler<sup>a,\*</sup>, Jens Lehmann<sup>a</sup>, Konrad Höffner<sup>a</sup>, Sören Auer<sup>a</sup>

<sup>a</sup>Department of Computer Science, University of Leipzig

Johannisgasse 26, 04103 Leipzig, Germany

{cstadler, lehmann, auer}@informatik.uni-leipzig.de, konrad.hoeffner@uni-leipzig.de

**Abstract.** Data integration on and off the web requires comprehensive datasets and vocabularies to enable the disambiguation and alignment of information. Many of such 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 development of a spatial Data Web by elaborating on how the collaboratively collected OpenStreetMap data can be interactively transformed and represented adhering to the RDF data model. We describe how this data is 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 several applications, including a faceted geo-browser. The spatial data, vocabularies, interlinks and some of the applications are openly available in the LinkedGeoData project.

Keywords: Linked Data, Spatial Data, Open Data, Interlinking, RDF, RDB2RDF, OpenStreetMap, LinkedGeoData

## 1. Introduction

The Semantic Web eases data integration tasks by providing the basis for overcoming structural and semantic heterogeneity through RDF and ontologies. 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* [10], 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 Web<sup>1</sup>.

Many real-life information integration and aggregation tasks are, however, impossible without compre-

hensive background knowledge related to spatial features of the ways, structures and landscapes surrounding us. Such tasks include, 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)<sup>2</sup> 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.

The goal of our *LinkedGeoData* (LGD) project is to provide a rich integrated and interlinked geographic dataset for the Semantic Web. The majority of our data is obtained by converting data from the popular OpenStreetMap community project to RDF and deriving a lightweight ontology from it. Furthermore, we perform interlinking with *DBpedia*, *GeoNames* and other datasets as well as the integration of icons and multilingual class labels from various sources. As a side ef-

---

\*This work was supported by a grant from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and LATC (GA no. 256975).

<sup>1</sup>See, for example, the listing at <http://ckan.net/group/lodcloud> and an overview at <http://lod-cloud.net>.

---

<sup>2</sup><http://openstreetmap.org>

fect, we are striving for the establishment of an OWL vocabulary with the purpose of simplifying exchange and reuse of geographic data.

After our initial LGD release in 2009 [1], we invested substantial efforts in maintaining and improving LinkedGeoData, which include improvements of the project infrastructure, the generated ontology and data quality in general. Our new contributions since then are:

- A flexible system for mapping OpenStreetMap data to RDF, resulting in *improved data quality*.
- The *SPARQL endpoint* was made publicly available, which led to more 3rd party applications and demos.
- *Better support for ways*: The geometry of a way is now associated with the corresponding RDF resource. Also, all nodes referenced by the way are available both via the Linked Data interface and the SPARQL endpoints.
- An *improved REST interface* with integrated search functions.
- A new publicly accessible *live SPARQL endpoint* that is being interactively updated with the minutely changesets that OpenStreetMap publishes.
- A simple *replication method* of the corresponding RDF changesets so that LinkedGeoData data consumers can replicate our store.
- Direct *interlinking* with *GeoNames* and the *UN FAO* data (interlinks with DBpedia have been updated).
- An *improved LinkedGeoData browser*.
- Implementation of the *Vicibit* application to facilitate the integration of LGD facet views in external web pages.
- Integration of appropriate *icons and multi language labels* for LinkedGeoData ontology elements from external sources.

The paper is structured as follows: after introducing the OpenStreetMap project in Section 2, we outline the LinkedGeoData architecture in Section 3. We describe how the OSM data can be transformed into the RDF data model in Section 4 and how it is republished as Linked Data in Section 7. We present statistics about LinkedGeoData in Section 8 and describe the interlinking with existing data sources on the Data Web in Section 6. In Section 9, we showcase a faceted geo-data browser and editor as well as some 3rd party applications being built around LinkedGeo-

Data. We present related work in Section 10 and conclude in Section 11 with an outlook to future work.

## 2. OpenStreetMap

OpenStreetMap is a collaborative project to create a free editable map of the whole world. It was inspired by Wikipedia and as such it provides well known wiki features such as an edit-tab and a full revision history of the edits. However, rather than editing articles, users edit geographic entities. The three fundamental ones are as follows:

- *Nodes* are the most primitive entities and represent geographic points with a latitude and longitude relative to the WGS84 reference system.
- *Ways* are entities that have a list of at least two node references associated with them. Depending on whether the first reference equals the last one, a way is called *closed* or *open*, respectively.
- *Relations* relate points, ways and potentially other relations to each other, thereby forming complex objects. Each entity participating in a relation plays a certain *role* in it. Multipolygons are modelled with relations.

Each of these entities has a numeric identifier (called *OSM ID*), a set of generic attributes, and most importantly is described using a set of key-value pairs, known as *tags*.

An example of a relation is the administrative boundary of Germany having the OSM identifier 51477.<sup>3</sup> It comprises more than 1000 ways, which represent certain segments of the German border; the German border with Luxembourg e.g. is composed out of approx. 40 way segments. The relation currently has about 30 associated tag-value pairs, which, for example, contain the name of Germany in different languages. One of those tag-value pairs (`boundary=administrative`) indicates that this relation represents an administrative boundary. This information is used by the OSM map renderer to decide how this relation should be rendered on the map. Further tags are used for timezone, currency, and ISO country. The relation has also a few meta-data entries (such as the timestamp of the last edit and the last editor) attached.

---

<sup>3</sup><http://www.openstreetmap.org/browse/relation/51477> can be used to browse this relation.

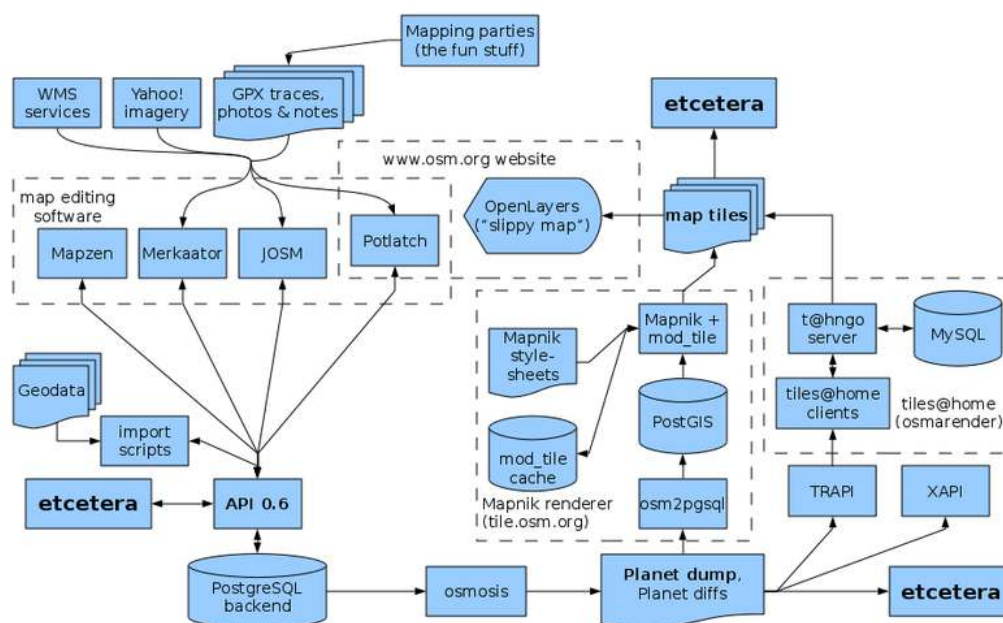


Fig. 1. Overview of OpenStreetMap's architecture.

Source:[http://wiki.openstreetmap.org/w/images/1/15/OSM\\_Components.png](http://wiki.openstreetmap.org/w/images/1/15/OSM_Components.png) as of 2011 Apr 27th

To manage those datastructures, an infrastructure evolved encompassing multiple map editing tools, tile renderers, and data sources, as shown in Figure 2.

The data is stored in a relational database (PostgreSQL backend). It can be accessed, queried and edited by using a REST API, which basically uses HTTP GET, PUT and DELETE requests with XML payload (similar to the example shown in Listing 4). The data is also published as complete dumps of the database in such an XML format on a weekly basis. It currently accounts for more than 16GB 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. The dumps as well as the changesets can be processed with the *Osmosis* tool.

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*, *Merkaartor* and *Mapzen*. The editors use complementary external services and data such as *Yahoo! satellite imagery* or *Web Map Services (WMS)*. Additionally, users can upload GPS traces which serve as raw material for modelling the map. Two different rendering services are offered for the rendering of raster

maps on different zoom levels. With *Tiles@home*, the performance-intensive 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.

Since the use of tags and values is not restricted, but governed by an agile community process, it is important to obtain an overview on emerging tags and tag values possibly specific to a certain region. Services such as *TagWatch*<sup>4</sup> periodically compute tag statistics for different areas. In order for the data to be machine interpretable, as for instance for map rendering, contributors must follow certain editing standards and conventions<sup>5</sup>.

Currently, OSM is in the process of switching from the Creative Commons CC-BY-SA license to the Open Database License<sup>6</sup>. The term *Volunteered Geographic Information (VGI)* was coined [8] for the harnessing of tools to create, assemble, and disseminate geographic data provided voluntarily by individuals – with OSM being a driving force behind VGI.

<sup>4</sup><http://tagwatch.stoecker.eu/>

<sup>5</sup>[http://wiki.openstreetmap.org/wiki/Map\\_Features](http://wiki.openstreetmap.org/wiki/Map_Features)

<sup>6</sup><http://www.opendatacommons.org/licenses/odbl/>

Category	June 2009	April 2010	May 2011	Growth (past two years)
Users (Thousands)	127	261	397	+ 213%
Uploaded GPS points (Millions)	915	1500	2298	+ 151%
Nodes (Millions)	374	600	1073	+ 187%
Ways (Millions)	30	48	92	+ 207%

Table 1

OpenStreetMap statistics 2009 - 2011.

(Obtained from [http://www.openstreetmap.org/stats/data\\_stats.html](http://www.openstreetmap.org/stats/data_stats.html) at the specified months.)

The growth of the OpenStreetMap data has been enormous (cf. Table 1): Since the founding in July 2004 until now, more than one billion nodes, about 90 million ways and close to 1 million relations have been contributed by the users<sup>7</sup>. Some of the data was imported from public domain datasources such as TIGER<sup>8</sup> for US, AND Automotive Navigation Data<sup>9</sup> for The Netherlands, and GeoBase data from the Canadian government<sup>10</sup>.

### 3. Architecture

The goal of the LinkedGeoData the project is to contribute rich, open, and integrated geographical data to the Semantic Web using OpenStreetMap as its base. This is analogous to the well known DBpedia project, which follows a similar approach based on Wikipedia. The necessary work for reaching this goal comprises the conversion of OSM data to RDF, the interlinking with other knowledge bases, the dissemination of the resulting data, and keeping the datasets up-to-date. In this section, we give an overview of the LinkedGeoData architecture, followed by explanations of the details of the involved components in the next sections.

The architecture of LinkedGeoData is illustrated in Figure 2. It shows that the data from OpenStreetMap is processed on different routes: The *LGD Dump Module* converts an OSM planet file to RDF and loads the data into a triple store. This data is then available via the *static SPARQL endpoint*. A copy of that triple store serves as the initial basis for the *live SPARQL endpoint*. The *LGD Live Sync Module* downloads minutely changesets from OpenStreetMap, and computes corresponding changesets on the RDF level

in order to update that triple store accordingly. By publishing these RDF changesets (see Section 7.2), we enable data consumers to sync their own triple store with ours. Note, that not all OSM entities are loaded into the SPARQL endpoints due to performance reasons. We offer SPARQL endpoints for the static and live version, because some use cases require up-to-date information whereas for others, it is more suitable that queries yield the same result over a longer period of time, e.g. due to caching mechanisms.

For data access LinkedGeoData offers downloads, a REST API interface, Linked Data, and SPARQL endpoints. The *REST API* provides limited query capabilities for RDFized data about *all* nodes and ways of OpenStreetMap (relations are currently not supported). It draws its data from a local replica of the OpenStreetMap PostGIS database. The OpenStreetMap community developed a tool named *Osmosis*<sup>11</sup>, which supports setting up such database from a *planet file* and applying changesets to it. In future work, we aim for stronger support of spatial SPARQL queries by exposing PostGIS features via SPARQL.

### 4. RDF Mapping

In this section, we explain our approach to the generation of RDF triples from OpenStreetMap entities. Recall that all such OSM entities have a numeric ID and carry information in form of values for predefined attributes and sets of tags. The values for the predefined attributes, such as the version, the contributing user, and timestamp are static and can also be seen as tags.

We generate URIs for nodes and ways according to the pattern `lgd:node<id>` and `lgd:way<id>`, respectively.<sup>12</sup> The resource corresponding to a way's list of nodes is `lgd:way<id>/nodes`.

<sup>7</sup>[http://www.openstreetmap.org/stats/data\\_stats.html](http://www.openstreetmap.org/stats/data_stats.html), retrieved 2011 May 2nd

<sup>8</sup><http://www.census.gov/geo/www/tiger>

<sup>9</sup><http://www.and.com/>

<sup>10</sup><http://www.geobase.ca>

<sup>11</sup><http://wiki.openstreetmap.org/wiki/Osmosis>

<sup>12</sup>See Appendix A for prefix declarations.

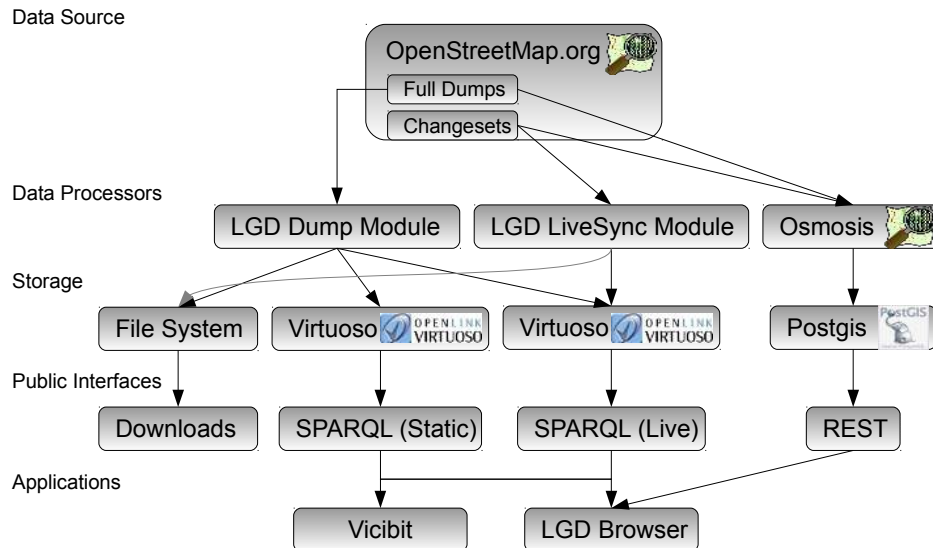


Fig. 2. Overview of LinkedGeoData's architecture.

These URIs are non-information resources, i.e. they represent real-world entities. As such, stating that a resource corresponding to a pub was created by a building company would be correct, however stating that it was created with the map editor “JOSM” would be wrong. In general, there are two possible solutions to permit both kinds of statements: a) introduce distinct URIs for each of the two different meanings, b) make use of annotation properties, which are intended for this purpose and do not have any logical implications. We chose the latter approach, because it avoids doubling the number of resources and keeps the data simple.

Our tag mapping approach is based on the assumption that each tag can be mapped in isolation, i.e. without taking other possibly existing tags into account. For example, entities with the tag (amenity, school) become instances of `lgdo:School`. Note, that this approach does not support more complex rules such as mapping all entities having both tags (amenity, place\_of\_worship) and (religion, christian) to e.g. `lgdo:Church`. Therefore, the generated RDF structures are very close to the structure in OpenStreetMap.

We now specify the mapping process. A *tag mapper* is an object for generating RDF from tags. It consists

of a *tag pattern* that specifies what tags to match, and a *transformation function* for generating the RDF.

Tag patterns can 1) match a specific key-value pair, such as (amenity, school), 2) match all tags with a certain key (regardless of the values), e.g. (tourism, \*), or 3) match every tag. More specific patterns take precedence, e.g. a matching pattern in category 1 overwrites matching patterns in category 2 and 3.

We implemented the following four tag mappers:

- *Resource*: Maps a tag to a specific property and object, whereas both must be URIs. Therefore it can be used for mapping to both object properties and classes. In the latter case the property has to be set to `rdf:type`. Examples are (religion, christian) and (amenity, school) which are mapped to `lgdo:religion lgdo:christian` and `rdf:type lgdo:School`, respectively.
- *Text*: Treats a tag's value as a plain literal. For example (note, nice view).
- *Datatype*: Interpret a value e.g. (seats, 4) with regard to a specific datatype.
- *Language*: A mapper for tags whose key contains a language, such as `name:en`.

All of these mappings are implemented as Java classes, whose instances are configured with an XML

snippet. Listing 1 shows an example of a configuration of a resource tag mapper that is interpreted as follows: The 'simple' in the name reflects our limitation that tags are being mapped in isolation. The mapping rule is applied to every entity that has a tag matching the pattern `(religion, *)`. The element `objectAsPrefix` controls whether a tag's value should be appended to the value given as the object. So in this case, a tag, such as `(religion, pastafarian)`, is mapped to the predicate `lgdo:religion` and object `lgdo:pastafarian`. The element `describesOSMEntity` specifies whether the resulting RDF describes a real world entity's representation on OpenStreetMap or the entity itself. Therefore, it determines whether a mapping's property should become an instance of `owl:AnnotationProperty`.

The text- and datatype tag mappers are both similar to the resource tag mapper, except that they map tag values to objects that are plain or typed literals, respectively. Therefore the language and datatype of these mappers can be set to a constant in their configuration.

The language tag mapper is used for mapping tag values to plain literals with language tags inferred from the tags' keys. For instance `(name:en, Vienna)` would become `(rdfs:label, "Vienna"@en)`. The key of its tag-pattern must be a regular expression containing a group for matching the language, such as `name: ([^:]+)`. Every match for this group is then cross checked against a list of known languages. This avoids for example matching 'alt' as a language from the key `name:alt` for alternative names.

Listing 1: Example of a mapping declaration.

```
<SimpleResourceTagMapper>
  <property>
    http://linkedgedata.org/ontology/religion
  </property>
  <tagPattern>
    <key>religion</key>
  </tagPattern>
  <describesOSMEntity>>false</describesOSMEntity>
  <objectAsPrefix>>true</objectAsPrefix>
  <object>
    http://linkedgedata.org/ontology/
  </object>
</SimpleResourceTagMapper>
```

This approach makes it possible to add new mappings that require more complex processing easy. For example, a future tag mapper could extract the values of `opening_hours` tags (used 60K times on nodes) and generate RDF in the *Good Relations*<sup>13</sup> vocabulary.

<sup>13</sup><http://www.heppnetz.de/projects/goodrelations/>

#### 4.1. The LinkedGeoData Ontology

Based on the OpenStreetMap tags, we derived a lightweight OWL ontology. Subclass relationships are inferred from resource tag mapper configurations: If there are two tag patterns for `(tag1, tag2)` and `(tag1, *)`, which use the `rdf:type` property, then the object of the first tag pattern becomes a subclass of the second tag pattern. For example, Listing 2 shows an example of such tag mappings for the `(amenity, restaurant)` and `(amenity, *)` tag patterns.

Listing 2: Subclass relationship example.

```
<SimpleResourceTagMapper>
  <property>rdf:type</property>
  <tagPattern>
    <key>amenity</key>
  </tagPattern>
  <describesOSMEntity>>false</describesOSMEntity>
  <objectAsPrefix>>false</objectAsPrefix>
  <object>
    http://linkedgedata.org/ontology/Amenity
  </object>
</SimpleResourceTagMapper>
<SimpleResourceTagMapper>
  <property>rdf:type</property>
  <tagPattern>
    <key>amenity</key>
    <value>restaurant</value>
  </tagPattern>
  <describesOSMEntity>>false</describesOSMEntity>
  <objectAsPrefix>>false</objectAsPrefix>
  <object>
    http://linkedgedata.org/ontology/Restaurant
  </object>
</SimpleResourceTagMapper>
```

In order to determine datatype properties, we scanned all OSM tags for those that had keys for which the majority of values could be parsed as boolean, integer, and float datatype values. In order to deal with dirtiness in tag usage, we applied the following two criteria on the relative and absolute error rate:

- At least 99% of a key's values must succeed to parse.
- The absolute number of errors must not exceed 5000.

The most specific datatype meeting these criteria then became the range of the key's corresponding property. If a datatype was determined, all invalid values were omitted in the RDF output.

Object properties were identified as follows: Intuitively, tags that might be suitable for being mapped to object properties meet the condition, that a low number of distinct values covers most its uses. However, this heuristic only serves as an indicator for tag candidates, as the final choice may be subjective. For instance,

only 7 distinct values for the key *note:ja* are used in more than 99% of almost 3.5mio tags. However, since the tag corresponds to a note, we considered a datatype property to be the right choice. An example for an object property is *lgdo:religion*, which links to resources in the *lgdo* namespace, such as *christian*, *muslim*, and *buddhist*. Another example is *lgdo:wheelchair*, which specifies the extent of wheelchair accessibility, using resources mainly corresponding to the values *yes*, *no*, *limited*, and *unknown*. Using those heuristics, we could generate seed mappings for OpenStreetMap, which were then manually reviewed and refined.

#### 4.2. Multilingual labels and icons

The OpenStreetMap community conducts various internationalization efforts, such as for their website, their map editing tools, and their search engine. Some of these efforts are coordinated on *TranslateWiki*, which describes itself as “a localisation platform for translation communities, language communities, and free and open source projects.”<sup>14</sup> Essentially, this wiki enables contributors to assign texts in multiple languages to keys. The group *OpenStreetMap - Website* defines 1441 keys, and has a 100% translation coverage for 13 languages and 12 more languages with a coverage of more than 90%<sup>15</sup>. The keys with the prefix *geocoder.search\_osm\_nominatim.prefix* correspond to human readable representations of individual tags, and as such form a rich, multilingual, and high quality source of labels for classes, properties, and instances, which we integrated into the LinkedGeoData ontology.

As for icons, there exists a *CC-0* licensed collection of 307 SVG map icons (of which 47 icons are alternative versions) from SJJB Management.<sup>16</sup> Currently the LinkedGeoData ontology associates 90 of them with classes, using the annotation property *lgdo:schemaIcon*. The icons themselves are republished on our server. They simplify the creation of visually appealing LGD based applications and mashups.

<sup>14</sup><http://translatewiki.net>

<sup>15</sup><http://translatewiki.net/wiki/Translating:OpenStreetMap/stats/trunk> retrieved 5th May 2011.

<sup>16</sup><http://www.sjjb.co.uk/mapicons/> retrieved 6th April 2011

## 5. Data Access

As briefly mentioned in Section 3, we provide several ways to access LinkedGeoData:

- dataset downloads (HTML download table<sup>17</sup> and actual files<sup>18</sup>), including live sync changesets relative to the latest release<sup>19</sup> (explained in Section 7)
- a static SPARQL endpoint<sup>20</sup>
- a live SPARQL endpoint<sup>21</sup>
- Linked Data via 303 content negotiation (RDF/XML, Turtle, N-Triples, HTML formats supported)
- a REST API

We first show an example data excerpt and then explain the REST API.

### 5.1. Data example

In Listing 3, we give a brief example on how the data in LinkedGeoData looks like. The whole type hierarchy is already inferred, as it is being done in DBpedia, i.e. *rdf:type* relations to all super classes are asserted. The *lgdo:directType* property was added on request in order for applications to easily determine the most specific types of instances. For every way, there exists a triple that contains the positions of all nodes. For open and closed ways the predicates are *georss:linestring* and *georss:polygon*, respectively. Note that this interpretation is not always correct, as in the general case closed ways have to be interpreted in the context of the ways’ tags in order to determine whether the enclosed area counts to the way or not. All nodes belonging to a way are kept in an RDF sequence. In the SPARQL endpoints, geographical coordinates are represented as point geometries that are typed with *virtrdf:Geometry*. OpenLink’s Virtuoso<sup>22</sup> enterprise edition database system automatically indexes such points in an R-tree.

Listing 3: Example dataset in Turtle syntax.

```
lgd:way4009992
```

<sup>17</sup><http://linkedgeo.org/Datasets>

<sup>18</sup><http://downloads.linkedgeo.org>

<sup>19</sup><http://downloads.linkedgeo.org/releases/latest/changesets/>

<sup>20</sup><http://linkedgeo.org/sparql>

<sup>21</sup><http://live.linkedgeo.org/sparql>

<sup>22</sup><http://virtuoso.openlinksw.com>

```

a          lgdo:Tennis, lgdo:Sport, lgdo:Way;
lgdo:directType lgdo:Tennis;
lgdo:contributor lgd:user2274;
lgdo:hasNodes  <http://.../way4009992/nodes>;
georss:polygon  "52.1523857 -1.026259
                 52.1522675 -1.0264068 ..." .
<http://.../way4009992/nodes>
a          rdf:Seq;
rdf:_1     lgd:node21179607;
rdf:_2     lgd:node21179608;
...
lgd:node21179607 geo:geometry
"POINT(-1.02626 52.1524)"^^virt:rd:Geometry

```

## 5.2. The REST API

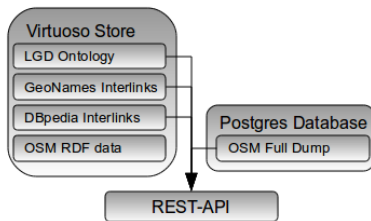


Fig. 3. Data Sources of the REST API.

The LinkedGeoData REST API gives access to all of OpenStreetMap's nodes and ways. It offers a set of methods that all have in common that they return RDF for responses. Each call to the REST API can be combined with content negotiation to format these responses as RDF/XML, N-Triples, or Turtle. The API is backed by two things: On the one hand there is a PostGIS database that is loaded with an OSM planet file and which is updated with minutely OSM change-sets. On the other hand, the data for the ontology and interlinking is drawn from the SPARQL endpoints, as depicted in Figure 3.

An excerpt of the available methods is given in Table 2. In general, the REST API returns a set of spatial entities along with their RDF descriptions, which can be filtered in numerous ways:

- by area: Either a circular or rectangular area can be selected via WGS84 coordinates.
- by class: Returned resources can be restricted to a single LinkedGeoData class.
- by name (`rd:label`): It can be set whether returned points should contain or start with a certain string. Furthermore, it can be specified whether name search should be case sensitive and whether only names with a particular language tag should be considered.

Using area and label search combined with class restrictions were the most requested features in applications, which is why we provide them in the REST interface. The main purpose of the REST API is to lower the entry barrier for data consumers and to internally optimise the performance of the most commonly used queries.

## 6. Interlinking

In this section, the interlinking between *LinkedGeoData* and *DBpedia*, *GeoNames* and the *Food and Agriculture Organization of the United Nations (FOA)* is described. The interlinking is done on a per-class basis, where all instances of a set of classes of LGD are matched with all instances of a set of classes of another data source using labels and spatial information. As an example, cities in LGD and DBpedia are matched between all instances of `lgdo:City`, `lgdo:Town`, `lgdo:Village` and `lgdo:Hamlet` on one side and `dbo:Settlement` on the other. Only LinkedGeoData nodes are used for the matching as they have names as well as positions. In contrast, a LinkedGeoData way does not have a position itself, but has a potentially high number of nodes, each of which has a WGS84 position. It should be noted, however, that many ways in OpenStreetMap have reference points, e.g. characteristic points for a given way. Those reference points are not necessarily located in the geometric center of a way, but represent a typical point by OSM community consensus.

For each class-mapping, a link specification is created and executed using the *Silk Link Discovery Framework* [16]. The link specs usually include a metric, which is a linear classifier depending on the labels and the geographic distance, which can be calculated from the values of `wgs84:lat` and `wgs84:long` properties which are provided by all considered data sources. By combining classification, naming and spatial features, we are able to obtain very precise interlinking heuristics as shown later.

We used the following matching criterion, which we explain in detail below:

$$\frac{2}{3}s(a, b) + \frac{1}{3}g_c(a, b) > 0.95$$

- $a$  and  $b$  are the resources to be compared
- $s(a, b)$  is a the *Jaro-Winkler distance*, between the labels of  $a$  and  $b$ . If there are multiple labels, the



URLs relative to <a href="http://linkedgedata.org/triplify/near/">http://linkedgedata.org/triplify/near/</a> (General syntax and specific example)	Description
<code>&lt;latmin&gt;-&lt;latmax&gt;, &lt;lonmin&gt;-&lt;lonmax&gt;</code> 51.02-51.04, 13.72-13.74	Resources located in the given rectangle.
<code>&lt;lat&gt;, &lt;lon&gt;/&lt;radius&gt;</code> 51.02-51.04/1000	Resources located in specified radius in meters from the given point.
<code>&lt;lat&gt;, &lt;lon&gt;/&lt;radius&gt;/class/&lt;classname&gt;</code> 51.033333, 13.733333/1000/class/PlaceOfWorship	Resources in specified radius belonging to the given class.
<code>&lt;lat&gt;, &lt;lon&gt;/&lt;radius&gt;/class/ &lt;classname&gt;/label/&lt;lang&gt;/contains/&lt;label&gt;</code> .../class/Amenity/label/en/contains/flower	Resources in specified radius, belonging to the given class with a label in the specified language containing a specific string.

Table 2

Excerpt of the methods supported by the LGD REST API.

pair with the maximum score is chosen, ignoring the language-tag. While this could cause false links in the special case that the label of a resource in one language is very similar to the label of a resource in a different language, this type of error was not found in our evaluation. An advantage of this approach is that it works for several languages even if the proper language tags are actually missing.

- $c$  is the maximum distance that two points describing the same object are reasonably expected to differ. While a good value for  $c$  is easily chosen in some cases (a shop does not span more than a few hundred meters), it is nontrivial in cases of large variances in size such as in cities, mountains or islands. The value of  $c$  varies greatly between the classes and is explained by the choice of reference points, which can differ in each of the considered knowledge bases.

$$- g_c(a, b) = \begin{cases} 0 & d > c \\ 1/(1 + e^{-12d'+6}) & \text{otherwise} \end{cases} \quad \text{In}$$

this formula,  $d$  is the distance between  $a$  and  $b$ . The distance is approximated by the *haversine formula*, which uses a spherical model of the earth. We then define  $d' = 1 - d/c$  which is a linear function with a value of zero at distance  $d = c$  and one for  $d = 0$ . In order to not punish a slight discrepancy between two points as much as a linear function would,  $d'$  is not used directly. Instead, we employ a scaled logistic curve. The remaining parameters are adjusted such that two objects at distance  $c$  with exactly the same labels almost exactly matches the threshold of 0.95 in the formula above, which is the intended meaning of the parameter  $c$ .

### 6.1. Interlinking with DBpedia

Since the initial interlinking between LinkedGeoData and DBpedia as described in [1] in 2009, both knowledge bases have grown and changed significantly, resulting in the need of a new interlinking as well as an exhaustive re-evaluation of the quality of the interlinks. Table 3 shows the created class-mappings and the size of the linksets and their estimated precisions. The links were manually evaluated with a random sample of 250 instances each. In cases where the number of links is smaller than or only slightly higher than 250 as in the case of the universities, all of the instances were evaluated.

Table 3  
LinkedGeoData-DBpedia linksets.

DBpedia class	in-stances	LGD equivalent	c in km	nodes	links	precision
Airport	9520	Aerodrome	2.5	43734	8404	1.0
Settlement	239630	several <sup>1</sup>	0.1	620387	88377	1.0
Country	2505 <sup>2</sup>	Country	1000	231	222	0.991
University	11607	University	2.5	17715	268	1.0
Stadium	5539	Stadium	1	13001	133	1.0
School	22686	School	1	262566	2470	1.0
Island	2371	Island	100	31121	449	1.0
Mountain	8742	Peak	100	177702	3258	0.992
Overall	302600			1166457	103581	0.966

<sup>1</sup> City  $\cup$  Suburb  $\cup$  Town  $\cup$  Village

<sup>2</sup> The large number of countries is caused by former countries like *Republic of Texas* and *Inca Empire*.

## 6.2. Interlinking with GeoNames

The GeoNames database contains over 10 million geographical names and has 7.5 million unique features. It integrates sources such as the *National Geospatial-Intelligence Agency's (NGA)* and the *U.S. Board on Geographic Names*. While at the time of this writing there is no official SPARQL endpoint yet, an RDF-dump and an ontology are available. The ontology is very flat, with only two layers of disjunctive classes, where the superclasses are called *feature classes* and the subclasses *feature codes*. The feature codes are very detailed, for example there are 97 feature codes for the feature type  $\mathbb{T}$  (*Peak*). Linking GeoNames with LinkedGeoData makes this detailed features available to LinkedGeoData. In addition to the steps used for linking LinkedGeoData with DBpedia, the labels (which are represented by the properties `gn:name` and `gn:alternateName` in GeoNames) are first transformed by removing all occurrences of the name of class of the instances (e.g. “city”). This increases the string similarity score for pairs like (“Fananu”, “Fananu Island”). Again, 250 links per class were evaluated and the results are shown in Figure 4.

## 6.3. Interlinking with the Food and Agriculture Organization of the United Nations (FOA)

The FOA provides detailed information about organisations and countries from which the latter were chosen for interlinking. While it does not provide a latitude and a longitude, it provides official, list and short names and the names for the countries' currency and nationality in many languages. Also bordering countries, the gross domestic product, the agricultural area and a validation interval for former states such as the *Soviet Union* are given. This makes the FOA a very worthwhile target for interlinking. While FOA does not provide a SPARQL endpoint, the data was available as RDF which we uploaded on a local endpoint. Since no positional information is given, the spatial part of our matching formula is omitted and the properties `foa:shortName`, `foa:listName` and `foa:officialName` are used for string similarity matching. Between the 207 instances of `foa:self_governing` and the 231 instances of `lgdo:Country`, the linkset contains 191 links with a precision of 0.984.

## 6.4. Discussion

Overall, we generated 103 581 links to DBpedia, 571 642 to GeoNames and 191 to UN FAO. It should be noted that we aimed at a high precision of links at the cost of potentially lower recall, which we deem reasonable when establishing `owl:sameAs` links. We performed a comprehensive evaluation in which we manually verified 6 526 links. The average precision weighted by the number of links is 98.3%. In some cases, it was difficult to pick the best value for the parameter  $c$  described earlier in this section. In future work, we aim to control to precision-recall tradeoff more precisely via supervised machine learning techniques, which will potentially allow us to increase the number of links with only slightly less precision.

During our evaluation, we observed the following issues, which were responsible for some of the mistakes:

1. wrongly classified instances in data sources
2. part vs. whole relations (‘West Anvil Point’, ‘Anvil Point’),
3. part vs. another part relations (‘West Anvil Point’, ‘East Anvil Point’), (‘Red Wall Number 1’, ‘Red Wall Number 2’)
4. subtle spelling differences (‘Bären-Klippe’, ‘Beerensklippe’)

The first problem is a data quality issue and can only partially be solved on our side by helping to improve the involved knowledge bases. The other issues could be improved by a higher threshold, in particular for string similarity. However, we found out that this had a very negative effect on recall. The problem could be remedied by applying techniques like the *Stable Marriage Problem* to interlinking, which requires to incorporate support for this in the underlying interlinking tools and is subject to future work. A further problem, which we encountered in the matching problem was that despite several improvements in SILK, e.g. the introduction of blocking, the matchings still took several days to compute. We expect this time scale to shrink with new techniques, such as presented in [12] becoming available. This will, in turn, allow us to run more extensive tests with different parameter settings.

## 7. Live Synchronization

OpenStreetMap data is constantly being updated by its contributors. For instance, hundreds of shops are

Table 4  
Matching classes and created links between LGD and Geonames.

GeoNames feature class or code	number of features	LinkedGeoData class	c	number of nodes	links	precision
PCL ∪ PCLD ∪ PCLF ∪ PCLI ∪ PCLIX ∪ PCLS	237	Country	1 000 km	235	218	0.995
PRK	71 764	Park	5 km	151 833	55 648	0.992
PPL ∪ PPLA ∪ PPLA2 ∪ PPLA3 ∪ PPLA4 ∪ PPLC ∪ PPLF ∪ PPLG ∪ PPLL ∪ PPLQ ∪ PPLR ∪ PPLS ∪ PPLW	2 821 405	Hamlet ∪ Village ∪ Town ∪ City	100 km	818 893	-	<sup>1</sup>
SCH	224 217	School	1 km	340 039	168 545	1.0
PRK	72 130	Park	5 km	157 862	55 648	0.992
STDM	753	Stadium	1 km	13 001	24	1.0
FRM ∪ FRMQ ∪ FRMS ∪ FRMT	207 171	Farm	6 000 m	3 834	54	1.0
AIRH ∪ AIRP ∪ AIRQ ∪ AIRB ∪ AIRF	32 449	Airport ∪ Aerodrome ∪ Aerialway ∪ Aeroway	10 km	175 006	21 552	1.0
MALL ∪ MKT	18 240	Supermarket ∪ Shop ∪ Mall	1 km	572 833	59	0.949
TMPL ∪ CH ∪ CTRR	231 678	PlaceOfWorship	1 km	352 673	201 318	0.976
REST	1 195	Restaurant	1 km	167 293	55	1.0
HTL	82 876	TourismHotel	200 km	63 516	2 214	0.958
HSP	16 606	Hospital	5 km	58 095	11 032	0.976
PO	31 244	PostOffice	1 km	50 962	20 718	1.0
GDN	380	Garden	1 km	35 542	11	1.0
PP	1 209	Police	1 km	28 363	24	1.0
LIBR	10 712	Library	1 km	25 637	9 225	1.0
SHRN	16 379	Memorial	100 m	22 613	168	1.0
MUS	4 409	TourismMuseum	2 km	21 442	3 291	0.996
CLF	7 668	Cliff	2 km	18 738	4 414	1.0
UNIV	363	University	2 km	17 715	48	0.896
BAY	45 230	Bay	5 km	16 595	14 670	1.0
BCHS ∪ BCH	7 533	Beach ∪ TourismBeach ∪ NaturalBeach	10 km	14 129	2 028	1.0
CSTL	3 615	Castle	2 km	8 406	252	1.0
RECG	6 288	GolfCourse	5 km	6 880	51	1.0
GLCR	6 471	Glacier	10 km	6 495	375	1.0
Overall (without cities)	100 817			2 329 737	571 642	0.990

<sup>1</sup> As the matching takes several days for the large classes, there is no data for cities yet. It will however be there in the camera-ready version.

added, removed or updated every day. Static snapshots of this data cannot reflect such recent changes, which makes them unsuitable for use cases where users need up-to-date information. As a solution to this problem, we implemented a live-synchronization module, which converts the minutely changesets published by OpenStreetMap to RDF and updates a triple store accordingly. Additionally, we publish our changesets in an intuitive way that enables user of the LinkedGeoData service to synchronize their own RDF store with it.

In the remainder of this section we first briefly describe general requirements we pose on the update procedure. Afterwards, we explain the changeset formats of OpenStreetMaps and LinkedGeoData. Finally, we discuss concrete cases that must be considered by our live-sync module and give a sketch of the algorithm.

### 7.1. General requirements

Our major design goals for the live sync procedure were high performance and cleanliness: On the one

hand, the update procedure must be capable of processing minutely changesets from OpenStreetMap in much less than a minute in order to catch up any lag to OpenStreetMap. On the other hand, the updates should not leave our store in a dirty state - i.e. upon a modification or deletion of an OSM entity all RDF statements about the corresponding resources must reflect the entity's most recent state, and no left-over statements of a previous state must remain. Meeting both demands results in a non trivial procedure.

## 7.2. Changeset formats

We first explain the format of changesets provided by OpenStreetMap, and the format of our published RDF changesets. This eases the understanding of the requirements and details of the live sync procedure that are explained in the sequel.

OpenStreetMap publishes changesets as sequentially numbered files in the XML-based OSM-Change (OSC) format. For instance, changeset #786001 is published at `<base-path>/000/786/001.osc.gz`.

The root of an OSC document is formed by the *osmChange*-element, whose immediate children may be any number of occurrences of *create*, *modify*, and *delete* elements. Each of these elements then contains a number of OSM entities that were changed, as shown in Listing 4.

Listing 4: Example of an OSM change file.

```

<!-- The attributes timestamp, uid, user, and
changeset are omitted in this example -->
<osmChange version="0.6" generator="Osmosis 0.37">
  <modify>
    <node id="1" version="5" lat="50" lon="8" .../>
    <node id="2" version="5" lat="51" lon="8" .../>
    <node id="3" version="5" lat="50" lon="9" .../>
  </modify>
  <create>
    <way id="1" version="5" ...>
      <nd ref="1"/>
      <nd ref="2"/>
      <nd ref="3"/>
      <tag k="amenity" v="school"/>
      <tag k="name:en" v="Mountain School"/>
    </way>
  </create>
  <delete>
    <node id="4" version="5" lat="50" lon="9" .../>
    <tag k="created_by" v="Merkaartor 0.12"/>
  </node>
</delete>
</osmChange>

```

The interpretation of the data in the context of *create*, *modify* and *delete* is as follows:

- Create: The state of the newly created entity.
- Modify: The state after the modification.

- Delete: The state prior to the deletion.

There are two things worth noting: Firstly, changes are not given on a per-tag, but on a per-entity basis and, secondly, the prior state to a modification is not given in the OSC file.

Whenever the LGD live sync module processes an OSC file with a sequence number *s*, it publishes two N-Triples files containing the added and removed triples, namely *s.added.nt.gz* and *s.removed.nt.gz*. As a result, verification whether our changesets are correct can be done by examining the corresponding *.osc* file.

Since the RDF-based live sync operates on a per-statement basis, but changes are given on a per-entity basis this implies that during the live sync many queries for checking the states of entity are necessary.

## 7.3. Observations

In this part, we present the key aspects that need to be considered for a synchronization procedure that meets our requirements. We classify them according to whether they are general, or pertain to the changes of nodes or ways.

### Common aspects

- Filtering: A vast amount of data is changed on OpenStreetMap every minute. Our experience with DBpedia [15] was that processing large amounts of changes in RDF can cause severe performance issues with triple stores. In order to be performance-wise on the safe side we decided from the beginning to put filters in place. This enables us to trade the completeness of the coverage of the data for performance by adjusting the amount of changes that will be processed.
- Relevance: Any update should leave the store only with relevant data, according to the filter configuration. This prevents the store from growing too large as updates are being applied, and also prevents users from receiving “dirty” answers to queries, such as wayNodes that are no longer connected to a way.
- Modifications: In the event of modifications, we do not get an entities state prior to the change. Therefore, we need to query our store for each modified entity in order to compute the changeset.

### Node-based aspects

- Repositioning of nodes: When a node position is changed, the polygons/linestring property of

all referencing ways needs to be updated accordingly.

- Deletions and Modifications: Whenever a node is deleted or modified and fails the relevance test it will be removed - unless it is referenced by a relevant way.

#### Way-based aspects

- Whenever a way is created or modified, it may contain references to nodes that are not in the changeset (as the points themselves were not changed). This makes it necessary to keep track of *all* to points, as every point may at some point be connected to way.
- LineStrings and Polygons: For each way the corresponding linestring or polygon must be assembled.
- For every relevant way, all its referenced nodes also need to be loaded.
- Irrelevant nodes that are referenced by relevant ways should not carry any information except for their position. Such nodes should not even be explicit instances of *Igdo:Node* in order to avoid many non-interesting triples which would increase the dataset size and reduce performance.
- Whenever a way is modified, it may be no longer relevant, and therefore needs to be removed. Whenever a way is removed, all nodes which are not relevant by themselves also need to be removed.

#### 7.4. Algorithm

Our live-sync algorithm is given in Listing 1 and explained as follows. Essentially, for each entity we need to determine its state before and after its modification. By this we can figure out the triples, which need to be added or removed from the store. Recall that we need to keep track of all node-positions because every creation or modification of a way might introduce a reference to it. Rather than creating triples for more than a billion node positions, we chose to keep the nodes' positions in a separate relational database, which we refer to as the *node store*. We load node positions into the triple store as needed. The *fetchRDF\_Node* and *fetchRDF\_Way* functions query the triple store for the previous state of an entity, whereas the corresponding *generateRDF* functions generate the new state. Note that in the case of ways this also involves all triples of the way's node-list (see Listing 3). The *shape triple* is the one stating the polygon or linestring of a way,

#### Algorithm 1. LinkedGeoData Live-Sync algorithm

**Input:** A changeset  $C$

**Output:** The sets *Additions* and *Removals* corresponding to the triples that need to be added and removed, respectively.

```

1: Let:  $N \leftarrow \emptyset, O \leftarrow \emptyset$ 
2: for all nodes  $n$  in  $C$  do
3:   if created( $n$ ) then
4:     Insert ( $n.id, n.position$ ) into node store
5:     if relevant( $n$ ) then
6:        $N \leftarrow N \cup generateRDF\_Node(n)$ 
7:     end if
8:   else if modified( $n$ ) then
9:     Update ( $n.id, n.position$ ) in node store
10:     $O \leftarrow O \cup fetchRDF\_Node(n)$ 
11:    if relevant( $n$ ) then
12:       $N \leftarrow N \cup generateRDF\_Node(n)$ 
13:    end if
14:    for all ways  $w$  where  $n$  is a member do
15:       $st_o \leftarrow fetchShapeTriple(w)$ 
16:       $O \leftarrow O \cup st_o$ 
17:       $st_n = createNewShapeTripleWithPositionReplaced(st_o, n)$ 
18:       $N \leftarrow N \cup st_n$ 
19:    end for
20:   else if deleted( $n$ ) then
21:     Remove entry for ( $n.id$ ) from the node store
22:      $O \leftarrow O \cup fetchRDF\_Node(n)$ 
23:   end if
24: end for
25: for all ways  $w$  in  $C$  do
26:   if created( $w$ ) then
27:     if relevant( $w$ ) then
28:        $m \leftarrow fetchNodePositionMap(w.nodeRefs)$ 
29:        $N \leftarrow N \cup generateRDF\_Way(w, m)$ 
30:     end if
31:   else if modified( $w$ ) then
32:      $w_o \leftarrow fetchRDFWay(n)$ 
33:      $O \leftarrow O \cup w_o$ 
34:     if relevant( $w_o$ ) and not relevant( $w$ ) then
35:       RemoveIrrelevantNodes( $w_o.nodeRefs$ )
36:     end if
37:     if relevant( $w$ ) then
38:        $m \leftarrow fetchNodePositionMap(w.nodeRefs)$ 
39:        $N \leftarrow N \cup generateRDF\_Way(w, m)$ 
40:     end if
41:   else if deleted( $w$ ) then
42:      $O \leftarrow O \cup fetchRDF\_Way(n)$ 
43:     RemoveIrrelevantNodes( $w.nodeList$ )
44:   end if
45: end for
46: procedure REMOVEIRRELEVANTNODES( $nodes$ ) ▷
47:   for all nodes  $n$  in  $nodes$  do
48:      $d \leftarrow fetchRDF\_Node(n)$ 
49:     if not relevant( $d$ ) then
50:        $O \leftarrow O \cup d$ 
51:     end if
52:   end for
53: end procedure
54:  $Additions \leftarrow N \setminus O$ 
55:  $Removals \leftarrow O \setminus N$ 

```

and is updated accordingly on changes. The major optimizations are based on caching: We keep last recently used maps of the node positions and the state of resources in order to reduce the amount of database lookups, which speeds up the fetch functions. The caches are updated accordingly when changes are written to the triple store and node store.

### 7.5. Filtering

We use a simple filtering system where entities must pass the following three tag-based filters before their corresponding RDF data may end up in the dumps and SPARQL endpoints:

- EntityFilter: Rejects entities with at least one blacklisted tag.
- TagFilter: Removes all blacklisted tags from an entity.
- RelevanceFilter: Only accepts entities with certain white-listed tags.

For instance, in the current release the entity filter rejects all entities with a tag whose key equals 'railway', unless the corresponding value is 'station', 'halt' or 'tram\_stop'. By this, we rule out more than 160K nodes and 710K ways. As an example for the tag filter, we reject the `created_by` tag which seems to carry little information. As a result, just by considering nodes we can already omit approximately 20mio triples for the most frequently used value "JOSM". The relevance filter was introduced as it was noticed that only blacklisting certain tags still results in a lot of seemingly non-interesting data to get processed. The complete filter configuration is published together with each release. As a final filtering step, we reject ways with more than 20 nodes, since each node reference of a way results in two triples: one for the node-way membership and one for the node position.

## 8. Statistics

In this section we outline statistics about three things: 1) the usage of the LinkedGeoData service, 2) the LinkedGeoData dataset and 3) performance of the Live-Sync.

For determining the usage of LinkedGeoData, we evaluated the usage of both of our SPARQL-endpoints (static and live) in the time from from Nov 2010 until April 2011, i.e. after they were made publicly available. In this timespan, they were queried a total of 127.000 times from 422 distinct machines<sup>23</sup>. The top ten machines were responsible for 73% of those queries. More than 1.000 queries were issued by 19 of them. Figure 4 shows the number of queries per day. The diagram indicates that the usage of the LGD service has been increasing. However, whether the high

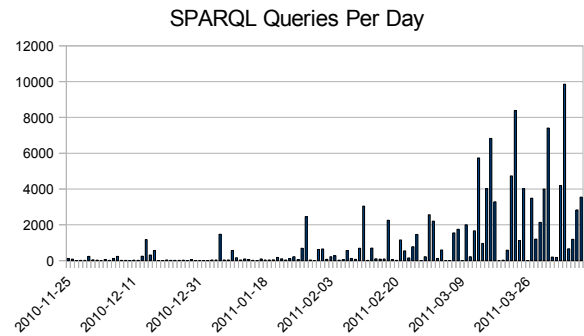


Fig. 4. Usage of the SPARQL endpoints.

query counts towards the end remains at that level is yet to be evaluated.

The current LGD release dataset contains about 65 million triples corresponding to about 6.3 million nodes and 66 million triples corresponding to 7.1 million ways. Table 5 gives an overview of selected instance counts in the static SPARQL endpoint, and their increase in number in LGD live one after processing changesets corresponding to roughly three weeks.

class	#instances (static)	#instances (live)
Ways	7 132 373	7 334 925
Nodes	6 251 067	7 022 481
Stream	2 377 952	2 419 467
Parking	520 901	537 477
Village	516 547	522 570
Shop	497 820	519 164
Hamlet	415 609	424 179
School	361 239	366 070
PlaceOfWorship	359 563	363 225
Restaurant	173 350	177 888
FastFood	67 980	69 772
Pub	67 279	68 279

Table 5

Comparison of the static dump from April 6th with the live data at April 30th.

Regarding LGD live sync performance, we measured the following values: On average, the processing time of a single minutely OSM changeset takes 5 seconds with our filter configuration. Between April 6 and April 30, about 40 000 changesets were processed, each of them corresponding to an average addition of 620 and removal of 42 triples affecting 102 distinct resources.

In the initial LGD release of 2009, there were 50 object properties. However most of them were consid-

<sup>23</sup>Not counting the queries from our own network.

ered to be better suited as classes, resulting in the relatively low number of only 9 object properties in the current release.

## 9. Tools using LinkedGeoData

### 9.1. LGD Browser

In order to showcase the benefits of revealing the structured information in OSM, we developed a facet-based browser and editor for LinkedGeoData (see Figure 9)<sup>24</sup>. It allows to browse the world by using a slippery 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.

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.

To resolve this problem and compute facets more efficiently, we established a quadtile (also called z-curve) index over OSM data. Such an index combines latitude and longitude into a single bitstring, which can then be efficiently indexed. Once each point can be associated to a tile and 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.

Furthermore, there are several new smaller LGD browser features compared to its previous version described in [1]. For instance, an RDF export of the current map selection including its facets can now be performed. This allows to easily extract a relevant fragment of LinkedGeoData for use within other tools. For each point on the map, its RDF source can be retrieved and it can be edited on OpenStreetMap. The browser has been extended by a search function powered by *OpenStreetMap Nominatum*. The facet support has been extended to object properties, i.e. values of those properties can now be restricted in the facet selection. Finally, the LGD browser now provides a permanent link feature.

### 9.2. STEVIE

STEVIE<sup>25</sup> [3] is an application developed by the Institute for Web Science and Technologies at the University of Koblenz, which uses LinkedGeoData. STEVIE allows to create and edit points of interests (POIs) (see Figure 9.2) and annotate them semantically. The annotations use the LinkedGeoData ontology and are also interlinked to DBpedia. The annotations allow to employ clustering techniques in STEVIE, which are used to group sets of similar objects within the limited screen size of a mobile phone. The application allows the creation of events and, therefore, combines spatial and temporal information. An emphasis is put on providing an intuitive user interface for navigating those two dimensions. In order to display POIs and classify them, STEVIE uses the LinkedGeoData REST interface, ontology and SPARQL endpoint.

### 9.3. BeAware

BeAware<sup>26</sup> is a website, which allows to manage events and integrates them with geographic information. It uses its own ontology for events and integrates LinkedGeoData for choosing locations. In particular, the curated ontology of LinkedGeoData provides ben-

<sup>24</sup>Available online at: <http://browser.linkedgedata.org>

<sup>25</sup><http://tiny.cc/stevie10>

<sup>26</sup><http://beaware.at/>

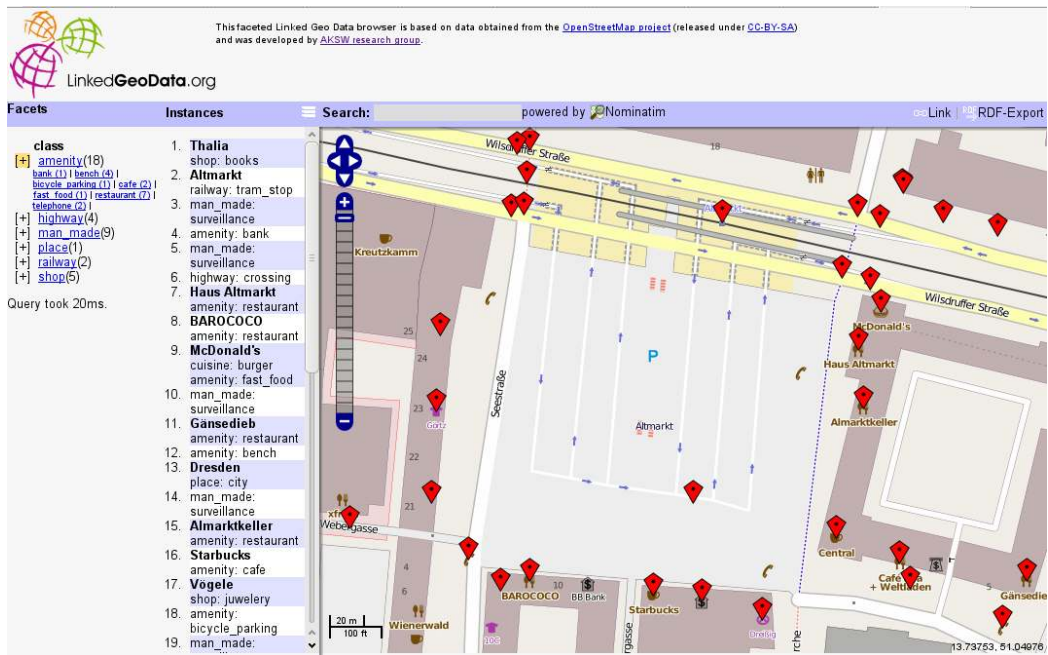


Fig. 5. LinkedGeoData Browser.

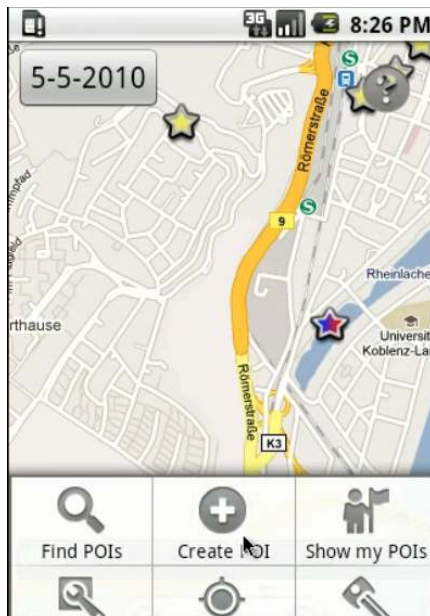


Fig. 6. Creation of a point of interest in STEVIE. The application includes a temporal dimension and highlights POIs where events take place in the selected time span.

efits for the application<sup>27</sup>: “First of all, LinkedGeo-

Data ontology that connects all OpenStreetMap categories and properties excellently suits our interface of new place choosing (in addition, it allows to use inference engine, for example, for retrieving buildings of all types).” Figure 9.3 shows a screenshot for choosing the location of an event. An advantage gained by this association is that it facilitates querying for events at a particular location or within a particular city. In addition, in some cases further information about the location from an interlinked data source is available and can be presented to the user.

#### 9.4. Layar

Layar<sup>28</sup> is an augmented reality browser for mobile phones. Within Layar, a LinkedGeoData layer was developed. This allows to view the surrounding objects of a person via the mobile phone camera. The LinkedGeoData ontology is used to classify objects and map them to displayed icons. The layer uses `rdfs:label`, which is aggregated from several tags in OpenStreetMap, to display the name of an object. Further triples describing an object are shown in a detail view.

<sup>27</sup><http://alexidsa-en.blogspot.com/2010/06/rdf-vs-nonrdf-for-geodata-at-beaware.html>

<sup>28</sup><http://www.layar.com>



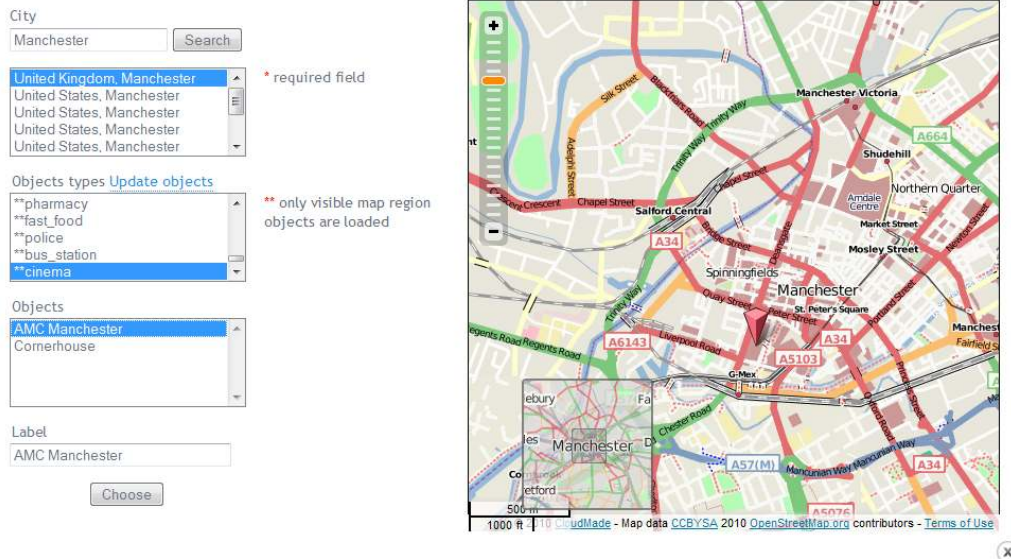


Fig. 7. Marking the location of an event in BeAware.

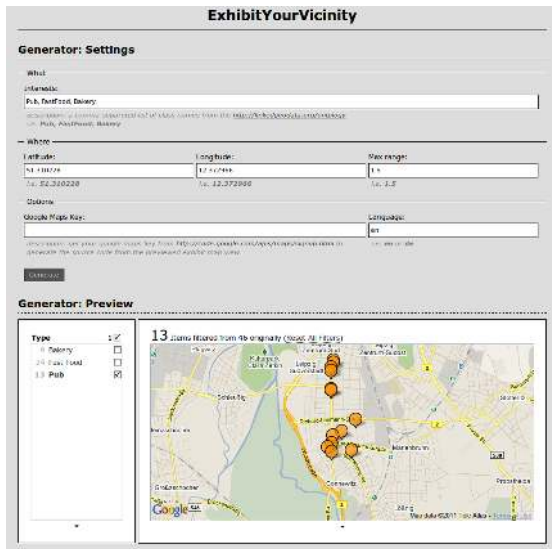


Fig. 8. Vicibit is a tool to generate custom views on LinkedGeoData via Exhibit. The example shows a faceted view on nearby pubs, bakeries and shops. The code generated by Vicibit can easily be pasted into blogs, forums and web pages.

9.5. Vicibit

Vicibit<sup>29</sup> (“exhibit your vicinity”) is a tool working on top of LinkedGeoData Live, which allows to create customised views on LinkedGeoData. It allows users

to enter classes in the LinkedGeoData ontology they are interested in as well as a default map section, which should be displayed. The tool then generates HTML code, which creates a map displaying all items belonging to the selected classes as well as the ability to filter by facets. Technically, this is realised by applying the Exhibit framework<sup>30</sup> on data in the LinkedGeoData Live SPARQL endpoint. A typical use case is that a webpage or blog entry describing a particular event can be enriched with a map of nearby pubs and other shops (see Figure 8).

10. Related Work

We split related work in three parts: First, we describe initiatives for integrating spatial information in the Web of Data. Afterwards, we summarize work on techniques for converting relational databases to RDF, which is the task we had to face in LinkedGeoData. Finally, we give pointers to interlinking frameworks and explain our choice of using SILK and LIMES.

10.1. Spatial RDF Datasets

In the following, we describe spatial data sets, which are available as RDF and we consider important.

<sup>29</sup><http://vicibit.linkedgedata.org>

<sup>30</sup><http://www.simile-widgets.org/exhibit/>

Ordnance Survey<sup>31</sup> is the national mapping agency in Great Britain. Over the past years, they released some of their products as Linked Data<sup>32</sup>. Ordnance Survey provides very accurate high-quality data and represents a major contribution to the spatial data web. In a comparison between Ordnance Survey data [9] focused on England and London in particular, OSM data was, however, also fairly accurate. A main difference between both efforts is that OpenStreetMap, and thereby LinkedGeoData, are world-wide community-based approaches.

GeoNames is a comprehensive global spatial database containing several million entities. This data has been converted to RDF<sup>33</sup> and is provided as Linked Data. GeoNames provides RDF properties for navigating spatial hierarchies (parent/child), publishes postal codes, labels, population figures, type information (via feature codes) and other properties of spatial entities. Due to this wealth of information, we provided a fine-grained interlinking between LinkedGeoData and GeoNames as described in Section 6. A difference between GeoNames and OpenStreetMap is that OSM allows free tags, which makes it easier to extend. For instance, shops in OSM sometimes (specifically 60 thousand times as of April 2011) contain opening hours. Another example is the wheelchair tag, used 24 thousand times, which indicates whether or not a spatial entity is accessible via a wheel chair. OpenStreetMap also has a larger community than GeoNames with several hundred thousand users and more fine-grained data, which even include traffic lights and trash bins.

The United Nations FAO (Food and Agriculture Organisation) Geopolitical Data [5] provides RDF descriptions of countries and other political units as well as relations between them. While it contains only a small number of instances (298 in May 2011), it provides very detailed information on those instances. For this reason, we decided to provide interlinks with UN FAO.

GeoLinkedData.es is an open initiative to provide Spanish geospatial data [2]. It focuses on hydrography features and integrates several existing data sources.

NUTS (Nomenclature of Territorial Units for Statistics) provides a hierarchical system for describing the economic territory of the European Union<sup>34</sup>. The NUTS hierarchy is established by EuroStat. EU NUTS data has been converted to RDF<sup>35</sup>. It allows to explore the hierarchy via Linked Data, e.g. a possible path along the “partOf” property is Inner London East → Inner London → London → UK.

## 10.2. Relational Database to RDF Conversion and Mapping

Converting relational databases to RDF is a significant area of research with several approaches published and tools available. In particular, there is a W3C RDB2RDF working group, which aims to standardize a database to RDF mapping language [6]. Instead of providing an in-depth overview, we refer to recent surveys [13,14] and overviews<sup>36</sup> on this topic. There are various tools available implementing the surveyed approaches such as D2R, Triplify, DartGrid, DataMaster, MapOnto, METAmorphoses, ODEMapster, RDBToOnto, RDOLE, Virtuoso RDF Views and VisAVis. For LinkedGeoData, we decided to use a custom mapping solution as described in Section 4, despite the number of available conversion tools. The reason for this choice was the particular tag structure of OSM, which allows us to provide a highly flexible schema as well as handle a very high amount of data via our approach.

## 10.3. Interlinking and Ontology Mapping

There have been several decades of research starting with the integration of different database schemata. Tools like COMA [7] provide rich support for various matching operations between databases as well as between RDF knowledge bases. [4] 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

<sup>31</sup><http://www.ordnancesurvey.co.uk>

<sup>32</sup><http://data.ordnancesurvey.co.uk>

<sup>33</sup><http://www.geonames.org/ontology/>

<sup>34</sup>[http://epp.eurostat.ec.europa.eu/portal/page/portal/nuts\\_nomenclature/introduction](http://epp.eurostat.ec.europa.eu/portal/page/portal/nuts_nomenclature/introduction)

<sup>35</sup><http://rdfdata.eionet.europa.eu/ramon/nuts2008/>

<sup>36</sup><http://esw.w3.org/topic/Rdb2RdfXG/StateOfTheArt>

illustrate the idea. [11] describes an approach integrating geo data from multiple sources, which also incorporates a temporal dimension. For interlinking LinkedGeoData, we mainly searched for instance matching tools, since our main goal is to match specific points of interests in different knowledge bases. In this area, SILK [16] and LIMES are the most widely used applications. We extended SILK with an appropriate metric for matchings based on WGS84 distance between points, which was later included in the official SILK release. A main benefit for SILK as well as LIMES, which we both use, is their ability to handle large volumes of data and use SPARQL endpoints as input source.

## 11. Conclusions and Future Work

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 and GeoNames as the central interlinking hubs for spatial information on the Web of Data. Despite the recent advances in RDF data management, it became clear during our work on LinkedGeoData that spatial data of the size of OpenStreetMap still poses a major challenge wrt. scalability. Substantial engineering effort was required to optimize the performance of the querying interfaces, live synchronisation as well as the interlinking.

In the future, we plan to execute SPARQL queries directly on the relational database by employing an RDB-RDF mapping, which translates incoming SPARQL queries into SQL queries. Although in this regard substantial progress was made during the last years and implementations are now more robust, scalability is still an issue preventing a direct deployment in case of LinkedGeoData. Another stream of future work is the better support for geometries according to the current NeoGeoVocabulary development<sup>37</sup>, which we are supporting. A semantic misrepresentation currently found in LinkedGeoData for example is the missing separa-

tion of geometry and features. In the future, we plan to attach geometries to entities, i.e. points of interest, instead of identifying both.

## Acknowledgements

The authors would like to thank OpenLink for providing an enterprise edition of the Virtuoso database system that offers support for spatial SPARQL queries. Furthermore, the authors would like to thank the members of the LinkedGeoData community and 3rd party application developers for their feedback. In particular, we would like to thank Robert Schulze for his work on Vicibit. This work was supported by a grant from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and LATC (GA no. 256975).

## References

- [1] S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData - adding a spatial dimension to the web of data. In *Proc. of 8th International Semantic Web Conference (ISWC)*, 2009.
- [2] L. M. V. Blázquez, B. Villazón-Terrazas, V. Saquicela, A. de León, Ó. Corcho, and A. Gómez-Pérez. Geolinked data and INSPIRE through an application case. In D. Agrawal, P. Zhang, A. E. Abbadi, and M. F. Mokbel, editors, *GIS*, pages 446–449. ACM, 2010.
- [3] M. Braun, A. Scherp, and S. Staab. Collaborative creation of semantic points of interest as linked data on the mobile phone. In *Extended Semantic Web Conference (Demo Session)*. Springer, 2010.
- [4] D. F. Brauner, C. Intrator, J. C. Freitas, and M. 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.
- [5] C. Caracciolo, M. I. Sucasas, and J. Keizer. Towards interoperability of geopolitical information within FAO. *Computing and Informatics*, 27(1):119–129, 2008.
- [6] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. World Wide Web Consortium, Working Draft WD-r2rml-20110324, Mar. 2011.
- [7] H. H. Do and E. Rahm. COMA - A system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. Morgan Kaufmann, 2002.
- [8] M. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, Aug. 2007.
- [9] M. Haklay. How good is volunteered geographical information? A comparative study of openstreetmap and ordnance survey datasets. July 2010.
- [10] J. Lehmann, C. Bizer, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

<sup>37</sup><http://geovocab.org/doc/neogeo.html>

- [11] H. Manguinhas, B. Martins, and J. L. Borbinha. A geotemporal web gazetteer integrating data from multiple sources. In *ICDIM*, pages 146–153. IEEE, 2008.
- [12] A.-C. Ngonga Ngomo and S. Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.
- [13] S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. T. Jr, S. Auer, J. Sequeda, and A. Ezzat. A survey of current approaches for mapping of relational databases to rdf, 01 2009.
- [14] D.-E. Spanos, P. Stavrou, and N. Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web Journal*. under review.
- [15] C. Stadler, M. Martin, J. Lehmann, and S. Hellmann. Update Strategies for DBpedia Live. In *6th Workshop on Scripting and Development for the Semantic Web Colocated with ESWC 2010 30th or 31st May, 2010 Crete, Greece*, 2010.
- [16] J. Volz, C. Bizer, M. Gaedke, and G. 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.

## Appendix

### A. Prefixes Used

The following prefixes are used in the paper:

1	lgd:	<a href="http://linkedgeodata.org/triplify/">http://linkedgeodata.org/triplify/</a>
2	lgdo:	<a href="http://linkedgeodata.org/ontology/">http://linkedgeodata.org/ontology/</a>
3	wgs84:	<a href="http://www.w3.org/2003/01/geo/wgs84_pos#">http://www.w3.org/2003/01/geo/wgs84_pos#</a>
4	foa:	<a href="http://www.fao.org/countryprofiles/geoinfo/geopolitical/resource/">http://www.fao.org/countryprofiles/geoinfo/geopolitical/resource/</a>
5	dbpedia:	<a href="http://dbpedia.org/resource/">http://dbpedia.org/resource/</a>
6	rdf:	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
7	rdfs:	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
8	owl:	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>
9	xsd:	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>
10	georss:	<a href="http://www.georss.org/georss/">http://www.georss.org/georss/</a>