LOD2 Deliverable D5.1.1: Initial release faceted spatial-semantic browsing component

Claus Stadler, Jens Lehmann, Konrad Höffner, Sören Auer

Abstract: This prototype deliverable consists of a software release of JavaScript widgets for faceted spatial-semantic browsing and an accompanying deliverable report. The software is open source and can be downloaded at https://github.com/AKSW/SpatialSemanticBrowsingWidgets. The widgets are being used in the LinkedGeoData project. This deliverable describes these widgets in more detail.

Collaborative Project

LOD2 - Creating Knowledge out of Interlinked Data

Project Number: 257943 Start Date of Project: 01/09/2010 Duration: 48 months

Deliverable 5.1.1
Initial release faceted spatial-semantic browsing component

Dissemination Level	Public
Due Date of Deliverable	Month 12, 31/08/2011
Actual Submission Date	31/08/2011
Work Package	WP5, Adaptive Linked Data Visualization, Browsing and Authoring
Task	Task T5.4
Type	Prototype
Approval Status	Approved
Version	1.0
Number of Pages	29
Filename	deliverable-5.1.1.pdf

Abstract: This is a deliverable accompanying a software release on widgets for faceted spatial-semantic browsing. The widgets will be integrated into the LOD2 Stack as well as into the Open Government Data repository developed in the LOD2 Use Case WP9.

Project funded by the European Commission within the Seventh Framework Programme (2007 – 2013)

History

Version	Date	Reason	Revised by
0.1	2011-08-25	Initial version	Claus Stadler
1.0	2011-08-31	Final version	Claus Stadler, Sören Auer

Author list

Organisation	Name	Contact Information
ULEI	Claus Stadler	cstadler@informatik.uni-leipzig.de
ULEI	Jens Lehmann	lehmann@informatik.uni-leipzig.de
ULEI	Konrad Höffner	konrad.hoeffner@uni-leipzig.de
ULEI	Sören Auer	auer@informatik.uni-leipzig.de

Executive summary

This prototype deliverable consists of a software release of JavaScript widgets for faceted spatial-semantic browsing and an accompanying deliverable report. The software is open source and can be downloaded at `https://github.com/AKSW/SpatialSemanticBrowsingWidgets`. The widgets are being used in the LinkedGeoData project. This deliverable describes these widgets in more detail.

Contents

1 Goal and Scope

This document summarises the release of the *faceted spatial-semantic browsing widgets*. It describes the software prototype developed by the participants with respect to Task 5.4. We plan to integrate the widgets into the LOD2 Stack (as developed in WP1 and WP5) as well as into the Open Government Data repository (as developed in the Use Case WP9).

1.1 Availability

The source code is available at:
`https://github.com/AKSW/SpatialSemanticBrowsingWidgets`

The widgets are used in the browser of the LinkedGeoData project[1], available at:
`http://browser.linkedgeodata.org`.

2 Widgets

2.1 Overview

In accordance with the description of work, we implemented the following three widgets, that are depicted in Figure 1.

- *Map Widget*: This widget is used to display a geographical map with markers indicating the locations of points and polygons of interest. It is based on OpenLayers[2] and uses the tiles rendered by OpenStreetMap[3] as the default

[1] `http://linkedgeodata.org`
[2] `http://openlayers.org`
[3] `http://openstreetmap.org`

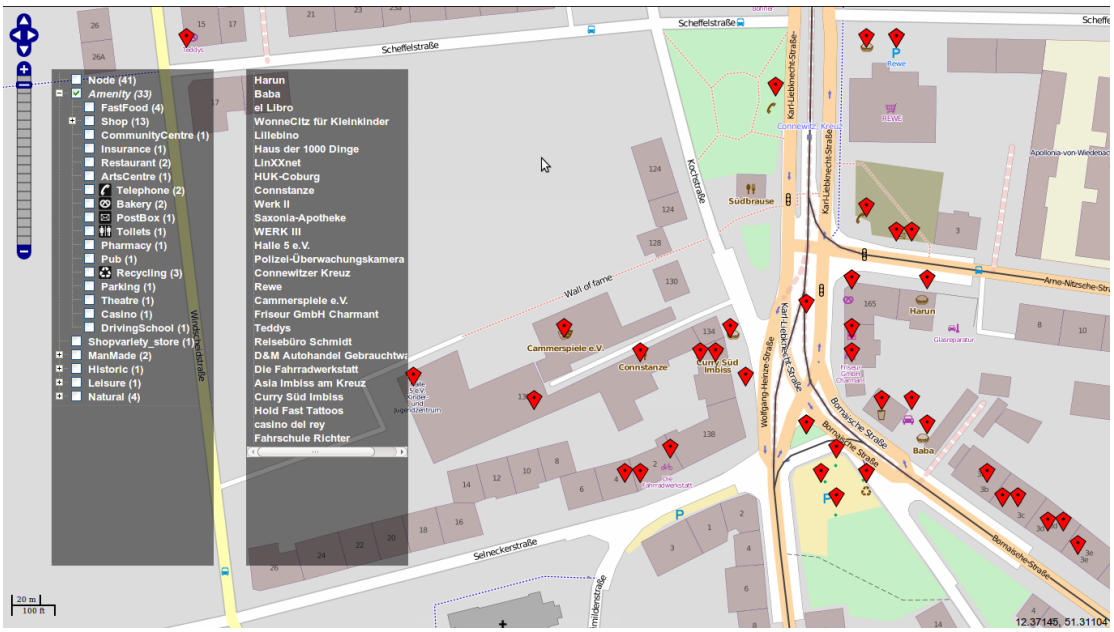

Figure 1: Screenshot of the widgets.

map layer. Clicking on an object of interest will display detailed information about it.

- *Facet Widget*: This widget serves two purposes: The first one is to display the class and property hierarchy based on the instances in the currently visible area. The second one is to allow one to set filters on the classes and properties. Only resources that match the chosen filter criteria are displayed in the map view.

- *Result Widget*: The result view displays a list of the labels and most specific types of the resources visible in the map view.

2.2 Architecture

In order to make the widgets reusable, the following design decisions were made: The widgets are written in pure JavaScript, and only require a SPARQL Endpoint for data access. Ideally, such an endpoint has *Cross-Origin Resource Sharing* (CORS)[4] enabled, which means that a client browser may directly perform cross domain requests to it.

Furthermore, the widgets operate on local spatial regions, meaning that they do not depend on global meta-data about the data in the SPARQL endpoint. In order to cleanly synchronize the widgets, they depend on a set of model classes, which fire events when their state changes. The following model classes exist:

[4] `http://www.w3.org/TR/cors`

- *Visible Area*: Objects of this class represent the state of the visible rectangular area in a map widget. As soon as the visible area changes, all three widgets will be notified. As a result, they will fetch the necessary data (information about the visible instances and their meta-data) and update their views accordingly. For instance, the facet widget will send a SPARQL query in order to retrieve information about the classes and object properties in the area.

- *Visible Instances*: This class keeps track of the positions of resources within the visible area. The map widget will display markers for these resources.

- A *Filter Configuration* object represents the state of the chosen filter criteria. A change in the filter criteria causes the map view and result view to update their display.

- *Class and Property Hierarchy Cache*: These objects keep track of a client side snapshot of the class and property hierarchy in the SPARQL endpoint: Whenever an instance is encountered whose classes or properties have not yet been categorized into this snapshot, the missing information, such as the parent classes, will be fetched from the SPARQL endpoint. The hierarchy in the facet view is computed from this snapshot.

- *Resource-Icon and Resource-Label Map*: These maps serve the purpose of allowing lazy-loading of the icons and labels of resources: For classes and properties, the corresponding icons and labels only need to be queried once as they can be cached for subsequent requests. Therefore, once the caches are warm, there is no need to create SPARQL queries for this kind of information. However, in the case of cold caches, the data can be fetched asynchronously, and the widgets can update their display as soon as the data becomes available.

3 Limitations

The widgets currently expose the following limitations:

- On large geospatial knowledge bases, such as LinkedGeoData, the visible area must not become too large in order for the queries generated by the widgets to return quickly. For instance, if one wanted to query all airports in Germany, the SPARQL endpoint would have to scan through multiple million entities[5], resulting in infeasible query times. A possible solution which we might explore in the future is to have different SPARQL endpoints for different zoom levels.

[5]Even if a geospatial index restricted the search e.g. to Germany, then still all entities within this border have to be checked for whether they are airports.

- Currently only Virtuoso is supported. For the final release we plan to add support for other stores.
- There is currently neither support for RDFa templates nor support for a personalization of the map view. They are planned for the final release in *Deliverable 5.1.3*.

4 LinkedGeoData

We have attached our paper about LinkedGeoData, which is a project about converting OpenStreetMap data to RDF and making this data freely publicly accessible. In the course of this project, the widgets developed for this deliverable have been deployed for browsing the extracted data. The paper is currently under review at the Semantic Web Journal[6].

[6] `http://www.semantic-web-journal.net/content/linkedgeodata-core-web-spatial-open-data`

Undefined 0 (0) 1
IOS Press

LinkedGeoData: A Core for a Web of Spatial Open Data

Claus Stadler [a,*], Jens Lehmann [a], Konrad Höffner [a], Sören Auer [a]
[a] *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. The Semantic Web eases data and information integration tasks by providing an infrastructure based on RDF and ontologies. 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. This transformation will simplify information integration and aggregation tasks that require comprehensive background knowledge related to spatial features such as ways, structures, and landscapes. 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 an infrastructure based on 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* [14], 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[1].

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.

The goal of our *LinkedGeoData* (LGD) project is to lift OSM's data into the Semantic Web infrastructure. We believe that this will simplify real-life information integration and aggregation tasks that require comprehensive background knowledge related to spatial features. Such tasks might include, for example, to locally depict 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.

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

*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).

[1] See, for example, the listing at `http://ckan.net/group/lodcloud` and an overview at `http://lod-cloud.net`.

[2] `http://openstreetmap.org`

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: We now support nicer URIs (camel case), typed literals, language tags, and a simplified mapping of the OSM data to classes and properties. Together this accounts for an *improved data quality*.
- *Better support for ways:* Ways are OpenStreetMap entities used for modelling things such as streets but also areas (see Section 2). The geometry of a way (a line or a polygon) is now stored in a literal of the corresponding RDF resource, which makes it easy to e.g. display such a resource on a map. Furthermore, all nodes referenced by a 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 *republication 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. Subsequent sections explain, how the OSM data is transformed into the RDF data model (Section 4), how the data can be accessed (Section 5), and how we interlinked it with other knowlegde bases (Section 6). The live synchronization is explaind in Section 7. We present statistics about LinkedGeoData in Section 8. In Section 9, we showcase a faceted geo-data browser and editor as well as some 3rd party applications being built around LinkedGeoData. 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.[3] It is comprised by more than 1000 ways, which represent certain segments of the German border; the German border with Luxembourg e.g. is composed 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 metadata entries

[3] `http://www.openstreetmap.org/browse/relation/51477` can be used to browse this relation.

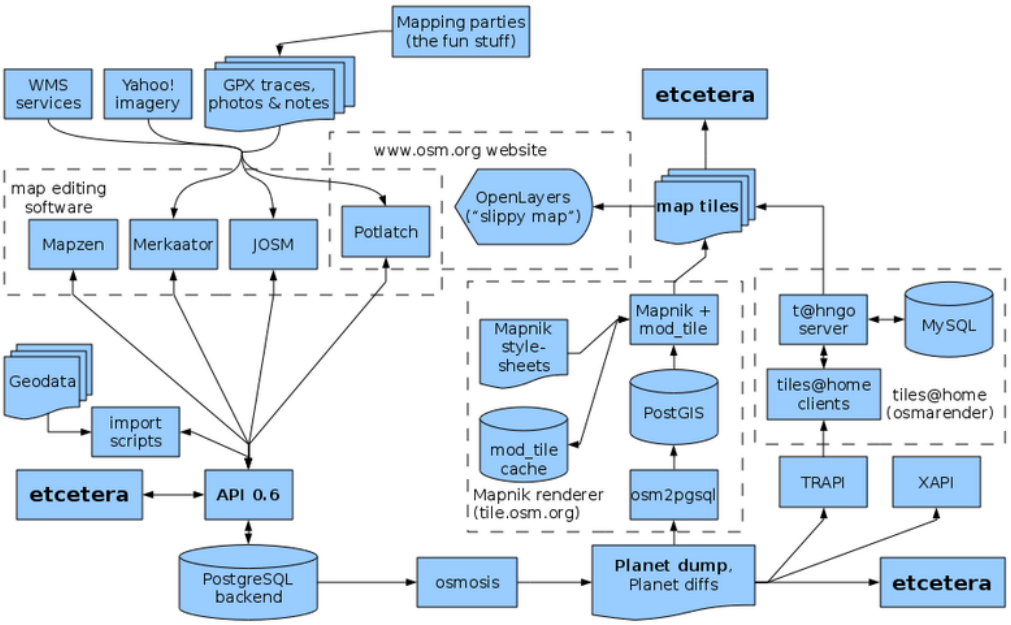

Fig. 1. Overview of OpenStreetMap's architecture.
Source:`http://wiki.openstreetmap.org/w/images/1/15/OSM_Components.png` as of 2011 Apr 27th

(such as the timestamp of the last edit and the last editor) attached.

To manage those datastructures, an infrastructure evolved encompassing multiple map editing tools, tile renderers, and data sources, as shown in Figure 1.

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.

OpenStreetMap's community has build different authoring interfaces. 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-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.

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[4] 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[5].

Currently, OSM is in the process of switching from the Creative Commons CC-BY-SA license to the Open Database License[6]. The term *Volunteered Geographic Information* (VGI) was coined [9] for the harnessing of tools to create, assemble, and disseminate geographic

[4] `http://tagwatch.stoecker.eu/`
[5] `http://wiki.openstreetmap.org/wiki/Map_Features`
[6] `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` at the specified months.)

data provided voluntarily by individuals – with OSM being a driving force behind VGI.

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[7]. Some of the data was imported form public domain datasources such as TIGER[8] for US, AND Automotive Navigation Data[9] for The Netherlands, and GeoBase data from the Canadian government[10].

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*[11], which supports setting up such a 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>`,

[7] `http://www.openstreetmap.org/stats/data_stats.html`, retrieved 2011 May 2nd
[8] `http://www.census.gov/geo/www/tiger`
[9] `http://www.and.com/`
[10] `http://www.geobase.ca`
[11] `http://wiki.openstreetmap.org/wiki/Osmosis`

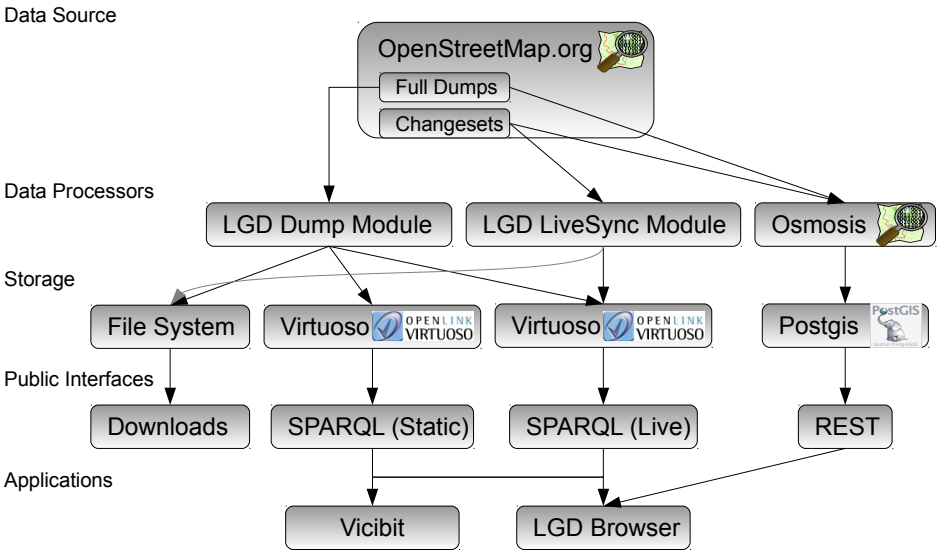

Fig. 2. Overview of LinkedGeoData's architecture.

respectively.[12] The resource corresponding to a way's list of nodes is `lgd:way<id>/nodes`.

These URIs are non-information resources, i.e. they represent real-word 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, where 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.

[12] See Appendix A for prefix declarations.

– *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, hindi)`, is mapped to the predicate `lgdo:religion` and object `lgdo:hindi`. 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://linkedgeodata.org/ontology/religion
  </property>
  <tagPattern>
    <key>religion</key>
  </tagPattern>
  <describesOSMEntity>false</describesOSMEntity>
  <objectAsPrefix>true</objectAsPrefix>
  <object>
    http://linkedgeodata.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*[13] vocabulary.

4.1. The LinkedGeoData Ontology

Based on the OpenStreetMap tags, we derived a lightweight OWL ontology[14]. A depiction of an excerpt in shown in Figure 3.

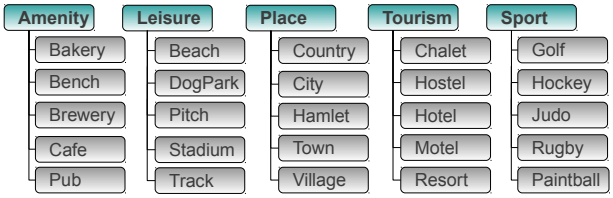

Fig. 3. An excerpt of the LinkedGeoData ontology. More classes and subclasses exist in the actual version.

The process of creating it is explained as follows: 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://linkedgeodata.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://linkedgeodata.org/ontology/Restaurant
  </object>
</SimpleResourceTagMapper>
```

[13] `http://www.heppnetz.de/projects/goodrelations/`

[14] The complete ontology is available at `http://linkedgeodata.org/ontology/`

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."[15] 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%[16]. They 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.

These labels could serve as a basis for answering queries posed in different languages: For a query such as "Bakeries in Munich" and its German equivalent "Bäckereien in München", the search words could be mapped to corresponding classes and instances from the LinkedGeoData knowledge base. A system already capable of processing such types of queries is described in [20].

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.[17] 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.

5. Data Access

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

- dataset downloads (HTML download table[18] and actual files[19]), including live sync changesets relative to the latest release[20] (explained in Section 7)
- a static SPARQL endpoint[21]
- a live SPARQL endpoint[22]
- Linked Data via 303 content negotation (RDF/XML, Turtle, N-Triples, HTML formats supported)
- a REST API

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

[15] http://translatewiki.net

[16] http://translatewiki.net/wiki/Translating:OpenStreetMap/stats/trunk retrieved 5th May 2011.

[17] http://www.sjjb.co.uk/mapicons/ retrieved 6th April 2011

[18] http://linkedgeodata.org/Datasets

[19] http://downloads.linkedgeodata.org

[20] http://downloads.linkedgeodata.org/releases/latest/changesets/

[21] http://linkedgeodata.org/sparql

[22] http://live.linkedgeodata.org/sparql

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[23] enterprise edition database system automatically indexes such points in an R-tree.

Listing 3: Example dataset in Turtle syntax.

```
lgd:way4009992
  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)"^^virtrdf:Geometry
```

5.2. 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 changesets. On the other hand, the data for the ontology and interlinking is drawn from the SPARQL endpoints, as depicted in Figure 4.

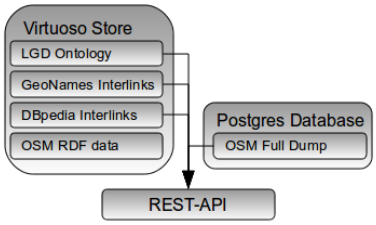

Fig. 4. Data Sources of the REST API.

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 (`rdfs: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*, *DBpedia*, *GeoNames* and the *Food and Agriculture Organization of the United Nations (FOA)* is described. In all cases, we first manually aligned the classes of these knowledge bases with classes from LinkedGeoData on an best effort basis. The interlinking is then 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

[23] `http://virtuoso.openlinksw.com`

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

Table 2
Excerpt of the methods supported by the LGD REST API.

all instances of `lgdo:City`, `lgdo:Town`, `lgdo:Village` and `lgdo:Hamlet` on one side and `dbo:Settlement` on the other. The interlinking is performed using the tools SILK [24] and LIMES [17], which use the static SPARQL endpoint as the backend. As Virtuoso's index support for geometries is limited to points, we needed to constrain the interlinking to LinkedGeoData nodes. An overview of triple stores supporting complex geometries is given in Section 10.3.

It should be noted, 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* [24]. 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* [25], between the labels of a and b. If there are multiple labels, the 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 classes and is explained by the choice of reference points, which can differ in each of the considered knowledge bases.
- $g_c(a,b) = \begin{cases} 0 & d > c \\ 1/(1+e^{-12d'+6}) & otherwise \end{cases}$ 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	instances	LGD equivalent	c in km	nodes	links	precision
Airport	9520	Aerodrome	2.5	43734	8404	1.0
Settlement	239630	several[1]	0.1	620387	88377	1.0
Country	2505[2]	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

[1] City ∪ Suburb ∪ Town ∪ Village
[2] 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 `T` (*Peak*). Linking GeoNames with LinkedGeoData makes these 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 Table 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`, `listName` and `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 tech-

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[1]	818 893	34 907	0.984
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	2 922 222			3 148 630	606 549	0.989

[1] because of many incorrect links in the original matching, only links of settlements with a distance of at most 5 km were finally used.

niques, 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‘, ‘Beerenklippe‘)

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* [16] 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. Initial experiments with LIMES gave comparable results in significantly less time. We expect, that with such a new technology, will be able 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 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 users of the LinkedGeoData service to synchronize their own RDF store with it.

An example of an application of LinkedGeoData live is the service *MovieGoer*[24], which scrapes websites about cinemas in Munich and Innsbruck for their program and stores the result as RDF. This data was then interlinked with LinkedGeoData, as the SPARQL endpoints provide a simple means of retrieving the addresses and names for these cinemas. The locations that were found out to be missing during the interlinking were added to OpenStreetMap, which made them also available at the live LinkedGeoData endpoint. As a result, a benefit for all involved services was created.

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.

[24] `http://lokino.sti2.at/`

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 children of the *create*, *modify* and *delete* elements are elements describing the affected OSM entities. These descriptions are interpreted in context of their parent element as follows:

- Create: The state of the newly created entity.
- Modify: The new state of the entity after its modification.
- Delete: The state of the entity prior to its 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 [23] 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. Relevance in determined in regard to a filter configuration consisting of black- and whitelisted tags (See 7.5). The filtering 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* the nodes, as each of them may at some point in time become connected to a 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 *lgdo: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, 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 the nodes, we can already omit approximately 20 million 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[25]. As a final filtering step, we reject ways with more than 20 nodes in order to prevent filling the store mainly with way-node relations rather than information based on tags, as each way-node relation results in two triples: one for relating the way to its node, and one for each node position. Therefore, a single way with a relevant tag and 20 nodes already results in more than 40 triples.

[25] For the filter configuration of the release at the time of writing, see the files LiveEntityFilter.txt, LiveRelevanceFilter.txt, and LiveTagFilter.txt at `http://downloads.linkedgeodata.org/releases/110406/`

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.

Let: $N \leftarrow \emptyset, O \leftarrow \emptyset$
for all nodes n in C **do**
  if created(n) **then**
    Insert (n.id, n.position) into node store
    if relevant(n) **then**
      $N \leftarrow N \cup generateRDF\_Node(n)$
    end if
  else if modified(n) **then**
    Update (n.id, n.position) in node store
    $O \leftarrow O \cup fetchRDF\_Node(n)$
    if relevant(n) **then**
      $N \leftarrow N \cup generateRDF\_Node(n)$
    end if
    for all ways w where n is a member **do**
      $st_o \leftarrow fetchShapeTriple(w)$
      $O \leftarrow O \cup st_o$
      st_n = createNewShapeTripleWithPositionReplaced(st_o, n)
      $N \leftarrow N \cup st_n$
    end for
  else if deleted(n) **then**
    Remove entry for (n.id) from the node store
    $O \leftarrow O \cup fetchRDF\_Node(n)$
  end if
end for
for all ways w in C **do**
  if created(w) **then**
    if relevant(w) **then**
      $m \leftarrow fetchNodePositionMap(w.nodeRefs)$
      $N \leftarrow N \cup generateRDF\_Way(w, m)$
    end if
  else if modified(w) **then**
    $w_o \leftarrow fetchRDF_Way(n)$
    $O \leftarrow O \cup w_o$
    if relevant(w_o) and not relevant(w) **then**
      RemoveIrrelevantNodes(w_o.nodeRefs)
    end if
    if relevant(w) **then**
      $m \leftarrow fetchNodePositionMap(w.nodeRefs)$
      $N \leftarrow N \cup generateRDF\_Way(w, m)$
    end if
  else if deleted(w) **then**
    $O \leftarrow O \cup fetchRDF\_Way(n)$
    RemoveIrrelevantNodes(w.nodeList)
  end if
end for
procedure REMOVEIRRELEVANTNODES($nodes$) ▷
  for all nodes n in $nodes$ **do**
    $d \leftarrow fetchRDF\_Node(n)$
    if not relevant(d) **then**
      $O \leftarrow O \cup d$
    end if
  end for
end procedure
$Additions \leftarrow N \setminus O$
$Removals \leftarrow O \setminus N$

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[26]. The top ten machines were responsible for 73% of those queries. More than 1.000 queries were issued by 19 of them. Figure 5 shows the number of queries per day. The diagram indicates that the LGD service is being

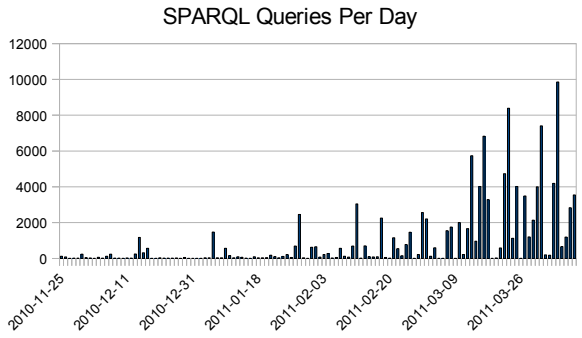

Fig. 5. Usage of the SPARQL endpoints.

actively used, and that the usage rate is increasing.

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 changsets corresponding to roughly three weeks.

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

[26] Not counting the queries from our own network.

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 6)[27]. It allows to browse the world by using a "slippy map"[28]. 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, becomes slower the greater the size of the search region. This is due to the fact that the database has to aggregate the facets from all the entities that fall into the search region.

To resolve this problem we precomputed tile-based statistics at various zoom levels. If zoomed in close, the exact counts will be measured, however if zoomed out the counts will be approximated by aggregating the the statistics of the visible tiles.

Furthermore, there are several new smaller LGD browser features compared to its previous version de-

[27] Available online at: `http://browser.linkedgeodata.org`

[28] `http://wiki.openstreetmap.org/wiki/Slippy_Map`

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 data from April 6th with live data from April 30th 2011.

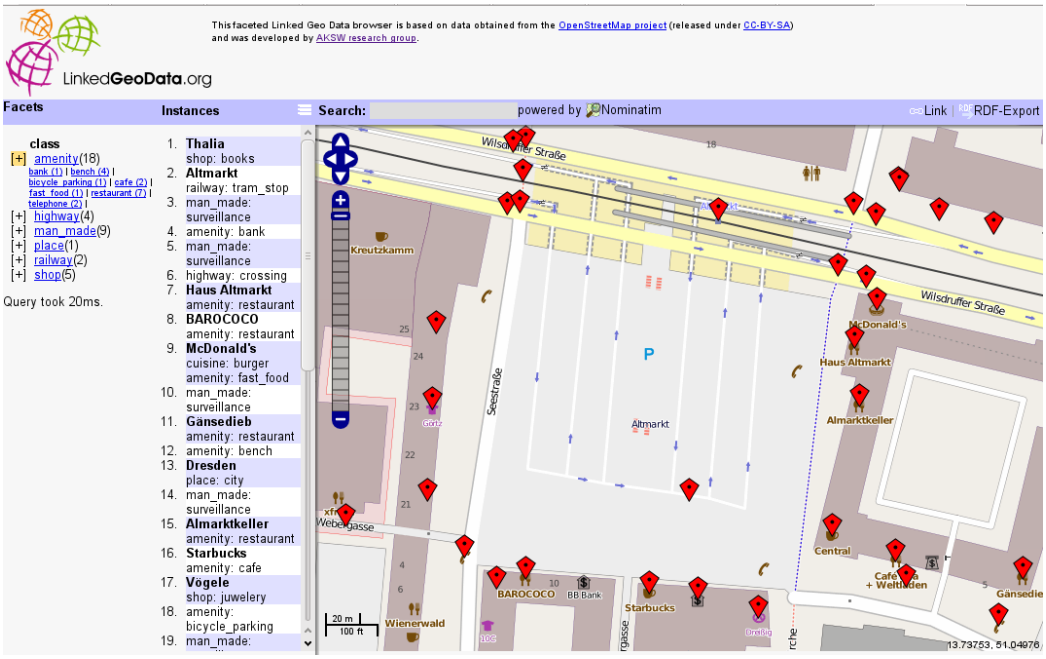

Fig. 6. LinkedGeoData Browser.

scribed in [1]. For instance, an RDF export of the current map selection including its facets can now be performed. This allows the easy extraction of 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 [29] [4] is an application developed by the Institute for Web Science and Technologies at the University of Koblenz, which uses LinkedGeoData. STEVIE allows one to create and edit points of interests (POIs) (see Figure 7) 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[30] is a website, which enables one 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 benefits for the application[31]: “First of all, LinkedGeoData 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 8 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 in-

[29] http://tiny.cc/stevie10

[30] http://beaware.at/

[31] http://alexidsa-en.blogspot.com/2010/06/rdf-vs-nonrdf-for-geodata-at-beaware.html

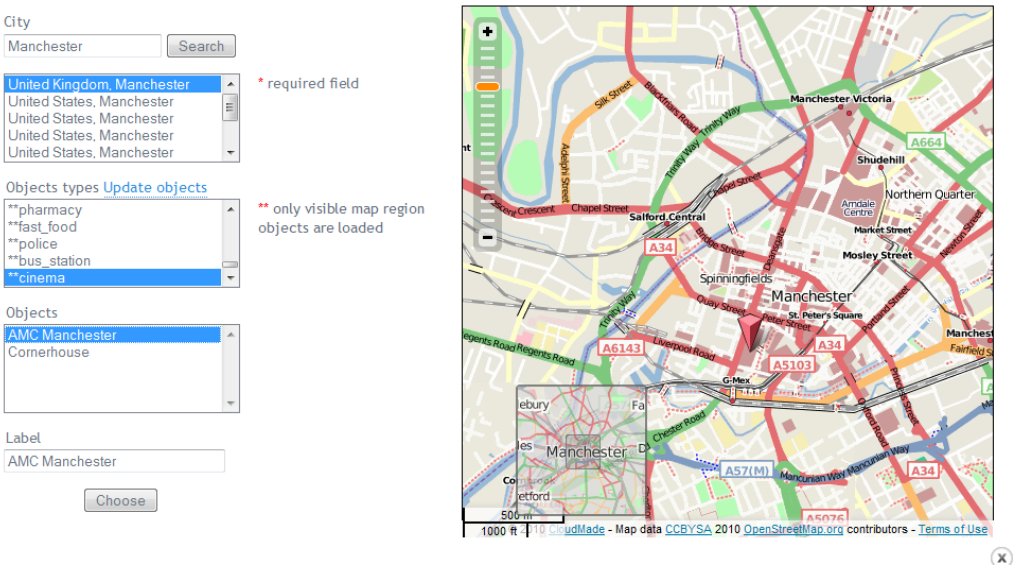

Fig. 8. Marking the location of an event in BeAware.

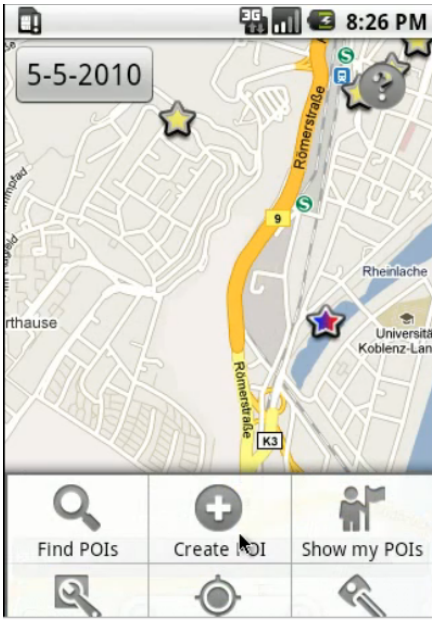

Fig. 7. Creation of a point of interest in STEVIE. The application includes a temporal dimension and highlights POIs where events take place in the selected time span.

formation about the location from an interlinked data source is available and can be presented to the user.

9.4. Layar

Layar[32] 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 show in a detail view.

9.5. Vicibit

Vicibit[33] ("exhibit your vicinity") is a tool working on top of LinkedGeoData Live, which allows to create customised views on LinkedGeoData. It enables users to pick classes from 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[34] on data in the LinkedGeoData Live SPARQL endpoint. A typical use case is that a

[32] `http://www.layar.com`
[33] `http://vicibit.linkedgeodata.org`
[34] `http://www.simile-widgets.org/exhibit/`

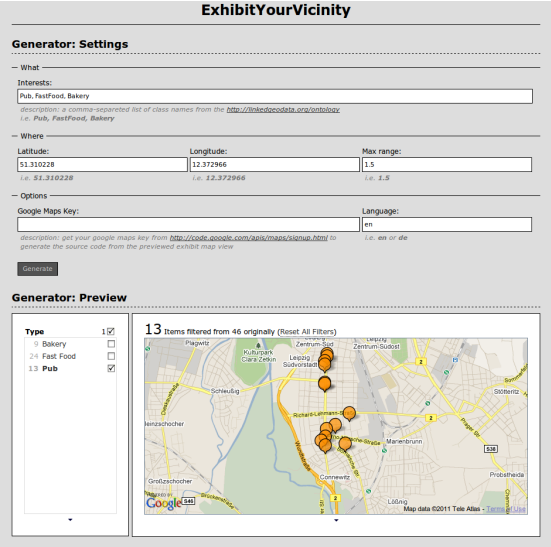

Fig. 9. Vicibit is a tool to generate custom views on LinkedGeoData via Exhibit. The example shows a faceted view on nearby pubs, bakeries and shops. The code generated by Vicibit can easy be pasted into blogs, forums and web pages.

webpage or blog entry describing a particular event can be enriched with a map of nearby pubs and other shops (see Figure 9).

10. Related Work

We split the presentation related work into five 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 a crucial task we faced in LinkedGeoData. We then give an overview of triple stores supporting geographic data more complex than point data. Finally, we review some related work done by the GIS community, 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.

Ordnance Survey[35] is the national mapping agency in Great Britain. Over the past years, they released some of their products as Linked Data[36]. 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 [10] 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 also LinkedGeoData, are world-wide and community-driven approaches.

GeoNames is a comprehensive global spatial database containing several million features, such as cites, forests, and peaks. This data has been converted to RDF[37] and is served 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 developed 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 and tailor for new, previously unforeseen usage scenarios. 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 includes entities such as traffic lights or trash bins.

The *United Nations FAO (Food and Agriculture Organisation) Geopolitical Data* [6] 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 includes 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 [3]. It focuses on hydrography features and integrates several existing data sources.

[35] http://www.ordnancesurvey.co.uk

[36] http://data.ordnancesurvey.co.uk

[37] http://www.geonames.org/ontology/

NUTS (Nomenclature of Territorial Units for Statistics) provides a hierarchical system for describing the economic territory of the European Union [38]. The NUTS hierarchy is established by EuroStat. EU NUTS data has been converted to RDF [39]. 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 many tools available. In particular, there is the W3C RDB2RDF working group, which aims to standardize a database to RDF mapping language [7]. Instead of providing an in-depth overview, we refer to recent surveys [21,22] and overviews[40] on this topic. There are various tools available implementing the surveyed approaches such as D2R, Triplify, DartGrid, DataMaster, MapOnto, METAmorphoses, ODEMapster, RDBToOnto, RDOTE, 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. Triple Stores supporting Complex Geometries

In the relational database realm, support for complex geometric data (points, lines, and polygons) is already well established. Examples are MySQL[41], PostgreSQL[42], and Oracle[43]. Meanwhile, in the Semantic Web, the support for geometric data has also increased significantly. Currently, we are aware of four triple stores supporting this kind of data: *OWLIM Standard Edition*[44] (OWLIM-SE, previously known as BigOWLIM), *Open Sahara*[45], *Parliament*[46], and *AllegroGraph*[47].

So far there has not been a standard query language for geospatial data in the Semantic Web. GeoSPARQL is a proposed extension to SPARQL 1.0 [19] with the aim to provide this functionality. It is currently at the stage of becoming a standard by the Open Geospatial consortium[48]. With these tools, tasks that require complex operations on geometries become possible in the Semantic Web.

We chose Virtuoso as the backend for LinkedGeoData because of its good performance [2] and our assumption that point geometries would be sufficient for most of our interlinking tasks. Although this turned out to be true, in the future we want to extend these tasks to complex geometries and therefore also evaluate the other stores.

10.4. Gazetteer Mapping

Our interlinking approach is very similar to the matching of gazetteers in traditional Geographic Information Systems (GIS): [11] discusses the alignment of two major gazetteers. For this purpose, place names, types and footprints (geographical entities such as point, lines and polygons) were identified as the most fundamental features of gazetteers. These entities closely resemble the concepts of classes, labels and geometries, on which we based our interlinking. Prior work about the derivation of an OWL ontology suitable for similarity searches from gazetteers is given in [12]. A comprehensive discussion about similarity search paradigms for Description Logics is lead in [13].

10.5. Interlinking and Ontology Mapping

There have been several decades of research starting with the integration of different database schemata. Tools like COMA [8] provide rich support for various matching operations between databases as well as between RDF knowledge bases. [5] 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 exten-

[38] http://epp.eurostat.ec.europa.eu/portal/page/portal/nuts_nomenclature/introduction
[39] http://rdfdata.eionet.europa.eu/ramon/nuts2008/
[40] http://esw.w3.org/topic/Rdb2RdfXG/StateOfTheArt
[41] http://www.mysql.com
[42] http://www.postgresql.org/
[43] http://www.oracle.com
[44] http://www.ontotext.com/owlim/geo-spatial
[45] http://opensahara.com
[46] http://parliament.semwebcentral.org/
[47] http://www.franz.com/agraph/allegrograph/
[48] http://www.opengeospatial.org/

sive experiment, carried out within the context of two gazetteers, GeoNames and the ADL gazetteer, to illustrate the idea. [15] describes an approach integrating spatial 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 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. These enhancements may further contribute to semantic-spatial search engines, such as [13,20], and enable a variety of new Linked Data applications such as geo-data syndication (publishing information about geographical entities via feeds). Another example is personalized and context-sensitive spatial Linked Data update propagation and consumption, which might be realized with systems such as sparqlPuSH [18].

The dynamics of the OpenStreetMap project will ensure a steady growth of the LinkedGeoData 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.

Currently, our transformation approach imposes the following *limitations* on the use of LinkedGeoData:

- The current *ontology* is mainly automatically derived from OpenStreetMap tags, with mostly just minor manual edits. However, it could benefit from axiomatizations, such that, for example, any *PlaceOfWorship* with religion *christian* is a *Church*. The extent to which the addition of disjointness axioms makes sense needs yet to be investigated. For instance, currently instances corresponding to hotels that also offer a restaurant are currently tagged with both types. However, an alternative solution would be to model such instance as a *Hotel*, that offers a feature that is a *Restaurant*. For these kinds of design decisions, we envision a solution similar to the *DBpedia Mapping Wiki*[49], that enables the community to contribute to the axiomatization of the ontology.
- Our *filtering* (see Section 7.5) currently discards a significant amount of data from OSM. Hence, there are use cases that are possible with OSM data, but not with LinkedGeoData yet. For example, since we filter out ways with more than 20 nodes, routing[50] is currently not possible based on the LinkedGeoData SPARQL endpoints.
- Because we do not support OpenStreetMap relations yet, information about compound entities is also not yet available in LinkedGeoData. Examples of such entities are: multipolygons (collections of polygons, where each member may act either as solid or as a hole), or designation signs. Further examples include large boundaries, waterways, and routes, that are modelled with way segments.

As for the latter two limitations, we are currently investigating whether and how these limitations can be overcome by directly rewriting SPARQL queries to SQL queries over the relational schema of OpenStreetMap. Although substantial progress was made in RDB-RDF mapping during the last years, and implementations are now more robust, scalability and the lack of support for geometry datatypes is still an issue preventing a direct deployment of these technologies for LinkedGeoData.

Another stream of future work is the better support for geometries according to the current NeoGeoVocabulary development[51], which we are supporting. A semantic misrepresentation currently found in LinkedGeoData, for example, is the missing separation of geometries (such as points and polygons) and features

[49] http://mappings.dbpedia.org
[50] http://wiki.openstreetmap.org/wiki/Routing
[51] http://geovocab.org/doc/neogeo.html

(such as hotels and pubs), which we plan to resolve in the future.

Finally, we identified further candidates that seem worthwhile for interlinking:

- The *CIA World Factbook*[52] contains detailed information on the country level, such as their conventional names, their birthrate, and their gross domestic product. An RDF version is hosted by the Free University of Berlin[53].
- The site `climb.dataincubator.org` hosts a collection of data of about 1400 climbing locations with latitude/longitude information. The resources were collected from various climbing web sites and converted to RDF.
- *Last.fm*[54] has information about music artists, as well as events, such as performances and festivals. Many event locations are geo-tagged, making them suitable candidates for interlinking. There exist at least two wrappers for the last.fm API that return RDF[55].

Acknowledgments

We 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 thank the members of the LinkedGeoData community and 3rd party application developers for their valuable feedback and contributions to the project. In particular, we would like to mention 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] C. Bizer and A. Schultz. The berlin sparql benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24, 2009.

[3] 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.

[4] 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.

[5] 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.

[6] C. Caracciolo, M. I. Sucasas, and J. Keizer. Towards interoperability of geopolitical information within FAO. *Computing and Informatics*, 27(1):119–129, 2008.

[7] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. World Wide Web Consortium, Working Draft WD-r2rml-20110324, Mar. 2011.

[8] H. H. Do and E. Rahm. COMA - A system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. Morgan Kaufmann, 2002.

[9] M. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, Aug. 2007.

[10] M. Haklay. How good is volunteered geographical information? A comparative study of openstreetmap and ordnance survey datasets. July 2010.

[11] L. L. Hill. Core elements of digital gazetteers: Placenames, categories, and footprints. In J. L. Borbinha and T. Baker, editors, *ECDL*, volume 1923 of *Lecture Notes in Computer Science*, pages 280–290. Springer, 2000.

[12] K. Janowicz and C. Keßler. The role of ontology in improving gazetteer interaction. *International Journal of Geographical Information Science*, 22(10):1129–1157, 2008.

[13] K. Janowicz, M. Wilkes, and M. Lutz. Similarity-based information retrieval and its role within spatial data infrastructures. In T. J. Cova, H. J. Miller, K. Beard, A. U. Frank, and M. F. Goodchild, editors, *GIScience*, volume 5266 of *Lecture Notes in Computer Science*, pages 151–167. Springer, 2008.

[14] 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.

[15] H. Manguinhas, B. Martins, and J. L. Borbinha. A geo-temporal web gazetteer integrating data from multiple sources. In *ICDIM*, pages 146–153. IEEE, 2008.

[16] D. G. McVitie and L. B. Wilson. The stable marriage problem. *Commun. ACM*, 14(7):486–490, 1971.

[17] 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.

[18] A. Passant and P. Mendes. sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. volume 699 of *CEUR Workshop Proceedings ISSN 1613-0073*, February 2010.

[19] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[20] R. S. Purves, P. Clough, C. B. Jones, A. Arampatzis, B. Bucher, D. Finch, G. Fu, H. Joho, A. K. Syed, S. Vaid, and B. Yang. The

[52] `https://www.cia.gov/library/publications/the-world-factbook/`

[53] `http://www4.wiwiss.fu-berlin.de/factbook/`

[54] `http://last.fm`

[55] `dbtune.org/last-fm/` and `lastfm.rdfize.com/`

design and implementation of spirit: a spatially aware search engine for information retrieval on the internet. *International Journal of Geographical Information Science*, 21(7):717–745, 2007.

[21] 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.

[22] D.-E. Spanos, P. Stavrou, and N. Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web Journal*. under review.

[23] 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.

[24] 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.

[25] W. Winkler. The state of record linkage and current research problems. Technical report, U.S. Bureau of the Census, 1999.

Appendix

A. Prefixes Used

The following prefixes are used in the paper:

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