Highlights

Linking geospatial data with Geo-L - analysis and experiments of big data readiness of common technologies
Christian Zinke-Wehlmann, Amit Kirschenbaum

- System for geospatial linking that improves the state-of-the-art
- Enhance scalability, efficiency, robustness and interoperability for Linked Data linking tasks
- Saves time and space for retrieval of resources
- Allows simple datasets definition by SPARQL queries
Linking geospatial data with Geo-L - analysis and experiments of big data readiness of common technologies

Christian Zinke-Wehlmann*, Amit Kirschenbaum*

Institut für Angewandte Informatik e.V. an der Universität Leipzig (InfAI), Goerdelerring 9 04109 Leipzig, Germany

ARTICLE INFO

Keywords:
geospatial linking
topological relations
semantic web

ABSTRACT

Geospatial Linked Data is an emerging domain, with growing interest in research and industry. The growing sources of open-source geospatial Linked Data need to be interlinked and easily integrated with private and industrial linked data clouds, e.g., in the agricultural domain. The presented paper analyzes common link discovery tools for the feasibility of big linked data processing. Further, it introduces Geo-L, a system for discovery of RDF spatial links based on topological relations. Experiments show that the proposed system improves state-of-the-art spatial linking processes in terms of mapping-time and -accuracy, as well as concerning resources retrieval efficiency and robustness.

1. Introduction

The Web of Data, or Semantic Web, is a continuously growing global data space (https://www.w3.org/2013/data/). Semantic Web standards, such as RDF [1, 2], OWL [3, 4], and SPARQL [5] were developed to express and exchange semantic information on the Web, which tackle the challenge of interoperability [6]. In the geospatial context, most prominent is the GeoSPARQL initiative, which offers a necessary vocabulary to develop geo-related data on the Semantic Web [7]. In recent years, geospatial linked data gained increasing attention [8], also due to advances in the Earth Observation domain [9]. Thus, numerous resources of linked geospatial data have been developed, e.g., LinkedGeoData [10], Smart Point Of Interest [11], Spanish Cases [12], and Ireland’s national geospatial data [13]; and the domain is constantly growing within the Linked Data Cloud. Notably, the domain of geospatial data contains complex datasets, as NUTS [14], which describe territories using polygons that may be more than 1700 points long.

According to the Linked Data principles, published data should be interlinked with other datasets on the Web [15]. In general, linking (and fusing) of geospatial linked data sources enable large-scale inferences and data integration [16]. Nevertheless, clear links are often not part of the dataset and should be discovered automatically, even in a distributed cloud environment and huge datasets. These linking activities are one pillar to foster the development of innovative software solutions. In particular, the linking of geospatial data is a challenging task, since the links express relations which depend on complex geometric computations. Based on a comparison of existing approaches, services and tools, the goal of the paper is to give an overview and introduce a novel specialized tool for link discovery on geospatial link data with the following requirements:

1. Scalability and efficiency: As mentioned before, the Linked Data cloud is continually growing employing new sources and data sets. The service should be able to handle big data sets. The idea is to provide a service for different Linked Data environments (open or closed). Therefore, the time efforts have to be reduced on a significant minimum. The vision is to discover even extensive data sets in near real-time.

2. Robustness: The service must retain functionality under unforeseen conditions, as missing or corrupted data. This is especially true for crowd-sourced or automatically generated data sets, which are likely to include errors as the size of data grows.

3. Interoperability and flexibility: The service has to be handled as easy and transparent as possible. The (SPARQL affine) user should be able to easily formulate queries to retrieve source and target datasets, as well as the linking condition. This includes the ability to handle data whose representation is not compatible for computing of, e.g., topological relations. Further, the service has to handle on-the-fly requests by a RESTful input processing. It has to operate easily as a standalone system or as a module integrated into other applications.

The paper is structured as follows: Firstly, we introduce the theoretical background of geospatial link discovery. Secondly, we describe previous work related to geospatial link discovery. Thirdly, we present our solution. Fourthly, we evaluate our solution and compare it to the state-of-the-art system.

2. Background

Linked (Open) Data refer to an area which focuses on the publishing of RDF (Resource Description Framework) on the Web of Data. However, the Linked Data approach is strongly linked to the Tim Berners-Lee Linked Data Principles [15]. The basic idea of link discovery is to find data items within the target dataset which are logically connected to the source dataset. More formally this means: Given $S$ and $T$, sets of RDF resources, called source and target resources respectively, and a relation $R$, the aim of link discovery methods is to find a mapping $M = \{(s,t) \in S \times T : R(s,t)\}$. Naive computation of $M$ requires quadratic time complexity to test for every $s \in S$ and $t \in T$ whether $R$ holds, which is unfeasible for large data sets.

In geospatial context, $S$ and $T$ are geometries in a two dimensional space, and the links may be based on proximity.
or on topological relations. In the latter case, relations are expressed by the Dimensionally Extended nine-Intersection Model (DE+9IM) [17, 18], which was accepted as an ISO standard [19]. DE+9IM classifies binary spatial relationships two between geometries, \(a\) and \(b\), which may be points, lines, or polygons, based on intersection of interiors (I), boundaries (B) exteriors (E) of \(a\) with those of \(b\).

A combination of these six geometric parts/features defines topological relations, which are described in a \(3 \times 3\) matrix as follows:

\[
\text{DE+9IM}(a, b) = \begin{bmatrix}
\dim(I(a) \cap I(b)) & \dim(I(a) \cap B(b)) & \dim(I(a) \cap E(b)) \\
\dim(B(a) \cap I(b)) & \dim(B(a) \cap B(b)) & \dim(B(a) \cap E(b)) \\
\dim(E(a) \cap I(b)) & \dim(E(a) \cap B(b)) & \dim(E(a) \cap E(b))
\end{bmatrix}
\]

The intersection \(S\) of some feature of \(a\) with a feature of the \(b\), may be either empty or in itself a geometric object, namely: a point, a line, or a polygon. \(\dim(S)\) returns the dimension of the geometry \(S\); if \(S\) consists of multiple geometries then \(\dim(S)\) is the maximal dimension of intersection if it is a of multiple parts.

\[
\dim(S) = \begin{cases}
-1 & \text{if } S = \emptyset \\
0 & \text{if } S \text{ contains at least one point, but no lines or polygons} \\
1 & \text{if } S \text{ contains at least one line, but no polygons} \\
2 & \text{if } S \text{ contains at least one polygon}
\end{cases}
\]

In addition to the dimensions values the matrix may contain the values \(T\) (\(\dim(S) \geq 0\)), \(F\) (\(\dim(S) = -1\)), and \(\ast\) ("don’t-care" value, which means that the value in this matrix cell has no influence on the outcome of a function applied to this matrix). The model defines topological predicates to describe the spatial relations between the two geometries in a compact and human-interpretable manner, which are defined by pattern matrices: \text{equals}, \text{disjoint}, \text{intersects}, \text{touches}, \text{crosses}, \text{overlaps}, \text{within}, \text{and contains}. For example, the pattern matrix for the relation \text{within} is defined by the following pattern matrix \(^1\)

\[
a.\text{within}(b) = \begin{bmatrix}
T & \ast & F \\
\ast & \ast & F \\
\ast & \ast & F
\end{bmatrix}
\]

formally described as \((I(a) \cap I(b) \neq \emptyset) \land \neg(I(a) \cap E(b) \neq \emptyset) \land \neg(B(a) \cap E(b) \neq \emptyset) \land \neg(E(a) \cap E(b) \neq \emptyset)\).

3. Related Work

Link discovery of topological relations among RDF data sets has received growing interest in recent years, and various methods for this problem have been proposed. These methods usually define the topological relations between two geometries based on their relations computed between their minimum bounding boxes. A minimum bounding box (MBB) is the rectangle of minimum area that encloses all coordinates of geometry and is a commonly used as an approximation to the geometry to reduce computational costs that involve this geometry [21].

Smeros and Koubarakis [22] use the \text{MultiBlocking} technique [23] to discover topological relations. This technique divides the earth surface into curved rectangles, and assigns each geometry to all blocks in which it intersects, based on the geometry’s MBB. Relations discovered within each block are then aggregated to construct the links. This method is embedded in the Silk framework [24].

RADON [25] divides the space into hyper-cubes and uses optimized sparse space tiling to index geometries. This is done by mapping each geometry to the set of hyper-cubes over which it’s minimum bounding box (MBB) spans. The method first indexes geometries \(s \in S\) and then only index geometries \(t \in T\) that may potentially reside in hyper-cubes already contained in the index. To minimize the size of the index, the method implements a swapping strategy, that is, prior to the indexing phase it calculates an estimated total hypervolume (\(eth\)) for each of the datasets \(S\) and \(T\). If \(eth(T) < eth(S)\) then it swaps the two datasets and computes the reverse relation of the requested relation \(R\). The link generation itself is done using a method that reduces computations on a subset of DE+9IM relations. \text{RADON} is implemented as part of the LIMES framework [26].

Faria et al. [27] adapt the AgreementMakerLight (AML) [28], a framework for automated ontology matching, to tackle the task of topological relations. This is done by utilizing \text{ESRI Geometry API}, which uses quadtree as means to index geometries and detect topological relationship among them.

These methods, as well as OntoIdea [29], were evaluated on several sets of geometries: Achichi et al. [30] apply them to discover topological relations between LineStrings, constructed of trajectories from the TomTom\(^4\) dataset. [31] apply these methods to find relations between LineStrings to LineStrings and between LineStrings to Polygons, from TomTom dataset and Spaten dataset [32] respectively. All datasets included at most 2000 instances. Both evaluations report that the methods mentioned above discover links correctly, that is, the \(F\)-score of most of them is 1.0 (apart from OntoIdea which \(F\)-score lies between 0.91 and 0.99, and did not take part in the tasks for link discovery between linestrings and polygons).

Strabon [33] is an open-source geospatial RDF store. It is based on the RDF4J (previously Sesame) RDF store and adds geospatial capabilities to it by implementing the OGC-standard GeoSPARQL, where as part of the implementation the stored geometries in Strabon are indexed with an R-Tree (GiST). Implementing GeoSPARQL means that Strabon includes topological functions; thus, queries that use these functions can be viewed as a means to discover topological relations. Sherif et al. [25] compares the performance of Silk, Strabon, and \text{RADON} where they are applied to discover links between different subsets of NUTS and \text{CORINE Land Cover}\(^5\) datasets, which map land and land-usage re-

\(^1\)see also Strobl [20]

\(^2\)https://github.com/dice-group/LIMES/

\(^3\)https://github.com/Esri/geometry-api-java/

\(^4\)https://www.tomtom.com

\(^5\)https://www.eea.europa.eu/publications/CORINE-landcover
spectively. The biggest dataset used in their experiments is of size 2, 209, 538.

The evaluations compare the running times of these methods with different dataset sizes. It has already been acknowledged that a significant portion big data is geospatial data [34, 35], thus our interest lies in the performance of these systems on large datasets. Table 1 summarizes how well the methods described above perform, regarding the criteria for useful geospatial link discovery systems, discussed in Section 1, as reported in the literature [25, 30, 31].

As can be observed in Table 1, the LIMES system, that implements RADON, was the one who completed all the link discovery task for all topological relations and performed best for most of them. We, therefore, take LIMES as our main reference point. Nevertheless, LIMES as it is, is not sufficiently flexible to accommodate geospatial data of different formats, and requires external pre-processing of input. Additionally, LIMES assumes an error-free download and curated data-sets, which is not always the case in reality. This motivates us to incorporate advantages of existing techniques in a single solution and test what existing technologies might be used for an efficient, flexible, robust and interoperable system for on-the-fly semantic linking of geospatial data.

4. Geo-L

As shown, LIMES performs the geospatial linking tasks very well, but lacks flexibility and accurate error handling. Starting as an experiment, we do not expand LIMES for this special case of linking big geospatial data (LIMES is the state-of-the-art tool for various kinds of link discovery tasks), but rather build a new geospatial linking tool to meet the requirements mentioned above. Our implementation focuses on allowing flexible input, short download and caching time as well as efficient link discovery.

4.1. Input

The input for a link discovery task provides the resources to be linked and the conditions upon which the links are generated, in a simple, yet flexible manner. In particular, our method offers a way to retrieve relevant properties from the endpoint via a SPARQL query; thus it natively supports manipulation of data, without any need for external pre-processing. This is useful, for example, when geometry values at the endpoint are not represented in a format that directly allows them in the database. Additional requests for data items from the same endpoint will be first served from the cache if the items are already indexed. This ensures a single local resource parallel to the endpoint, which serves arbitrarily many configurations, thus saves both time and storage. This differs from the behavior of LIMES, where data items may be downloaded multiple times, and duplicates of the data may be then stored. Algorithm 1 sketches the caching process. The method essentially compares the required triples range to the triple indices stored in an internal database, based on offset and limit parameters given in the configuration. It detects the indices of triples which are not already stored, retrieves the respective triples in chunks from the endpoint, and stores them in the database.

4.4. Link Discovery

The task of geo-links discovery requires efficient processing of spatial data, and therefore we use R-trees [36] as our underlying data structure. An R-tree is a data structure used to store and query multi-dimensional objects, in a way that and preserves spatial relations, as vicinity and nesting, among the indexed objects. An R-tree represents each object by its minimum bounding box (MBB), i.e., the smallest rectangle that encloses it, and a leaf node stores the MBB of that object and a pointer to the actual geometry. An R-tree is organized hierarchically; it groups MBBs by proximity and represents them by their MBB in a higher level of the tree. This process proceeds until all the MBBs are nested in a single bounding box - the tree root. R-Trees have shown to be efficient in processing spatial joins, to find topological relations between different data sets [37]. R-Trees support both individual elements search as well as range search, where all the items within a rectangle are retrieved.

A practical problem occurs when the data contain errors, i.e., invalid geometries. The implications of using such data are wrong results, application performance issues, etc. For this reason, geometries are examined before indexing; invalid geometries are not indexed, and thus do not participate in the link discovery.

4.5. Implementation

We use Python as our preferred programming language, since it became the language of choice for data science in general, and provides useful tools for handling geospatial data, in particular. We have experimented with the following technologies:

4.5.1. GeoPandas

Our initial implementation involved custom built caching and mapping mechanisms. We use Python’s GeoPandas library [38], which implements data structures for storing geometric types, as well as analysis tools for geospatial data. In particular, GeoPandas provides an interface for spatial joins, which allow combining observations stored in these data structures based on their spatial relations. For this purpose GeoPandas indexes geometries using R*-Tree [39], a variant of R-Tree that provides better search performance, at the cost of increased construction time. GeoPandas currently supports finding the following spatial relations: within, intersects, and contains.

We further experimented with cython [40], a language which is a superset of Python, where code can be compiled directly to C, generating efficient code. GeoPandas has been
Algorithm 1: Dataset Caching

**Input:** resource.endpoint, resource.id resource.geo, config.offset, config.limit  
**Output:** output

**Data:** Data

```python
/* create table if not exists for resource */
T ← get-table(resource,DB)
if T ∉ DB then
    T ← create-table(resource.id,resource.geo, server-offset)
create GIST-index(T.geo)
/* checking cache */
min-offset ← offset
[min-server-offset,max-server-offset] ← get-stored-offsets()
max-offset ← offset + limit - 1
if min-server-offset > 0 and max-offset < min-server-offset then
    // all queried records are before all stored ones
    /* download triples with from the given offset, save after every chunk */
    retrieve-triples(resource.endpoint, offset, limit, chunksize, T)
else if min-offset > max-server-offset then
    // all queried records are after all stored ones
    if endpoint-has-more-entries(min-offset + 1) then
        // are there any more entries at this offset?
        retrieve-triples(resource.endpoint, offset, limit, chunksize, T)
    else
        // queried entries and stored entries overlap
        intervals = list() // list of range pairs i.e., (start-range, end-range)
        if offset < min-server-offset then
            interval.append((offset, min-server-offset - 1))
            min-offset = min-server-offset
        if max-offset > max-server-offset then
            if endpoint-has-more-entries(max-server-offset + 1) then
                intervals.append((max-server-offset + 1, max-offset))
            max-offset ← max-server-offset
            missing-limit = max-offset - min-offset + 1
            /* find intervals of triple indices to be downloaded */
            missing-intervals = find-missing-data(min-offset, missing-limit)
            missing-intervals = concat(intervals, missing-intervals)
            if length(missing-intervals) > 0 then
                foreach interval ∈ missing-intervals do
                    interval-offset ← interval[0]
                    interval-limit ← interval[1] - interval-offset + 1
                    retrieve-triples(resource.endpoint, interval-offset, interval-limit, chunksize, T)
```

reimplemented in Cython in a way that optimizes the storage of geometries and should improve the performance of spatial operations.

- GiST indexes are “null safe”, therefore attempting to build an R-Tree on data which contains an empty geometry field will fail.
- GiST uses a compression technique which results in fast indexing.

4.5.2. PostgreSQL

Furthermore, we implement the system using PostgreSQL, an open source object-relational DBMS, with PostGIS extension, which provides functionality to manage geospatial data, such as geometry data types, efficient indexing, and spatial joins, and is compliant with the Open Geometry Consortium (OGC) OpenGIS specifications. PostGIS implements spatial indexing with an R-Tree-over-GiST [41]. GiST, Generalized Search Tree [42], is a height-balanced tree structure and allows arbitrary indexing schemes. The choice to use this as the backend of our is multi-fold:

- The database facilitates the implementation of the resource caching mechanism
- The source code of Geo-L is available at https://github.com/DServSys/Geo-L
### Table 1
Comparison of properties of systems for geospatial link discovery

| System | Scalability & Efficiency | Robustness | Interoperability & Flexibility |
|--------|--------------------------|------------|-------------------------------|
| Silk   | – long running time on large datasets | – instances limited to size of 64K | + standalone framework |
|        | – not evaluated for relations cover and covered by | | + has REST and programmable APIs |
| AML    | + achieves best run time for touches and intersects for LineStrings | – reaches time limit for disjoint (75 min.) | + uses ESRI, an external module for handling geometries |
|        | – long running time on large datasets for LineString/Polygon tasks for contains within covers | – no information is given about error handling | – strict linkage definition |
| Ontoldea | – long running time on large datasets | – not evaluated for disjoint | – no specification given |
|        | – not evaluated for large datasets | – no information about error handling | |
| Strabon | + run time for intersects on smaller datasets is better than that of LIMES | – did not finish any experiment on a large dataset within the time limit (2 hours) | + implements GeoSPARQL, thus is able to transform geospatial object in retrieval time |
|        | – doesn’t provide feedback about progress of its task | – no transparent error handling | |
| LIMES  | + addresses all tasks regarding topological link discovery | – data or server error interrupt whole process | + can be applied as part of a framework or as a part of an application via its API |
|        | + achieves the best run-time performance for most of the topological relations (except intersect, and touches) | | – strict linkage definition (XML), no direct SPARQL support |

5. Experimental Settings

5.1. Datasets

The evaluation has been done by finding different relations between points to polygons, and polygons to polygons in the following datasets.

- **SPOI** - Smart Points of Interest: A data set, which contains over 30 million Points of Interest important for tourism around the world [43].
- **OLU** - Open Land-Use: Maps land use on local and regional level; contains over 11 million geometries – Polygons and MultiPolygons [44].
- **NUTS** - Nomenclature of Territorial Units for Statistics: A standard for referencing European countries and their regions, for statistical processes [14].

These datasets are stored under different graphs in the SPARQL endpoint\(^7\) of the FOODIE project\(^8\). While SPOI and OLU are excellent examples for big (open) linked data, NUTS is a standard schema. NUTS geometries are not represented in WKT form, and must be be manipulated to conform to the form required by procedures of topological relations computation. Tools like LIMES, however, do not support such cases.

We compare the performance of LIMES and Geo-L with respect to both topological relations discovery and data retrieval time from endpoints.

5.2. Experiments

The performance of the Geo-L systems is evaluated in terms of runtime by conducting experiments on simulations test-sets as well as real-world scenarios. In order to compare the performance of our system with that LIMES, which is implemented using parallel processing. The task is viewed

\(7\)https://www.foodie-cloud.org/sparql
\(8\)http://www.foodie-project.eu/
as consisting of two stages: download and caching, and linking: we report the performance for each of them. The simulations enable evaluation of system performance under realistic conditions, with scenarios which otherwise might not be explored, and at the same time providing reliable way confirm their results. All experiments have been performed on a 64-bit Linux machine with an Intel Core i7-7800X CPU @ 3.50GHz and total of 12 threads (6 CPU cores × 2 threads per core).

5.2.1. Simulation
Our simulations consist of finding topological relations where the subsets of OLU dataset are used as both source and target datasets. This setting has multiple advantages: First, it allows to demonstrate the benefits of caching, regarding data sets retrieval. Additionally, the structure of the OLU set, which consists of separate geometries with non-hierarchical relations, facilitates the link quality evaluation. We used this approach to perform a preliminary comparison of three implementations on a subset of 165,000 entities (as source and target sets) and observed that the implementations which used GeoPandas performed considerably slower than the one which employed PostgreSQL with PostGIS. For example, the mapping time required for calculating the within relation was 38 seconds for the implementation which used GeoPandas, about 20 minutes for the GeoPandas cython implementation, and less than 4 seconds for the implementation which used PostgreSQL. Therefore, in the following experiments, the latter serves as our reference system.

We tested the systems with two subsets: the one contains the first 165,000 geometries, and the other the first 400,000 geometries. Figure 1 compares the dataset the retrieval times of OLU subsets for both LIMES and Geo-L. The first scenario shows that retrieval time for LIMES is about twice as long compared to Geo-L. The reason is that LIMES does not detect whether data already exist, and download the same OLU subset twice, both as source and target datasets. The second scenario emphasizes this phenomenon: Whereas Geo-L retrieves the data which has not already been downloaded yet, and does it only once, LIMES retrieves twice the subset of 400,000 geometries, which takes more than six times longer.

Moreover, LIMES stores redundant data e.g., the subset of the first 165,000 geometries is store four times, as it is contained in the 400,000 geometries subset.

Experiments have been repeated ten times for each topological relation type per subset, and the average mapping times are shown for both LIMES and Geo-L in Figure 2 and Figure 3. As can be observed, Geo-L discover topological links faster than LIMES, for all relations in these experiments. The coefficients of variation (CV) of runtimes for the different experiments were found to be low in all cases (CV < 0.1), which indicates that these results are consistent.

In addition, we found discrepancies between the links discovered by each system. For example, when looking for links of entities which stand in the equals relation in two sets with identical entities, the expected result is that each item in the source set would stand in this relation with exactly one entity of the target set, and that the size of the returned set would be equal to the size of each of sets. However, for the 165 · 10^3 OLU subset Geo-L found 164,935 links, whereas LIMES found 157,513. The 65 entities which Geo-L did not include had invalid geometries, which were detected already during construction and omitted from the search space. We examined the result computed by LIMES and noticed errors were mostly “false negatives”, i.e., valid geometries which were omitted from the result set. Also, there were also “false positives”, i.e., invalid geometries, which were included in the result set. Similar errors occurred also for other topological relations.

5.2.2. Real-World Scenarios
We experiment with topological relation discovery between pairs of geospatial resources mentioned in Section 5.1,
and compare their performance to that of LIMES. Figure 4 shows the performance, in terms of mapping runtime, on different subsets of SPOI and OLU. In this example the largest subset does not contain the other two: the first $500 \cdot 10^3$ entities of OLU contain geometries which caused LIMES system to crash, and therefore we chose a subset of the same size but specified a different offset.

Figure 5 shows the running times for mapping SPOI to NUTS with different subset sizes of SPOI. Since NUTS geometries are not represented in WKT format we used a configuration feature which defines a resource via a SPARQL query. In this case, the query also transforms the geometries into the required format. This, however, is not possible in LIMES, and therefore comparison of the systems is not presented.

Figure 6 shows mapping runtime for different subsets of OLU to NUTS, for different topological relations.
6. Conclusions

This paper presented Geo-L, a system for discovering RDF links between geospatial entities, based on topological relations. We conducted experiments to detect topological relations between points and polygons, and between polygons and polygons. The experiments show that Geo-L outperforms LIMES [26], a state-of-the-art link discovery system, for this task in several aspects:

- Scalability and efficiency: Geo-L configuration allows to form a dataset directly by the SPARQL query that defines it. This feature is, in particular, useful when data at the SPARQL endpoint are stored differently than specified for the linking task, but could be transformed into the required format through SPARQL functions.

- Download time: Datasets are cached not for a single task but are regarded as resources of their own. Thanks to its caching mechanism, Geo-L accesses the SPARQL endpoints only when data required in the dataset are missing, and expands existing datasets where possible.

- Mapping time: Geo-L utilizes PostgreSQL with PostGIS index for storing and indexing of the data. This enables efficient spatial joins between source- and target-datasets.

- Robustness: Geo-L includes multiple features that strengthen the robustness of the application.

- Caching: Geo-L caches portions of the data as they are downloaded, rather than writing the whole dataset after being downloaded. This property prevents data loss when, e.g., connection to the remote endpoint is lost.

- Mapping accuracy: Geo-L detects entities with invalid geometries (compliant to OGC OpenGIS specification) and does not include them in the search space. In addition, in several cases LIMES did not include valid geometries in the result set, whereas Geo-L correctly did.

- Interoperability and flexibility: Geo-L can be used as a stand-alone application or as a REST service (in a docker), which allows it to be integrated with other applications. The easy SPARQL-based and slim set-up of target and source configuration (as JSON) enables a very free usage of the tool.

Future work will examine relations between other types of geometries as well as explore geospatial relations based on various distance measures. The current implementation recalls the same items for each dataset once they are cached. In the future we will also address re-caching in case data at the SPARQL endpoint have been modified, an issue which is, to the best of our knowledge, not handled by other geospatial-linking systems.

References

[1] G. Klyne, Caroll, Resource Description Framework (RDF): Concepts and Abstract Syntax, 2004. URL: https://www.w3.org/TR/2004/ REC-rdf-concepts-20040210/

[2] RDF Working Group, Resource Description Framework (RDF), 2014. URL: https://www.w3.org/RDF/.

[3] S. Beechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al., Owl web ontology language reference, W3C recommendation (2004).

[4] OWL Working Group, OWL 2 Web Ontology Language Document Overview (Second Edition), 2012. URL: https://www.w3.org/2012/ pdf/REC-owl2-overview-20121211.pdf.

[5] E. Prud’hommeaux, A. Seaborne, SPARQL Query Language for RDF, W3C recommendation (2008).

[6] P. Hitzler, M. Krotzsch, S. Rudolph, Foundations of Semantic Web Technologies. Chapman and Hall/CRC, 2009.

[7] R. Battle, D. Kolas, Geosparql: enabling a geospatial semantic web, Semantic Web Journal 3 (2011) 355–370.

[8] C. Nikolaou, K. Dogani, K. Bereta, G. Garbis, M. Karpabtakis, K. Kyzirakos, M. Koubarakis, Sextant: Visualizing time-evolving linked geospatial data, Journal of Web Semantics 35 (2015) 35 – 52. URL: http://www.sciencedirect.com/science/article/pii/S1570826815000864. doi:https://doi.org/10.1016/j.websem.2015.09.004, geospatial Semantics.

[9] M. Koubarakis, K. Bereta, G. Papadakis, D. Savva, G. Stamoulis, Big, linked geospatial data and its applications in earth observation, IEEE Internet Computing 21 (2017) 87–91. URL: doi.ieeecomputersociety.org/10.1109/MIC.2017.2911438. doi:10.1109/MIC.2017.2911438.

[10] S. Auer, J. Lehmann, S. Hellmann, Linkedgeodata: Adding a spatial dimension to the web of data, in: International Semantic Web Conference, Springer, 2009, pp. 731–746.

[11] O. Čerba, K. Charvát, T. Mildorf, R. Béziriš, P. Vlach, B. Musilová, Sdi4apps points of interest knowledge base, in: Progress in Cartography, Springer, 2016, pp. 229–237.

[12] A. de León, V. Saquicela, L. M. Vilches, B. Villazón-Terrazas, F. Priyatna, O. Corcho, Geographical linked data: a spanish use case, in: Proceedings of the 6th International Conference on Semantic Systems, ACM, 2010, p. 36.

[13] C. Debruyne, E. Clinton, L. McNerney, A. Nautiyal, D. O’Sullivan, Serving Ireland’s geospatial information as linked data., in: International Semantic Web Conference (Posters & Demos), 2016.
Geo-L

[37] C. Zinke-Wehlmann, A. Kirschenbaum: Preprint submitted to Elsevier