Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 137 (2018) 139–150

Procedia Computer Science

www.elsevier.com/locate/procedia

SEMANTiCS 2018 – 14th International Conference on Semantic Systems

On the Effect of Geometries Simplification on Geo-spatial Link Discovery

Abdullah Fathi Ahmed[a], Mohamed Ahmed Sherif[a,b], Axel-Cyrille Ngonga Ngomo[a,b]

[a] *Paderborn University, Data Science Group, Pohlweg 51, D-33098 Paderborn, Germany*
E-mail: {firstname.lastname}@upb.de
[b] *Department of Computer Science, University of Leipzig, 04109 Leipzig, Germany*
E-mail: {lastname}@informatik.uni-leipzig.de

Abstract

Link discovery is central to the integration and use of data across RDF knowledge bases. Geospatial information is increasingly represented according to the Linked Data principles. Resources within such datasets are described by means of vector geometry, where link discovery approaches have to deal with millions of point sets consisting of billions of points. In this paper, we study the effect of simplifying the resources' geometries on runtime and F-measure of link discovery approaches. In particular, we evaluate link discovery approaches for computing the point-set distances as well as the topological relations among RDF resources with geospatial representation. The results obtained on two different real datasets suggest that most geospatial link discovery approaches achieve up to 67× speedup using simplification, while the average loss in their F-measure is less than 15%. Our implementation is open-source and available at http://github.com/dice-group/limes.

Keywords: Link discovery; geospatial resources; Point-set distance; Topological relation; Line simplification

1. Introduction

With the increasing growth of Linked Data in geospatial resources over recent years comes the need to develop highly scalable approaches for discovering links among such resources. As pointed out in previous works [1], only 7.1% of the links between resources connect geospatial entities. This is due to two main factors: 1) The large number of resources with geospatial representation available on the Linked Open Data (LOD), which require scalable algo-

* Corresponding author. Tel.: +49 5251 60-1764; fax: +49 5251 60-3436.
E-mail address: firstname.lastname@upb.de

1877-0509
10.1016/j.procs.2018.09.014

rithms for computing links between geospatial resources. For example, LinkedGeoData[1] contains more than 20 billion triples that describe millions of geospatial entities. 2) The vector representation of geospatial resources demands the computation of particular relations, i.e., distance and topological relations between geospatial resources. For example, finding the near by point of interest within a given radius.

According to the Linked Data principles,[2] the provision of links between knowledge bases in RDF[3] is of central importance for numerous *semantic web* tasks. However, the link discovery process become more challenging specially when dealing with geospatial resources in real-time application including structured machine learning [2], question Answering [3] and data fusion [4]. In such real-time application, the provision of explicit geospatial relations among resources is of central importance to achieving scalability.

Only a few sate-of-the-art approaches for Link Discovery (LD) have been developed to deal with geospatial data represented in RDF. For example, [1] uses the *Hausdorff* distance to compute the distance between geospatial entities. A survey of 10 point-set distance measures for LD is provided in [5]. Based on the *MultiBlock*, Silk [6] computes topological relations according to the DE-9IM standard. Recently, Radon [7] has provided an indexing method combined with space tiling that enables the efficient computation of topological relations between geospatial resources.

To the best of our knowledge, no previous work has studied the problem of discovery of geospatial relations among a simplified version of vector representations of geospatial resources. In this paper, we study the effect of applying two line-simplification algorithms as a preprocessing step prior to the discovery of geospatial relations among such resources. In particular, we consider the effect of simplification upon both efficiency of discovered relations (i.e., F-measure) and scalability of the LD approaches (i.e., runtime). The contributions of this paper are as follows:

- We present and formalize the problem of LD for geospatial resources as well as the line simplification problem.
- We study the effect of simplifying the geospatial representation of resources upon the quality of discovered relations.
- We study the speedup of various LD approaches when dealing with RDF resources with simplified geometries.
- We present an evaluation of two line-simplification approaches for different LD approaches and show that while such approaches only lose on overage 15% F-measure on the original data, they gain up to 67× speedup when applied to the simplified data.

The rest of this paper is structured as follows. We begin by introducing the Link Discovery problem over RDF knowledge bases in Section 2, where we formally define the topological and point-set distance functions. Then, we describe the line simplification problem and the two algorithms we used in this work in Section 3. In Section 4, we present our evaluation and results. We then in Section 5 discuss the state-of-the-art related work. Finally, we conclude our paper and present some future work in Section 6.

2. Link Discovery

Let K be a finite RDF *knowledge base*. K can be regarded as a set of triples $(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{R}$ is the set of all resources, $\mathcal{B}$ is the set of all blank nodes, $\mathcal{P}$ the set of all predicates and $\mathcal{L}$ the set of all literals.

The Link Discovery (LD) problem can be expressed as follows: Given two sets of resources S and T (for example hotels and gas stations) and a relation r (e.g., `:nearBy`), find all pairs $(s,t) \in S \times T$ such that $r(s,t)$ holds. The result is produced as a set of links called a *mapping*: $M_{S,T} = \{(s_i, r, t_j) | s_i \in S, t_j \in T\}$. Optionally a similarity score ($sim[0,1]$) computed by an LD tool can be added to the entries of mappings to express assurance of a computed link. Finding solutions for the LD problem is challenging due to the typically large volume and semantic heterogeneity of datasets making it difficult to meet main requirements such as high effectiveness (i.e maximize a fitness function such as F-measure) and high efficiency (i.e., minimize runtime).

[1] http://linkedgeodata.org
[2] https://www.w3.org/DesignIssues/LinkedData.html
[3] Resource Description Framework, see https://www.w3.org/RDF/.

2.1. Link Discovery of Topological Relations

The Dimensionally Extended nine-Intersection Model (DE-9IM) [8] is a topological model and a standard used to describe the spatial relations of two geometries in two-dimensional space. Since the spatial relations expressed by DE-9IM are topological, they are invariant to rotation, translation and scaling transformations [9]. The DE-9IM model is based on a 3×3 intersection matrix with the form:

$$DE9IM(g_1, g_2) = \begin{bmatrix} dim(I(g_1) \cap I(g_2)) & dim(I(g_1) \cap B(g_2)) & dim(I(g_1) \cap E(g_2)) \\ dim(B(g_1) \cap I(g_2)) & dim(B(g_1) \cap B(g_2)) & dim(B(g_1) \cap E(g_2)) \\ dim(E(g_1) \cap I(g_2)) & dim(E(g_1) \cap B(g_2)) & dim(E(g_1) \cap E(g_2)) \end{bmatrix} \quad (1)$$

where *dim* is the maximum number of dimensions of the intersection $\cap$ of the *interior*(I), *boundary*(B), or *exterior*(E) of the two geometries g_1 and g_2. The domain of *dim* is $\{-1, 0, 1, 2\}$, where -1 indicates no intersection, 0 stands for an intersection that results in a set of one or more points, 1 indicates an intersection made up of lines and 2 stands for an intersection that results in an area. A simplified binary version of $dim(x)$ with the binary domain $\{true, false\}$ is obtained using the Boolean function $\beta(dim(I(g)) = false$ iff $dim(I(g)) = -1$ and *true* otherwise. There is only a subset of the topological relations obtainable through DE-9IM that reflects the semantics of the English language [8] [10] including `equals`, `within`, `contains`, `disjoint`, `touches`, `meets`, `covers`, `coveredBy`, `intersects`, `crosses` and `overlaps`.

For discovering topological relations in RDF datasets, the Silk LD framework proposed an implementation based on the *multiBlocking* technique [6]. Recently, the Limes LD framework proposed the Radon approach [7] for the same task. We base our evaluation of topological relations only on Radon because it was proven[4] to be complete and efficient.

2.2. Link Discovery of Point-Sets Distance Measures

The input to a point-set distance measure is two sets of points, we denote $g_s = (s_1, \ldots, s_n)$ as a sequence of points to describe the source resource geometry g_s, and $g_t = (t_1, \ldots, t_m)$ as sequence of points to describe the target resource geometry g_t. We assume $(n >= m)$, where n resp. m stands for the number of distinct points in the geometry of g_s resp. g_t. A point p_i on the surface of the planet is entirely described by two values: its latitude $lat(p_i) = \varphi_i$ and its longitude $lon(p_i) = \lambda_i$. We denote points p_i as pairs (φ_i, λ_i).

The state of the art of link discovery includes many measures for computing the distance between the vector descriptions of RDF resources. We base our work in this paper on the Limes implementation of the *Hausdorff, mean, min, link* and *sumOfMin* point-set distances. Next, we will introduce both the *Hausdorff* and the *mean* distance functions as two examples of such measures. A detailed survey on state-of-the-art approaches for point-set distance measures for link discovery is available at [5].

2.2.1. Hausdorff Point-Set Distance

The *Hausdorff distance* [11] is defined as the maximum of the minimum pairwise distances between the two sets of points of source resp. target geometries. Formally, $D_{Hausdorff}(g_s, g_t) = \max_{s_i \in g_s} \left\{ \min_{t_j \in g_t} \left\{ \delta(s_i, t_j) \right\} \right\}$. where $\delta(s_i, t_j)$ is the minimum distance between two points, s_i and t_j. $\delta(s_i, t_j)$ can be accurately computed based on the *great elliptic curve distance* [12], but because of its high time complexity, most LD approaches depend on the *orthodromic distance* for computing $\delta(s_i, t_j)$. The orthodromic distance is formally defined as:

$$\delta(s_i, t_j) = R \cos^{-1} \sin(\varphi_{s_i}) \sin(\varphi_{t_j}) + \cos(\varphi_{s_i}) \cos(\varphi_{t_j}) \cos(\lambda_{s_i} - \lambda_{t_j}) \quad (2)$$

where $R = 6371\ km$ is the earth's radius, assuming the planet to be a perfect sphere.

[4] see [7] for the full poof.

2.2.2. Mean Point-Set Distance

The *mean* distance function [13] is one of the most efficient distance measures for point sets with complexity $O(n)$. First, a mean point is computed for each of the source and target point sets. Then, the distance between the two mean points is computed by using the orthodromic distance (see. Equation 2). Formally, the mean distance between the two geometries g_s, g_t is defined as: $D_{mean}(g_s, g_t) = \delta\left(\frac{1}{n}\sum_{s_i \in g_s} s_i, \frac{1}{m}\sum_{t_j \in g_t} t_j\right)$, where n and m are the sizes of g_s resp. g_t.

3. Line Simplification

Line simplification (in some literature dubbed *curve simplification*) has been adopted in many fields including computer vision, cartography and computer graphics. The input to a line simplification algorithm is a polygonized curve with n vertices composed of line segments (also called a Polyline in some contexts). The goal of a line simplification algorithm is to find an approximating polygonized curve with m vertices as output, where $m < n$. A closely related problem is to take a line with n vertices and approximate it within a defined error tolerance $\epsilon > 0$. We introduce in this work only the *Douglas-Peucker* and *VisvalingamWhyatt* algorithms as case studies due to their popularity. A detailed review of line simplification algorithms can be found in this survey [14].

3.1. The Douglas-Peucker Algorithm

The *Douglas-Peucker* algorithm [15] is the most widely used high-quality curve simplification algorithm. It was independently invented by many authors. At each iteration, the *Douglas-Peucker* algorithm tries to approximate a sequence of points by a line segment from the first point to the last point. As shown in Algorithm 1, the algorithm starts by the two end points of the input polyline. Then, it finds the point with farthest distance d from the line segment formed by the current start- and end-points. If d is below the simplification factor *dmax*, the approximation is accepted, otherwise the algorithm is recursively applied to the two polylines before and after the chosen point. The *Douglas-Peucker* algorithm, though not optimal, has generally been invented to generate the highest subjective- and objective-quality approximations when compared with many other heuristic algorithms. Its best case time cost is $\Omega(n)$, its worst case cost is $O(mn)$, and its expected time cost is about $\Theta(n \log m)$. The worst case behaviour can be improved, with some sacrifice in the best case behaviour, using a $\Theta(n \log)$ algorithm employing convex hulls [14].

Algorithm 1: Douglas-Peucker-Algorithm

```
Result: return ResultList[]
function DouglasPeucker(PointList[], epsilon);
dmax = 0;
index = 0;
for i = 2 to (length(PointList) - 1) do
    d = PerpendicularDistance(PointList[i], Line(PointList[1], PointList[end]));
    if d > dmax then
        index = i ;
        dmax = d;
    end
end
if dmax > epsilon then
    recResults1[] = DouglasPeucker(PointList[1...index], epsilon);
    recResults2[] = DouglasPeucker(PointList[index...end], epsilon);
    ResultList [] = recResults1[1...end-1] recResults2[1...end];
else
    ResultList[] = PointList[1], PointList[end];
end
```

3.2. The Visvalingam-Whyatt Algorithm

Visvalingam-Whyatt algorithm [16] (see Algorithm 2) uses the concept of *effective area* for progressive simplification of a line by point elimination. The basic idea behind this algorithm is to iteratively drop the less characteristic points. i.e., the ones which produce the least areal displacement from the current part-simplified line. The algorithm filters points on lines by a process of elimination rather than selection, while *Douglas-Peucker* Algorithm keeps the points on curves by selecting points rather than eliminating them. To delete points, the *Visvalingam-Whyatt* iteratively computes the area of all triangles formed by each three successive points. If the area of the smallest triangle is smaller than a threshold (area-tolerance), then its middle point is deleted.

Algorithm 2: Visvalingam-Whyatt Algorithm

Result: L
Input line L as a list of points, separate list R of ranked points;
Compute the effective area of each point on the line;
Delete all points with zero area and store them in a separate list;
for do
    Find the point with least effective area and call it current point;
    Delete the current point from the original list L and add it to the ranked list R with its effective area;
    Recalculate the effective area of the two adjacent points;
    if *Size of* $L = 2$ **then**
        Terminate
    end
end

4. Evaluation

We have now prepared all ingredients needed for our study. We study the impact of line simplification algorithms on the main requirements (i.e., efficiency and runtime) of link discovery over RDF knowledge bases containing geospatial entities. We evaluate the effect of simplification of geometries on the so-far used approaches in the geospatial link discovery: point-sets measures (e.g. Hausdorff and mean measures) and topological relations (e.g. contains and overlaps relations).

We aimed to answer four questions with our experimental evaluation:

- Q_1 How much performance (i.e., F-measure) each of the geospatial LD approaches loses, when to deal with the simplified geometries vs. when to deal with the original ones?
- Q_2 How well each of the geospatial LD approaches scale (i.e., runtime speedup), and when to deal with the simplified geometries?
- Q_3 Which relation is the most/least affected by the simplification process?
- Q_4 What is the run time cost of simplification?

4.1. Experimental Setup

Hardware. All the experiments were carried out on the *OCuLUS* cluster running OpenJDK 64-Bit 1.8.0_161 on Ubuntu 16.04.3 LTS. *OCuLUS* is a high performance machine located at the computer science institute in the main campus of *university Paderborn*. It consists of 9.920 processor cores 2.6 GHz *Intel Xeon "Sandy Bridge"* with main

memory of capacity 45 TB. For our created jobs, we assigned 16 CPUs and 200 GB of RAM for each job with time out of 4 hours.

Limes. For our experiments, we selected the LD framework Limes [17] to study the impact of the line simplification algorithms on the discovery of links between RDF resources with geospatial representation. We selected Limes as it implements the time-efficient approach Radon [7] for the discovery of topological relations and also because it implements various point-set distance functions [5].

Datasets . We evaluated our approach using two real-world datasets. (1) *NUTS*[5] is manually curated by the *Eurostat group of the European Commission*. NUTS contains a detailed hierarchical description of whole European regions. (2) *CORINE Land Cover* is an activity of the *European Environment Agency* that collects data regarding the land cover of European countries. CORINE Land Cover contains 44 sub-datasets ranging in size from 240 resources to 248, 242 resources.[6] We merged all CORINE Land Cover sub-datasets into one big dataset of 2, 209, 538 that we dubbed *CLC*. As Limes can only read geometries in *well known text* (WKT) format, we adopted the same preprocessing technique proposed by [7]. In particular, we preprocessed *NUTS* and *CLC* by converting the `ngeo:posList` serialization into the WKT, and lines larger than 64 KB were trimmed.

4.2. F-measure Analysis

For evaluating F-measure, we conducted four sets of experiments as follows:

In *the first set of experiments*, we used the Radon approach with the same setting in [7] for discovering the relations `equals`, `intersects`, `contains`, `covers`, `coveredBy`, `touches`, `crosses` and `overlaps`. We used the *NUTS* dataset as the source dataset and *CLC* as the target one. We then tested the impact of the line simplification algorithm of *Douglas-Peucker* [15] on Radon's performance (i.e., F-measure) when applied to the simplified data. For generating the simplified data, we applied the simplification factors of 0.05, 0.09, 0.10 and 0.2. Given that Radon is complete [7] (i.e., Radon always achieved an F-measure of 1), we used the results generated by applying Radon against the original dataset as our reference dataset. Using such reference dataset, we were able to compute the presented F-measures in Table 1. Our results show a reverse correlation between the simplification factor and F-measure. On average, Radon was able to achieve 0.94 F-measure when applied against the simplified geometries. This answer Q_1 for LD of topological relations when applied to simplified geometries using the *Douglas-Peucker* algorithm.

Relation/Factor	0.05	0.09	0.10	0.20	Average
Equals	1.00	1.00	1.00	1.00	1.00 ± 0.00
Intersects	0.99	0.97	0.97	0.94	0.97 ± 0.02
Contains	0.99	0.97	0.97	0.93	0.97 ± 0.03
Within	0.99	0.97	0.97	0.93	0.97 ± 0.03
Covers	0.99	0.97	0.97	0.93	0.97 ± 0.03
Coveredby	0.99	0.97	0.97	0.93	0.97 ± 0.03
Crosses	1.00	1.00	1.00	1.00	1.00 ± 0.00
Touches	1.00	1.00	1.00	1.00	1.00 ± 0.00
Overlaps	0.80	0.52	0.47	0.28	0.52 ± 0.21
Average	0.97 ± 0.07	0.94 ± 0.16	0.93 ± 0.17	0.90 ± 0.23	0.94 ± 0.03

Table 1. F-measures results of applying Radon against geometries generated using the *Douglas-Peucker* line simplification algorithm.

Using the same setting, we ran our *second set of experiments*, where we used the *Visvalingam-Whyatt* [18] algorithm for simplifying geometries. The results, as shown in Table 2, show that the selection of simplification parameter is more critical to the *Visvalingam-Whyatt* algorithm. Using the smallest simplification factor of 0.005 leads to the

[5] Version 0.91 (`http://nuts.geovocab.org/data/0.91/`) is used in this paper.
[6] For more details about CORINE Land Cover see `https://datahub.io/dataset/corine-land-cover`

best results with average F-measure of 0.97. Also, the reverse correlation between the simplification factor and the F-measure still holds. Those results answer Q_1 for LD approaches for topological relations when applied to simplified geometries using the *Visvalingam-Whyatt* algorithm.

Relation/Factor	0.005	0.05	0.09	Average
Equals	1.00	1.00	1.00	1.00 ± 0.00
Intersects	0.86	0.01	0.00	0.29 ± 0.49
Contains	0.86	0.01	0.00	0.29 ± 0.49
Within	0.86	0.01	0.00	0.29 ± 0.49
Covers	0.86	0.01	0.00	0.29 ± 0.49
Coveredby	0.86	0.01	0.00	0.29 ± 0.49
Crosses	1.00	1.00	1.00	1.00 ± 0.00
Touches	1.00	1.00	1.00	1.00 ± 0.00
Overlaps	0.86	0.03	0.00	0.30 ± 0.49
Average	0.94 ± 0.08	0.56 ± 0.52	0.56 ± 0.53	0.69 ± 0.22

Table 2. F-measures results of applying Radon against geometries generated using the *Visvalingam-Whyatt* line simplification algorithm.

For the *the third set of experiments*, we carried out a deduplication task for the whole NUTS dataset. i.e., we set the NUTS dataset as both the source S and target T datasets. To measure how well each of the point distance measures performs, we first created a reference mapping $M = \{(n, n) \in NUTS\}$, then we measured the distance between each of the geometries in $S \times T$. We then computed the F-measure achieved within the experiment by comparing the pairs in M' (generated by applying the point-set distances) with those in M. We used the implementations of the *Hausdorff, Mean, Min, Link Sum of minimums* and Surjection from Limes. The results of those experiments are listed as the last column of Table 3. We then used the *Douglas-Peucker* to simplify all the NUTS geometries with the simplification factors $\{0.05, 0.9, 0.1, 0.2\}$. As shown in Table 3, the simplification factors of 0.1 and 0.2 achieved the best average F-measure of 0.82. One of the most interesting results of those experiments was that the majority of point-set measures were not only able to achieve the same F-measure of the original dataset when applied on the simplified data but also outperform the F-measure on the original data in the cases of the *Hausdorff, mean* and *Min* measures.

Measure/Factor	0.05	0.9	0.1	0.2	0.3	Average	$F_{original}$
Hausdorff	0.90	0.91	0.91	0.91	0.91	0.91 ± 0.00	0.88
Mean	0.94	0.94	0.94	0.94	0.94	0.94 ± 0.00	0.94
Min	0.14	0.16	0.16	0.21	0.25	0.18 ± 0.04	0.13
Link	0.95	0.95	0.94	0.94	0.94	0.94 ± 0.00	0.94
SumOfMin	0.95	0.95	0.94	0.94	0.94	0.94 ± 0.00	0.94
avarege	0.77 ± 0.36	0.78 ± 0.35	0.78 ± 0.35	0.79 ± 0.32	0.80 ± 0.31		0.77 ± 0.36

Table 3. Average F-measures results of applying a deduplication task on the NUTS dataset using the point-set distance measures implementations in Limes. As input, we used both the original NUTS geometries (results are in the last column) and simplified geometries generated using the *Douglas-Peucker* line simplification algorithm.

In the *fourth set of experiments*, we used the same setting of the last set of experiments except for the simplification algorithm. In this set of experiments, we generated a simplified version of the NUTS geometries using the *Visvalingam-Whyatt* algorithm with the simplification factors $\{0.005, 0.05, 0.1\}$. We then computed the F-measure based on the following distance measure functions *Hausdorff, Mean, Min, Link* and *Sum of minimums*. The results in Table 4 show the comparison between the F-measure obtained from a simplified version of the data and the F-measure (denoted $F_{original}$) obtained from an original version of the data. The results clearly show the sensitivity of F-measure to changing the simplification factor, for instance, when the simplification factor equals 0.005 the average of F-measure is 0.77, while it dropped down to 0.26 with a simplification factor of 0.1.

Measure/Factor	0.005	0.05	0.1	Average	$F_{original}$
Hausdorff	0.88	0.24	0.02	0.38 ± 0.45	0.88
Mean	0.94	0.94	0.92	0.93 ± 0.02	0.94
Min	0.13	0.13	0.13	0.13 ± 0.00	0.13
Link	0.94	0.14	0.01	0.37 ± 0.51	0.94
Sum of Min	0.94	0.94	0.24	0.71 ± 0.40	0.94
Avarege	0.77 ± 0.36	0.48 ± 0.42	0.26 ± 0.38		0.77 ± 0.36

Table 4. F-measure results of applying a deduplication task on the NUTS dataset using the point-set distance measures implementations in Limes. As input, we used both the original NUTS geometries (results are in the last column) and simplified geometries generated using the *Visvalingam-Whyatt* line simplification algorithm.

4.3. Runtime Analysis

In order to answer Q_2, we evaluated the speedup gained by applying LD approaches to the simplified geometries. We measured the run time while performing the aforementioned four sets of experiments.

Figure 1 shows the runtime results for the first set of experiments, i.e., we measured the run time of applying Radon to discover topological relations when applied to the original datasets vs. when applied to the simplified datasets using the *Douglas-Peucker* algorithm. On average, Radon has provided a 4.9× speedup over its performance when applied to the original datasets. Moreover, there is a direct correlation between the achieved speedup and the simplification parameter. In particular, the lowest speedup of 3.7× is achieved when applying the simplification factor of 0.05 and the speedup monotonically increases up to 6.1× when applying the simplification factor of 0.2.

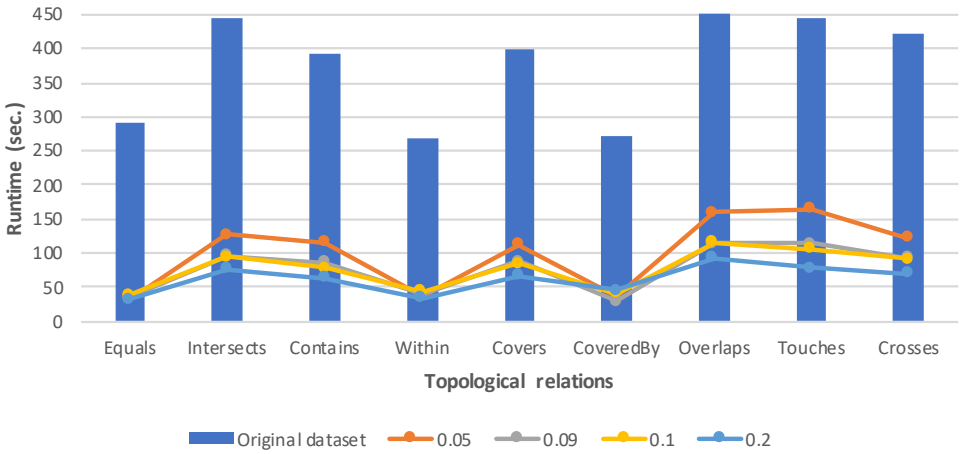


Fig. 1. Runtimes of Radon's implementation of topological relations LD for original $NUT \times CLC$ datasets vs. the runtimes of the simplified datasets using the *Douglas-Peucker* algorithm with simplification factors of $\{0.05, 0.09, 0.1, 0.2\}$.

For the second set of experiments, we also measured the runtimes when applying Radon against the original datasets vs. the simplified datasets using the *Visvalingam-Whyatt* algorithm. The results are shown in Figure 2. On average, Radon achieved 49.2× speed up, with a maximum 67.3× speedup in the case of a simplification factor of 0.09 but only 4.2× speedup with a simplification factor of 0.005.

Using the same technique, we measured the runtime for the third set of experiments. The results are presented in Figure 3. The point-set distance achieved an average speedup of 9.2 when applied to the simplified geometries using the *Douglas-Peucker* algorithm (min. = 2×, max. = 19.8×).

Figure 4 shows the results of the runtimes of the fourth set of experiments. The point-set measures achieved only an average speedup of 2.1 when applied to simplified geometries using the *Visvalingam-Whyatt* algorithm (only 1.1 in the case of a simplification factor of 0.005).

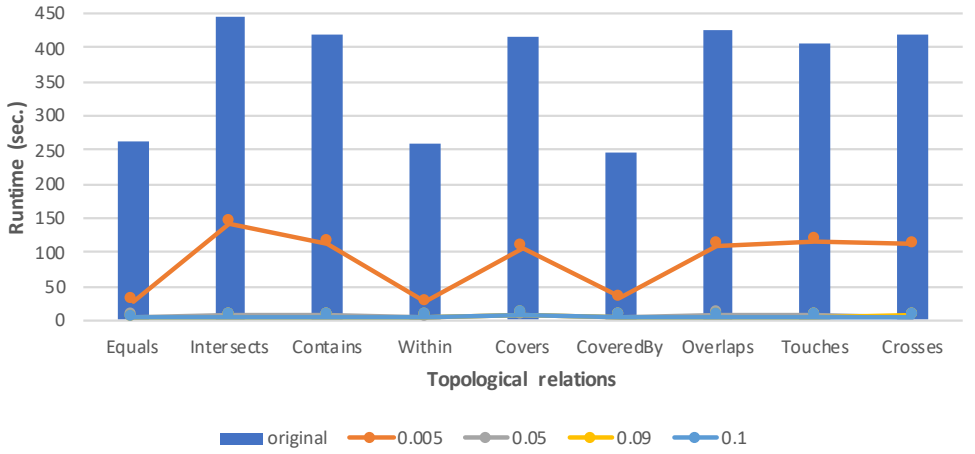


Fig. 2. Run times of Radon's implementation of topological relations LD for original $NUT \times CLC$ datasets vs. runtimes of the simplified datasets using the *Visvalingam-Whyatt* algorithm with simplification parameters $\{0.005, 0.05, 0.1\}$.

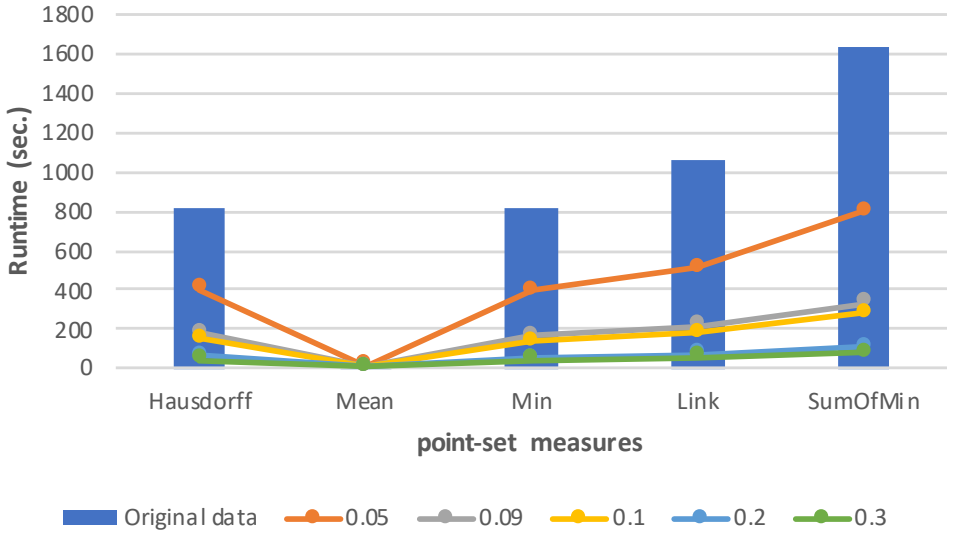


Fig. 3. Run times of Limes implementation of point-set measures LD deduplication for original NUT dataset vs. the run time of simplified dataset using the *Douglas-Peucker* algorithm with simplification parameters $\{0.05, 0.09, 0.1, 0.2, 0.3\}$.

4.4. LD Relations Analysis

From all the previous sets of experiments, we can now answer Q_3. In the case of the topological relations, the F-measure of `overlap` relation is the most affected by the *Douglas-Peucker* simplification. This can be seen in Table 1. Also, the `equals,crosses` and `touches` are not affected at all by any simplification (see Tables 1 and 2). In the case of point-set measures, the F-measure of `min` measure is the most affected, while all the other relations not only achieve the F-measure of the original data but also outperform it in many cases (see Tables 3 and 4).

For runtime of topological relations, the `equal` relation achieved the best speedup in the case of the *Douglas-Peucker* simplification (see Figure 1), while the `coverdBy` had the best speedup when using the *Visvalingam-Whyatt* algorithm (see Figure 2). For the runtime of point-set relations, the `mean` relation achieves the shortest run time even without any simplification, while the `sunOfMin` relation has the least speedup (See Figures 3 and 4).

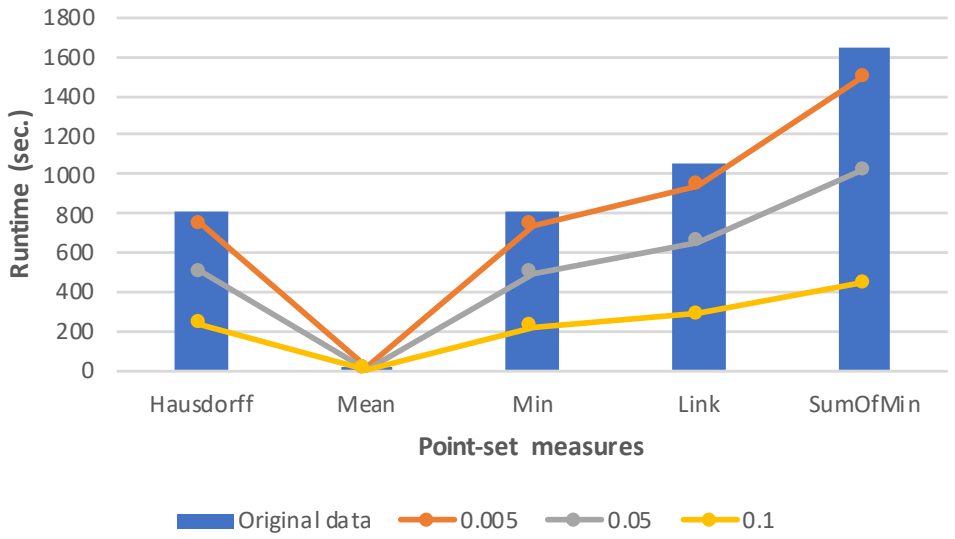


Fig. 4. Run times of Limes implementation of point-set measures LD deduplication for original *NUT* dataset vs. the run time of simplified dataset using the *Visvalingam-Whyatt* algorithm with simplification parameters $\{0.005, 0.05, 0.1\}$.

4.5. Simplification Runtime Analysis

To answer Q_4, we measured the runtime cost of applying both the *Douglas-Peucker* and *Visvalingam-Whyatt* line simplification algorithms. Because of the paper space restrictions, we present only the result of the first algorithm for topological relations. We computed the average simplification time needed while doing the first and second sets of experiments together with the average time to run Radon for each of the topological relations vs. the time needed to run Radon against the original datasets. The result is detailed in Figure 5.

In case we want to discover all the topological relations at once, Figure 5(a) shows the total run time needed for simplification on the left. Note that the simplification process is only done once for all topological relations. The total time for running Radon for all relations on the simplified data is plotted next in the figure. Next, comes the total time of running Radon in addition to the simplification time for all relations. Finally, the time for running Radon on the original data is plotted. As we can see, as we perform the simplification process once and use the simplified data for extracting all the relations, Radon is able to achieve on average 2.4× speedup.

Figure 5(b) shows the average run time needed for running Radon for only one relation on the original data vs. running it on a simplified one. As we can see, the simplification time is on average greater than the average time for a single Radon topological relation discovery task (see first and last columns in Figure 5 (b)). This clearly shows that using simplification for the discovery of a single relation is sub optimal.

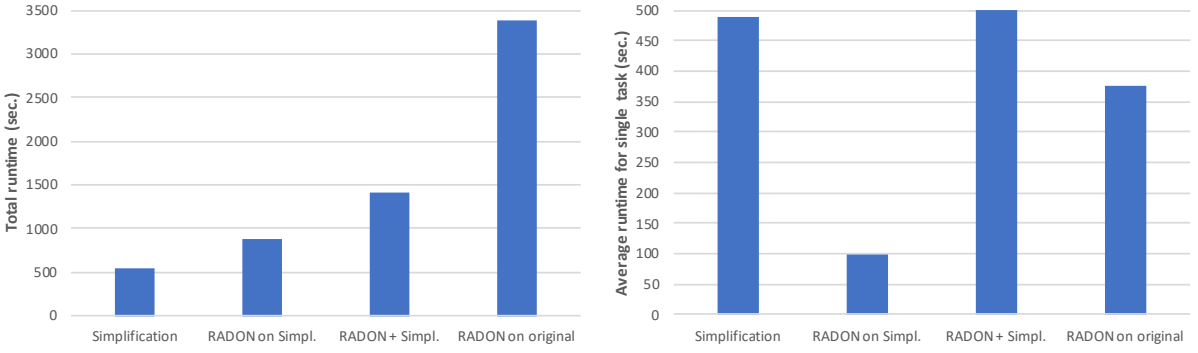
A complete answer for Q_4 would be that the more relations there are to be discovered, the more speedup will be gained from the usage of simplification. Moreover, the simplification runtime cost would be very high once a single relation discovery is required. I.e, we recommend not using any simplification for a single relation discovery.

5. Related Work

The work presented in this paper is related to two main areas of research: link discovery of geospatial relations and line simplification. We give a brief overview of these research areas in the following sections.

5.1. Link Discovery of Geospatial Relations

The discovery of topological relations has been paid little attention in previous research related to Link Discovery [19]. The reduction-ratio-optimal approach Orchid [1] optimizes the computation of point-sets based on the dis-



(a) Total runtime for all topological relations.

(b) Average runtime of a single topological relation.

Fig. 5. Runtimes of RADON on the original data vs. simplified data using the *Douglas-Peucker* algorithm.

tance among geospatial entities. The main idea of ORCHID is to apply space tiling for both source and target resources and compare only resources within a given range. While the use of ORCHID in [1] is based only on the *Hausdorff* metric, the work at [5] extends ORCHID to other point-set distance metrics such as *mean* and *sum of minimums*. RADON [7] proposes an indexing approach combined with space tiling and swapping technique for the efficient calculation of topological relations between geospatial data. Based on the *MultiBlocking* technique, [6] proposed an approach for spatial LD. Both the ORCHID and RADON are implemented in the the LD framework LIMES, while the *MultiBlocking* technique is based on the LD framework SILK. A review of the current state of LD frameworks is in [20].

5.2. *Line Simplification*

Line simplification techniques have been used in various fields including computer vision, cartography and computer graphics. [21] introduced an approach for line simplification based on image processing, which is specifically designed for raster data. An area-reserving subdivision simplification algorithm is proposed in [22], where the algorithm presents a set of topology constraints for rendering map data on the screen. In Advanced Driving Assistance Systems (ADAS) [23], the *Douglas-Peucker* line simplification algorithm is used to fix the total number of vertices for the resultant polygon from static free space extraction, which is convenient to ADAS and automotive restrictions. Recently, *Douglas-Peucker* has been used to simplify the massive Asia-Pacic Data Center trajectories data from *China Automatic Identification System* (AIS) for data-driven based automatic maritime routing [24]

6. Conclusions and Future Work

We present a study of the usage of simplification as a preprocessing step of LD approaches for the linking of RDF resources with geospatial representation. We studied the behaviour of two categories of geospatial linking approaches (i.e, the topological relations and point-set distances) when provided with simplified geometries vs. when provided with the original ones. In particular, we studied both the F-measure and the run time of each approach. Our evaluations show that such approaches achieve on average an F-measure of 0.94 when using the *Douglas-Peucker* simplification algorithm and 0.69 when using the *Visvalingam-Whyatt* algorithm. In addition, LD approaches gain up to 19.8× speedup when dealing with simplified geometries generated by the *Douglas-Peucker* algorithm and up to 67.3× when using the *Visvalingam-Whyatt*. This suggests that the usage of geometries simplification will be of a great help for real-time applications such as question answering where runtime is the key performance factor and result completeness comes in the second place.

In future work, we will study the provision of a simplification algorithm able to guarantee an minimum input F-measure. Also, we will determine for each set of relations the best geometry simplification algorithm together with its best parameters to achieve the best results (i.e., maximum F-measure and minimum runtime). Moreover, we will

study an optimization of a scalable geometries simplification approach able to deal with big volume of resources, each consisting of big number points such as the ones that exist in current RDF datasets.

Acknowledgments. This work has been supported by Eurostars Project SAGE (GA no. E!10882), the BMVI project the LIMBO (GA no. 19F2029C), the DFG project LinkingLOD (project no. NG 105/3-2), the BMWI Project GEISER (project no. 01MD16014) as well as the H2020 projects SLIPO (GA no. 731581) and HOBBIT (GA no. 688227).

References

[1] A.-C. Ngonga Ngomo, Orchid - reduction-ratio-optimal computation of geo-spatial distances for link discovery, in: Proceedings of ISWC 2013, 2013.
[2] M. Sherif, A.-C. Ngonga Ngomo, J. Lehmann, WOMBAT - A Generalization Approach for Automatic Link Discovery, in: 14th Extended Semantic Web Conference, Portorož, Slovenia, 28th May - 1st June 2017, Springer, 2017.
[3] J. Lehmann, T. Furche, G. Grasso, A.-C. Ngonga Ngomo, C. Schallhart, A. Sellers, C. Unger, L. Bühmann, D. Gerber, K. Höffner, D. Liu, S. Auer, Deqa: Deep web extraction for Question Answering, in: Proceedings of ISWC, 2012.
URL http://jens-lehmann.org/files/2012/iswc_deqa.pdf
[4] M. Sherif, A.-C. Ngonga Ngomo, J. Lehmann, Automating RDF dataset transformation and enrichment, in: Proceedings of 12th Extended Semantic Web Conference, Springer, 2015.
URL http://svn.aksw.org/papers/2015/ESWC_DEER/public.pdf
[5] M. A. Sherif, A.-C. N. Ngomo, A systematic survey of point set distance measures for link discovery, Semantic Web Journal.(Cited on page 18.).
[6] P. Smeros, M. Koubarakis, Discovering spatial and temporal links among rdf data., in: LDOW@ WWW, 2016.
[7] M. A. Sherif, K. Dreßler, P. Smeros, A.-C. Ngonga Ngomo, RADON - Rapid Discovery of Topological Relations, in: Proceedings of The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017.
URL https://svn.aksw.org/papers/2017/AAAI_RADON/public.pdf
[8] E. Clementini, J. Sharma, M. J. Egenhofer, Modelling topological spatial relations: Strategies for query processing, Computers & graphics 18 (6) (1994) 815–822.
[9] M. J. Egenhofer, R. D. Franzosa, Point-set topological spatial relations, International Journal of Geographical Information System 5 (2) (1991) 161–174.
[10] E. Clementini, P. Di Felice, P. Van Oosterom, A small set of formal topological relationships suitable for end-user interaction, in: International Symposium on Spatial Databases, Springer, 1993, pp. 277–295.
[11] D. P. Huttenlocher, G. A. Klanderman, W. J. Rucklidge, Comparing images using the hausdorff distance, IEEE Transactions on pattern analysis and machine intelligence 15 (9) (1993) 850–863.
[12] B. Bowring, The direct and inverse solutions for the great elliptic line on the reference ellipsoid, Bulletin géodésique 58 (1) (1984) 101–108.
[13] R. O. Duda, P. E. Hart, D. G. Stork, et al., Pattern classification, Vol. 2, Wiley New York, 1973.
[14] P. S. Heckbert, M. Garland, Survey of polygonal surface simplification algorithms, Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE (1997).
[15] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Cartographica: The International Journal for Geographic Information and Geovisualization 10 (2) (1973) 112–122.
[16] M. Visvalingam, J. D. Whyatt, Line generalisation by repeated elimination of points, The Cartographic Journal 30 (1) (1993) 46–51.
[17] A.-C. N. Ngomo, S. Auer, Limes-a time-efficient approach for large-scale link discovery on the web of data., in: IJCAI, 2011, pp. 2312–2317.
[18] M. Visvalingam, P. J. Williamson, Simplification and generalization of large scale data for roads: a comparison of two filtering algorithms, Cartography and Geographic Information Systems 22 (4) (1995) 264–275.
[19] A.-C. N. Ngomo, S. Auer, J. Lehmann, A. Zaveri, Introduction to linked data and its lifecycle on the web, in: Reasoning Web International Summer School, Springer, 2014, pp. 1–99.
[20] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, E. Rahm, A survey of current link discovery frameworks, Semantic Web 8 (3) (2017) 419–436.
[21] Y. Shen, T. Ai, Y. He, A new approach to line simplification based on image processing: A case study of water area boundaries, ISPRS International Journal of Geo-Information 7 (2) (2018) 41.
[22] T. Mendel, Area-preserving subdivision simplification with topology constraints: Exactly and in practice, in: 2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, 2018, pp. 117–128.
[23] H. M. Eraqi, J. Honer, S. Zuther, Static free space detection with laser scanner using occupancy grid maps, arXiv preprint arXiv:1801.00600.
[24] S.-k. Zhang, G.-y. Shi, Z.-j. Liu, Z.-w. Zhao, Z.-l. Wu, Data-driven based automatic maritime routing from massive ais trajectories in the face of disparity, Ocean Engineering 155 (2018) 240–250.