Building Linked Spatio-Temporal Data from Vectorized Historical Maps

Basel Shbita[1(✉)], Craig A. Knoblock[1], Weiwei Duan[2], Yao-Yi Chiang[2], Johannes H. Uhl[3], and Stefan Leyk[3]

[1] Information Sciences Institute and Department of Computer Science, University of Southern California, Los Angeles, USA
{shbita,knoblock}@isi.edu
[2] Spatial Sciences Institute and Department of Computer Science, University of Southern California, Los Angeles, USA
{weiweidu,yaoyic}@usc.edu
[3] Department of Geography, University of Colorado Boulder, Boulder, USA
{johannes.uhl,stefan.leyk}@colorado.edu

Abstract. Historical maps provide a rich source of information for researchers in the social and natural sciences. These maps contain detailed documentation of a wide variety of natural and human-made features and their changes over time, such as the changes in the transportation networks and the decline of wetlands. It can be labor-intensive for a scientist to analyze changes across space and time in such maps, even after they have been digitized and converted to a vector format. In this paper, we present an unsupervised approach that converts vector data of geographic features extracted from multiple historical maps into linked spatio-temporal data. The resulting graphs can be easily queried and visualized to understand the changes in specific regions over time. We evaluate our technique on railroad network data extracted from USGS historical topographic maps for several regions over multiple map sheets and demonstrate how the automatically constructed linked geospatial data enables effective querying of the changes over different time periods.

Keywords: Linked spatio-temporal data · Historical maps · Knowledge graphs · Semantic web

1 Introduction

Historical map archives contain valuable geographic information on both natural and human-made features across time and space. The spatio-temporal data extracted from these documents are important since they can convey where and

A. Harth et al. (Eds.): ESWC 2020, LNCS 12123, pp. 409–426, 2020.
https://doi.org/10.1007/978-3-030-49461-2_24

when changes took place. This type of data enables long-term analysis, such as detecting the changes in railroad networks between several map editions of the same region and can support decision-making related to the development of transportation infrastructure. Many applications assessing geographic changes over time typically require searching, discovering, and manually identifying relevant data. This is a difficult and laborious task that requires domain knowledge and familiarity with various data formats, and the task is often susceptible to human error.

Linked geospatial data has been receiving increased attention in recent years as researchers and practitioners have begun to explore the wealth of geospatial information on the Web. Recent technological advances facilitate the efficient extraction of vectorized information from scanned historical maps and other digital data to facilitate the integration of the extracted information with other datasets in Geographic Information Systems (GIS) [3–5,11].

Previous work on creating linked data from geospatial information has focused on the problem of transforming the vector data into RDF graphs [2,6,12]. However, this line of work does not address the issue of geospatial entity linkage, e.g., building linked geospatial data with a semantic relationship between vector data elements across maps of the same region. To better support analytical tasks and understand how map features change over time, we need more than just the extracted vector data from individual maps. In addition to the vector data extracted from historical maps, we need the relationship between the vector data elements (segments) composing the desired features across maps, and the detailed metadata and semantic properties that describe that data. To enable change analysis over time and across multiple spatial scales, we present an unsupervised approach to match, integrate, and relate vector data of map features using linked data principles and provide corresponding semantics for the representation of the data.

The task we address here is that given geospatial vector data extracted from numerous map editions covering the same region, we want to construct a knowledge graph depicting all the feature segments that represent the original data, their relations and their semantics across different points in time. Using the constructed knowledge graph, we enable tackling more specific downstream analysis tasks. These may include the visualization of feature changes over time and the exploration of supplementary information related to the region using additional knowledge bases we link to our graph.

As an example, consider the maps shown in Fig. 1a where changes in the Louisville, New Albany and Chicago railroad system have occurred between 1886 and 1904. Figure 1b shows the railroad segment changes between the different map editions. Segments that have been added are marked in red and segments that have been removed are marked in blue. Assuming we have the data available as vector data (which can be generated from scanned maps using [5]), our task in such a setting would be to construct a knowledge graph describing the different segment elements in these maps with a conventional semantic representation for the railroad line segments in each map edition. This would include objects

(a) Louisville, New Albany and Chicago railroad system maps in 1886 (left) and 1904 (right)

(b) Visual representation of the change in the railroad system; additions are in red, removals are in blue

Fig. 1. An example of a railroad system change over time (Color figure online)

from a list of common geographic features (points, lines, or polygons), their geocoordinates, and their matching temporal coverage to allow easy analysis and visualization.

Our approach is not only helpful in making the data widely available to researchers but also enables users to answer complex queries in an unsupervised manner, such as investigating the interrelationships between human and environmental systems. Our approach also benefits from the open and connective nature of linked data. Compared to existing tools such as PostGIS[1] that can only handle queries related to geospatial relationships within local databases, linked data can utilize many widely available knowledge sources, such as GeoNames and OpenStreetMap[2], in the semantic web and enable rich semantic queries.

Once we convert the map data into linked data, we can execute SPARQL queries to depict the changes in map segments over time and thus accelerate and improve spatio-temporal analysis tasks. Using a semantic representation that includes geospatial features, we are able to support researchers to query and visualize changes in maps over time and allow the development of data analytics applications that could have great implications for environmental, economic or societal purposes.

The rest of the paper is organized as follows. We present our proposed pipeline in Sect. 2. In Sect. 3 we evaluate our approach by automatically building a linked data representation for a series of railroad networks from historical maps covering two different regions from different time periods. Related work is discussed in Sect. 4. We conclude, discuss, and present future work in Sect. 5.

[1] https://postgis.net/.
[2] https://www.openstreetmap.org/.

2 Building and Querying Linked Spatio-Temporal Data

2.1 Taxonomy

Figure 2 shows the taxonomy of terms used in this paper. We refer to the elements composing the vector data extracted from historical maps as segments. These segments may be decomposed to small building blocks (we refer to these as segments or segment elements), which are encoded as WKT (well-known text representation of geometry) multi-line strings. These strings are composed from a collection of tuples of latitude and longitude coordinates.

2.2 Overall Approach

The unsupervised pipeline we propose for constructing linked data from vector data of extracted map feature segments consists of several major steps as illustrated in Fig. 3. These steps can be summarized as follows:

1. Automatically partition the feature segments from the original shapefiles (vector data) into collections of segments using a spatially-enabled database service (i.e., PostGIS) to form groups of segments (see Sect. 2.3).
2. Utilize a reverse-geocoding service (i.e., OpenStreetMap) to map the partitioned feature segment geometric literals to existing instances in the semantic web (i.e., LinkedGeoData [1]) (see Sect. 2.4)
3. Construct the knowledge graph by generating RDF triples following a pre-defined semantic model using the data we generated from previous steps (see Sects. 2.5 and 2.6)

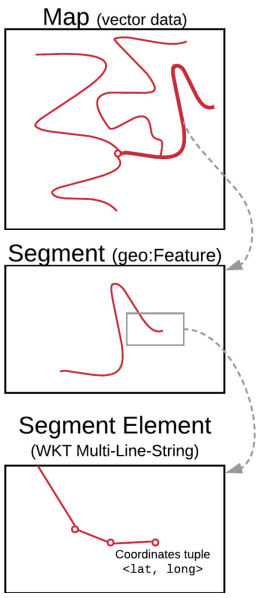


Fig. 2. Taxonomy of terms used in the paper

Once the RDF data is deployed, users can easily interact with the feature segment data and perform queries (Sect. 2.7), which allow end-users to visualize the data and supports the development of spatio-temporal applications.

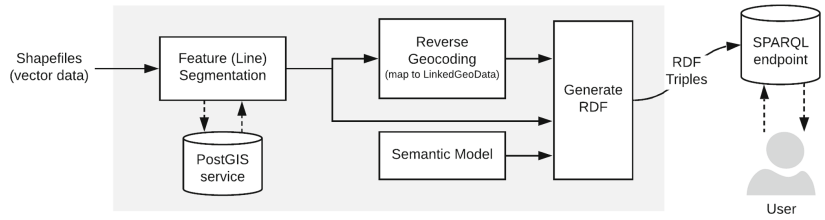
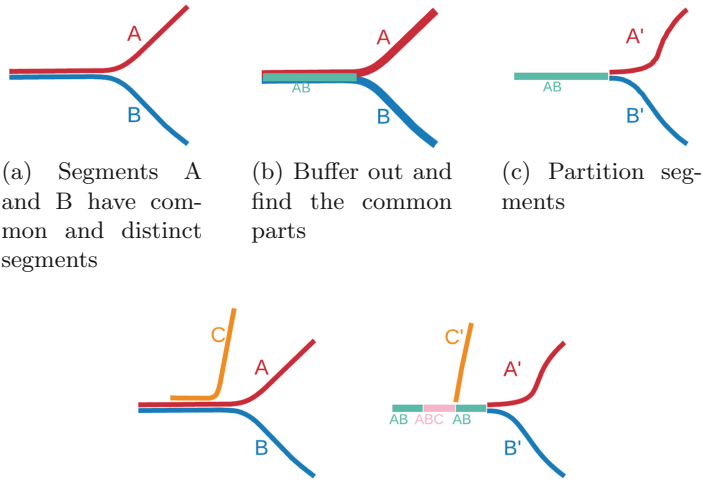


Fig. 3. Pipeline for constructing linked data from vector data

2.3 Automatic Feature Segmentation

The first task in our pipeline is the creation of segment partitions (elements) that can represent the various geographic features (e.g., railroad network) across different maps (same region, different map editions) in a granular and efficient fashion. This task can be classified as a common entity matching/linking and entity "partitioning" task. Given two segment elements from two map editions of the same region, we want to identify which parts of those elements coincide and thus represent the same parts of the feature. This allows us to generate segment groups of elements that are more granular and can be used to represent the common and the distinct parts of features.

Consider a simplified example consisting of segments (geometry of line type) from two map editions (Fig. 4a), where segment A is from an older map edition and segment B is from the latest map edition with a part of the feature that has been changed. In order to detect those parts that exist in both segments, we split each of these segments into several sub-segments based on the intersection of the segments, as shown in Figs. 4b and 4c. When a third source (another map edition also containing the feature), C, is added, a similar segmentation process is executed as shown in Figs. 4d and 4e.



(a) Segments A and B have common and distinct segments

(b) Buffer out and find the common parts

(c) Partition segments

(d) Segment C is added

(e) Final partitioning

Fig. 4. Segments partitioning process: spatial buffers are used to identify the same segments considering potential positional offsets of the data

As we mentioned earlier, we use a spatially-enabled database service to simplify handling data manipulations over feature segments. PostGIS is a powerful PostgreSQL extension for spatial data storage and query. It offers various functions to manipulate and transform geographic objects in databases. To handle

the task mentioned earlier efficiently and enable an incremental addition of maps over time, we implemented Algorithm 1. The algorithm performs the segmentation tasks by employing several PostGIS application programming interface (API) calls over the geometries of our segments that we loaded in the database.

In detail, the procedure works as follows. The **for** loop in line 1 iterates over each collection of vector data (segment geometries) in each map edition. For each iteration, lines 2–3 extract the set of feature segments from a given shapefile and map them to a node i (representing the segment) which is then added to the directed acyclic graph (data structure) $\mathcal{G}$. The graph $\mathcal{G}$ will eventually hold our final set of segments and record their relations in a data structure within the graph. Line 4 retrieves the leaf nodes from graph $\mathcal{G}$ to list $\mathcal{L}$. In the first iteration list $\mathcal{L}$ is empty. In the next iterations it will include the nodes which represent the segments that are similar to and distinct from segments from other map editions in the same degree (distance in the graph from the root) as their parent node. Then, for each "leaf" node we execute the following steps:

1. **Segment Intersection.** Line 6 extracts the set of feature segments from the leaf segment k. Line 7 performs an intersection of segment i with segment k by creating a buffer for the vector data and intersecting this buffer with the vector data in segment k. This results in the set of segments named F_α which is then mapped to segment α and added to the graph $\mathcal{G}$ as a child node of i and k (line 8).
2. **Segment Difference (local "Subtraction").** In line 9, we generate the segment elements of the data in segment k that are not in segment i, which results in the set of segment elements named F_γ. Then, F_γ is mapped to segment γ and added to the graph $\mathcal{G}$ as a child node of k (line 10).
3. **Segment Union-Difference (global "Subtraction").** Once we finish going over the list of leaves, we compute the unique segments that exist in the added segment (from the lastly added map edition) by reducing the union of the leaf node intersections (with previous processed maps) from the original map segment i as described in line 12. This results in the set of segment elements named F_δ. Then, in line 13, F_δ is mapped to segment δ and added to the graph $\mathcal{G}$ as a child node of i.

The above procedure is demonstrated in Figs. 4a, 4b and 4c where segments A and B are nodes i and k, respectively and AB, B' and A' are nodes α, γ and δ, respectively. The relations between the nodes in graph $\mathcal{G}$ carry a semantic meaning between the different segments and will play a critical role in the RDF generation and query mechanism since they represent the relations between the segment elements across different time extents of the same region. The hierarchical relationship is built with respect to these attributes and allows us to retrieve the segments that will represent the changes in feature segments when running a query.

Data: a set $\mathcal{M}$ of segments for different map editions of the same region (shapefiles)
Result: a directed acyclic graph $\mathcal{G}$ of feature segment objects (nodes) and their relations

foreach $i \in \mathcal{M}$ **do**
  $\mathcal{F}_i$ = set of geometry features (segment elements) in i;
  $\mathcal{G}$.add($i \mapsto F_i$);
  $\mathcal{L}$ = list of current leaf nodes in $\mathcal{G}$;
  foreach $k \in \mathcal{L}$ **do**
    $\mathcal{F}_k$ = set of geometry features (segment elements) in k;
    $\mathcal{F}_\alpha = \mathcal{F}_i \bigcap \mathcal{F}_k$;
    $\mathcal{G}$.add($\alpha \mapsto F_\alpha$) and set i, k as parents of α;
    $\mathcal{F}_\gamma = \mathcal{F}_k \setminus \mathcal{F}_\alpha$;
    $\mathcal{G}$.add($\gamma \mapsto F_\gamma$) and set k as parent of γ;
  end
  $\mathcal{F}_\delta = \mathcal{F}_i \setminus (\bigcup_{j \in \mathcal{L}} F_j)$;
  $\mathcal{G}$.add($\delta \mapsto F_\delta$) and set i as parent of δ;
end

Algorithm 1: The feature segments partitioning algorithm

2.4 Reverse Geocoding and Linking to Linked Open Vocabularies

In the last two decades there has been a major effort in publishing data following semantic web and linked data principles. There are now tens of billions of facts spanning hundreds of linked datasets on the web covering a wide range of topics. To better describe the semantics of data and reuse well-documented vocabularies in the linked data ecosystem, we propose a simple mechanism to allow linking the extracted segments from the processed historical maps to additional knowledge bases on the web. This is again a task of entity matching; this time it is with an entity in an external knowledge base.

Our proposed method is based on reverse geocoding. Reverse geocoding is the process of mapping the latitude and longitude measures of a point or a bounding box to a readable address or place name. Examples of these services include the GeoNames reverse geocoding web service[3] and OpenStreetMap's API.[4] These services permit for a given location the identification of nearby street addresses, places, areal subdivisions, etc.

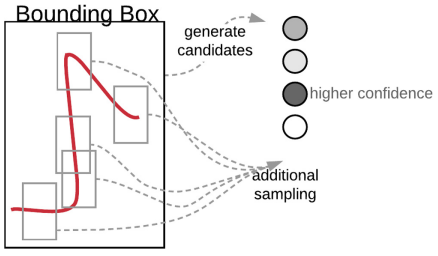


Fig. 5. Method for acquiring external knowledge base instances

The "geo-linking" process is depicted in Algorithm 2 and illustrated in Fig. 5. We start with individual features extracted from the original maps that are of

[3] http://www.geonames.org/export/reverse-geocoding.html.
[4] https://wiki.openstreetmap.org/wiki/API.

known type (T in Algorithm 2). In the case of the data we present later in the evaluation section, we are starting with the extracted segments of **railroads**, so we know the feature type we are searching for. Each input segment s here is an individual node in graph $\mathcal{G}$ from Sect. 2.3. We first generate a global bounding box for segment s and execute a reverse-geocoding API call to locate instances of type T on the external knowledge base, as described in lines 1–2. Some of these instances do not share any geometry parts with our inspected segment. Thus, we randomly sample a small number ($N = 30$) of coordinate pairs (`Points`), from those composing the segment s, as seen in lines 3–5, to gain more confidence in the detected instances. Finally, we reduce the list by filtering out the candidates that have a single appearance (high confidence that it is not part of the segment but is found in the bounding box). These resulting instances are used in later stages to enrich the knowledge graph we are constructing with additional semantics.

Data: segment s, number of samples N, feature type T
Result: list $\mathcal{L}$ of instances on LinkedGeoData composing our input segment s
1 B_s = bounding box wrapping s;
2 $\mathcal{L}$ = reverse-geocoding(B_s, T); // returns LinkedGeoData instances of T in B_s
for 1...N **do**
3 $\quad e$ = randomly sample a `Point` in segment s;
4 $\quad E$ = reverse-geocoding(e, T);
5 $\quad \mathcal{L}$.add(E);
6 **end**
7 filter out instances with a single appearance in $\mathcal{L}$;
8 return $\mathcal{L}$;

Algorithm 2: The "geo-linking" algorithm

2.5 Semantic Model

In order to provide a representation with useful semantic meaning and universal conventions, we defined a semantic model that builds on GeoSPARQL.[5] The OGC GeoSPARQL standard defines a vocabulary for representing geospatial data on the web and is designed to accommodate systems based on qualitative spatial reasoning and systems based on quantitative spatial computations.

Our approach towards a robust semantic model was motivated by the OpenStreetMap data model, where each feature is described as one or more geometries with attached attribute data. In OpenStreetMap, `relations` are used to organize multiple `nodes` or `ways` into a larger whole. For example, an instance of a bus route running through three different `ways` would be defined as a `relation`.

In GeoSPARQL, the class type `geo:Feature` represents the top-level feature type. It is a superclass of all feature types. In our model, each instance of this class represents a single segment (element) extracted from the original vector data. It is possible to compose different collections of segments representing the specified feature in some time extent using this instance and a property of type `geo:sfWithin` or `geo:sfContains` to denote a decomposition to smaller

[5] https://www.opengeospatial.org/standards/geosparql.

elements. The use of such properties enables application-specific queries with a backward-chaining spatial "reasoner" to transform the query into a geometry-based query that can be evaluated with computational geometry. Additionally, we use the property `geo:sfOverlaps` with subjects that are instances from the LinkedGeoData knowledge base in order to employ the web as a medium for data and spatial information integration following linked data principles. Furthermore, each instance has properties of type `dcterms:date`, to denote the time extent of the segment, and `dcterms:created`, to denote the time in which this segment was generated to record provenance data.

Complex geometries are not human-readable as they consist of hundreds or thousands of coordinate pairs. Therefore, we use dereferenceable URIs to represent the geometric objects instead. Using a named node in this capacity means that each geometric object has its own URI as opposed to the common blank-node approach often used with linked geometric objects. Thus, each segment instance holds a property of type `geo:hasGeometry` with a subject that is an instance of the class `geo:Geometry`. This property refers to the spatial representation of a given feature. The class `geo:Geometry` represents the top-level geometry type and is a superclass of all geometry types.

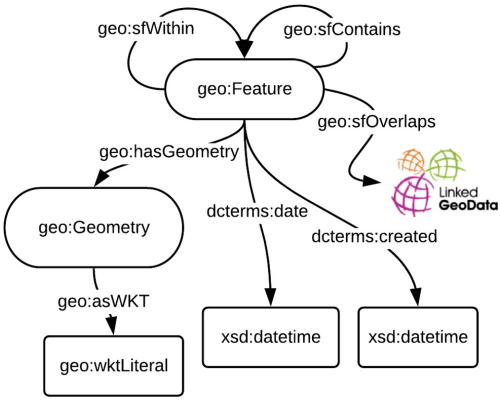


Fig. 6. Semantic model for linked maps

In order to describe the geometries in a compact and human-readable way we use the WKT format for further pre-processing. The `geo:asWKT` property is defined to link a geometry with its WKT serialization and enable downstream applications to use SPARQL graph patterns. The semantic model we described above is shown in Fig. 6.

2.6 Incremental Linked Data

Following the data extraction and acquisition tasks described in the previous section, we can now produce a structured standard ontologized output in a form of a knowledge graph that can be easily interpreted by humans and machines, as linked data.

This hierarchical structure of our directed acyclic graph $\mathcal{G}$ and its metadata management allows us to avoid an update across all the existing published geographic vector data as linked data and instead handle the computations incrementally once a new map edition of the feature is introduced.

The approach we present is complete and follows the principles of Linked Open Data by:

- Generating URIs as names for things, without the need to modify any of the previously published URIs once further vector data from the same region is available and processed
- Maintaining existing relations (predicates) between instances (additional relations may be added, but they do not break older ones)
- Generating data as a machine-readable structured data
- Using standard namespaces and semantics (e.g., GeoSPARQL)
- Linking to additional resources in the web of Linked Open Data.

2.7 Querying

The semantic model we presented in Sect. 2.5 and its structure provide a robust solution enabling a coherent query mechanism to allow a user-friendly and efficient interaction with the data.

In order to clarify the query construction idea, we describe the elements that are needed for a general query "skeleton" from which we can establish more complicated queries to achieve different outcomes as required. Figure 7 shows a query which retrieves all the leaf node segments (i.e., the "skeleton" query). As shown in the figure, we first denote that we are interested in a `geo:Feature` that has a geometry in `WKT` format which gets stored in the variable `?wkt` as shown in lines 3–4 (the variable we visualize in Fig. 14). Line 5 restricts the queried segments (`geo:Feature`s) to leaf nodes only (in graph $\mathcal{G}$). This is done by discarding all the nodes that hold a predicate of type `geo:sfContains`, which means that we retrieve only the nodes that are the "building blocks".

```
1  SELECT ?f ?wkt
2  WHERE {
3    ?f a geo:Feature ;
4       geo:hasGeometry [ geo:asWKT ?wkt ] .
5    FILTER NOT EXISTS { ?f geo:sfContains _:_ } }
```

Fig. 7. Our SPARQL query "skeleton"

This is important due to the incremental nature (and the way graph $\mathcal{G}$ "grows"): as we mentioned previously, every time we add an additional map edition of the feature, we decompose the existing leaf segments (smallest building blocks) to a new layer of leaf segments (newer and smaller building blocks, if subject to decomposition) and its metadata migrates to the lowest level of segment leaves. This property makes our solution robust and suggests an efficient way of querying, avoiding the need to "climb up" the graph for more complicated ("composed") segments.

If, for example, we are interested to see the full version of the segment from a specific time, we can add the clause {`?f dcterms:date <...> .`} inside the `WHERE` block (lines 2–6). If we are interested to see the changes from a different time, we can add an additional clause {`MINUS { ?f dcterms:date <...> . }`} as well. The syntax and structure of the query allows an easy adaptation for additional tasks such as finding the distinct segment parts from a specific time or finding the segment parts that are shared over three, four or even more points in time or map editions. The nature of our knowledge graph provides an intuitive approach towards writing simple and complex queries.

3 Evaluation

We tested two datasets of vector railroad data (the inspected feature) that were extracted from the USGS historical topographic map archives,[6,7] each of which covers a different region and is available for different time periods.

In this section, we present the results, measures and outcomes of our pipeline when executed on railroad data from a collection of historical maps of a region in Bray, California from the years 1950, 1954, 1958, 1962, 1984, 1988, and 2001 (shown in Fig. 8) and from a region in Louisville, Colorado from the years 1942, 1950, 1957 and 1965. Our primary goal is to show that our proposal provides a complete, robust, tractable, and efficient solution for the production of linked data from vectorized historical maps.

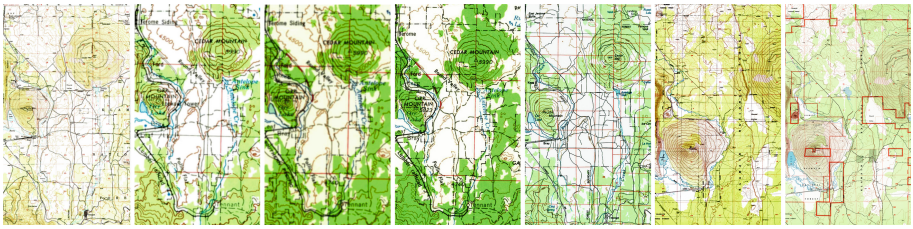


Fig. 8. Historical maps in Bray, California from 1950, 1954, 1958, 1962, 1984, 1988 and 2001 (left to right, respectively)

3.1 Evaluation on the Feature Segmentation Process

In order to evaluate the performance of this task, we look into the runtime and the number of generated nodes (in graph $\mathcal{G}$) for each region. The number of vector lines in the segment geometry (column '`# vecs`'), resulting runtimes (column '`Runtime`', measured in seconds) and total number of nodes following each sub-step of an addition of another map edition (column '`# nodes`') are depicted in Tables 1 and 2 for each region.

As seen in Tables 1 and 2, the railroad segments extracted from these maps vary in terms of "quality". That is, they have a different number of vector lines that describe the railroads and each one has a different areal coverage. This is caused by the vector extraction process (see [5]) and is not within

Table 1. Segmentation Statistics for Bray

Year	# vecs	Runtime (s)	# nodes
1954	2382	<1	1
1962	2322	36	5
1988	11134	1047	11
1984	11868	581	24
1950	11076	1332	43
2001	497	145	57
1958	1860	222	85

[6] https://viewer.nationalmap.gov.
[7] http://historicalmaps.arcgis.com/usgs/.

the scope of this paper. We also acknowledge that the quality and scale of the original images used for the extraction affects these parameters but we do not focus on such issues. We treat these values and attributes as a ground truth for our process.

Table 2. Segmentation Statistics for Louisville

Year	# vecs	Runtime (s)	# nodes
1965	838	<1	1
1950	418	8	5
1942	513	5	8
1957	353	4	10

First, we notice that the growth of the number of nodes in graph $\mathcal{G}$ is reasonable due to the way the railroad network changes in practice. Further, the runtime of each sub-step is also reasonable and tractable. As expected, the first two map editions (for both areas) generate results within less than a minute, requiring at most three computations: one geometry intersection between two elements and two additional subtractions: a local and a global one (as explained in Sect. 2.3). The following runtimes show that our computation cost is not exponential in practice. By inspecting Tables 1 and 2 we observe that the segmentation runtime somewhat depends on two factors: the number of vectors in the geometries and the number of nodes that exist in the graph. The more geometry elements we have and the more geometries exist, the more operations we need to run.

These results are not surprising because "leaves" in the graph will only be partitioned in case it is "required", that is, they will be partitioned to smaller unique parts to represent the different segments they need to compose. With the addition of map editions, we do not necessarily add unique parts since changes do not occur between all map editions. This shows that the data processing is not necessarily becoming more complex in terms of space and time, thus, providing a solution that is feasible and systematically tractable.

3.2 Evaluation on Linking to LinkedGeoData

In the process of linking our data to LinkedGeoData, we are interested in the evaluation of the running time and correctness (precision and recall) of this task.

The running time is linearly dependent on the number of nodes in graph $\mathcal{G}$, the number of samples using the OpenStreetMaps API, and the availability of the API. The API response time averages 2 s for each sample. The execution time for the set of maps from the region in Bray took approximately an hour (85 nodes) and only a few minutes for Louisville (10 nodes). This provides a feasible solution to a process that runs only once for a collection of map editions.

Due to the unsupervised characteristic of the linking task, we had to manually inspect and label the sets of instances found in each bounding box that we query for each segment. The measure we present here is in terms of coverage. It is the number of instances we detected out of the number of instances that are available on the external knowledge base and which make up the inspected railroad segment. Nonetheless, in terms of detecting which railroad it is (label), we are able to achieve 100% accuracy (by taking a majority vote on the labels of instances we queried).

As an example, let us observe the instances in the LinkedGeoData knowledge base that are linked to a segment that holds railroad lines in the map edition from 1950 in the data from Bray, as shown in Fig. 9. All of the LinkedGeoData instances linked to this segment show an `rdf:type` of type `lgdo:AbandonedRailway`, as seen in red in Fig. 10. This shows our ability to enrich and link our graph to the web of linked data.

```
<http://linkedmaps.isi.edu/69> a geo:Feature ;
    dcterms:created "2019-12-10T15:23:23"^^xsd:dateTime ;
    dcterms:date "1950-01-01T00:00:00"^^xsd:dateTime ;
    geo:hasGeometry <http://linkedmaps.isi.edu/69_sc_m_2091b7b4c0> ;
    geo:sfOverlaps <http://linkedgeodata.org/triplify/way177559134>,
        <http://linkedgeodata.org/triplify/way177559138> ;
    geo:sfWithin <http://linkedmaps.isi.edu/41> .
```

Fig. 9. A segment representation in RDF

We have set up a baseline for comparison with our "geo-linking" method. The baseline approach returns the set of all instances found in the bounding box. This is the list of candidates we generate in the first step of our method, without the additional sampling and filtering steps we have described in Sect. 2.4.

The precision, recall and F1 scores of each method over each dataset are shown in Table 3. The first row (`BRA-baseline`) provides the baseline's results applied on the Bray dataset. The second row (`BRA`) shows the results of our method when applied on the Bray dataset. The third (`LOU-baseline`) and fourth rows (`LOU`) show the results of the baseline method and our method, respectively, applied on the Louisville dataset. Due to the different geometries, areal coverage and available data in the external knowledge base for each region, our measure shows different scores for each dataset. However, our method achieves much higher F1 scores than the baseline (0.774 and 0.909 compared to 0.323 and 0.625 respectively) and achieves an acceptable score for this task.

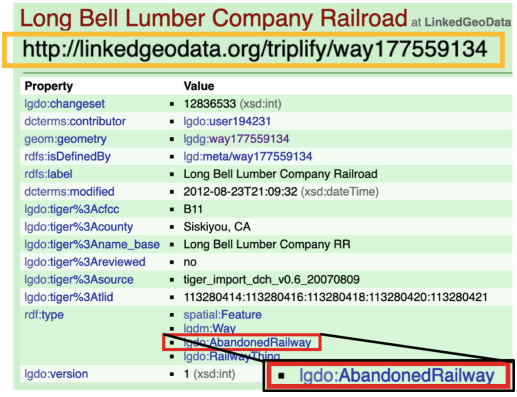


Fig. 10. Screenshot of an instance at LinkedGeoData knowledge base (Color figure online)

Table 3. "Geo-linking" Results

	Precision	Recall	F1
BRA-baseline	0.193	1.000	0.323
BRA	0.800	0.750	**0.774**
LOU-baseline	0.455	1.000	0.625
LOU	0.833	1.000	**0.909**

3.3 Evaluation on Querying the Data

We execute several query examples over the knowledge graph we constructed in order to measure our model in terms of query time, validity, and effectiveness. We had a total of 914 triples for the Bray region dataset and 96 triples for Louisville.

The generated RDF triples would be appropriate to use with any Triplestore. We hosted our triples in Apache Jena.[8] Jena is relatively lightweight, easy to use, and provides a programmatic environment. Each SPARQL query response was visualized using the Google Maps API.

```
SELECT ?f ?wkt WHERE {
  ?f a geo:Feature ;
     geo:hasGeometry [ geo:asWKT ?wkt ] ;
     dcterms:date "1962-01-01T00:00:00"^^xsd:dateTime ;
     dcterms:date "2001-01-01T00:00:00"^^xsd:dateTime .
  FILTER NOT EXISTS { ?f geo:sfContains _:_ } }
```

Fig. 11. SPARQL query generating similar railroad segments in both 1962 and 2001

In the first type of query we want to identify the parts of the railroad that remain unchanged in two different map editions (different time periods) for each region (i.e., Fig. 11). Table 4 shows the query-time results in the row labelled `SIM-BRA` for the region in Bray and by `SIM-LOU` for the region in Louisville. We executed a hundred identical queries for each area across different time extents to measure the robustness of this type of query.

We repeated the process for a second type of query to identify the parts of the railroad that were removed or abandoned between two different map editions for each region (i.e., Fig. 12). `DIFF-BRA` is the result for Bray and `DIFF-LOU` for Louisville.

The third type of query retrieves the parts of the railroad that are unique to a specific edition of the map (i.e., Fig. 13). `UNIQ-BRA` is the result for Bray and `UNIQ-LOU` for Louisville.

```
SELECT ?f ?wkt WHERE {
  ?f a geo:Feature ;
     geo:hasGeometry [ geo:asWKT ?wkt ] ;
     dcterms:date "1962-01-01T00:00:00"^^xsd:dateTime .
  FILTER NOT EXISTS { ?f geo:sfContains _:_ }
  MINUS { ?f dcterms:date "2001-01-01T00:00:00"^^xsd:dateTime . } }
```

Fig. 12. SPARQL query generating railroad segments present in 1962 but not in 2001

The execution times (average, minimum and maximum) are shown in Table 4. We notice that the query times are all in the range of 8–28 (ms) and do not seem to change significantly with respect to the number of map editions we process or the complexity of the query we compose. This is based on the fact that the data from Bray was constructed from 7 map editions compared to the data from Louisville, which was constructed from 4 map editions.

[8] https://jena.apache.org/.

```
SELECT ?f ?wkt WHERE {
  ?f a geo:Feature ;
     geo:hasGeometry [ geo:asWKT ?wkt ] ;
     dcterms:date "1958-01-01T00:00:00"^^xsd:dateTime .
  FILTER NOT EXISTS { ?f geo:sfContains _:_ }
  ?f dcterms:date ?date . }
GROUP BY ?f ?wkt
HAVING (COUNT(DISTINCT ?date) = 1)
```

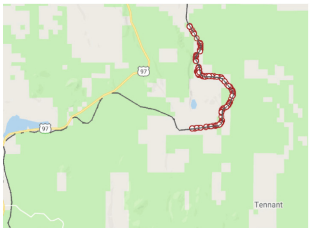
Fig. 13. SPARQL query generating unique railroad parts that are present only in 1958

In order to evaluate the validity of our graph we observe the visualized results of the query in Fig. 11, which are shown in Fig. 14a. This figure shows in red the unchanged segments between the years 1962 and 2001 for Bray. We notice that the geometries we retrieve do match what we observe in the original shapefiles (the line marked in black over the two maps represents the current railway, which has not changed since 2001). The results of the query in Fig. 12 are shown in Fig. 14b. This figure shows in blue the parts of the railroad that were abandoned between 1962 to 2001.

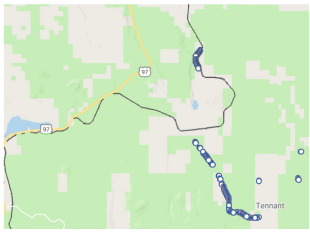
The query results above establish high confidence in our model, showing that we can easily and effectively answer complex queries in a robust manner. Overall, we demonstrated that our approach and the proposed pipeline can be effectively used to automatically construct linked data from geospatial information.

Table 4. Query Time Statistics (in milliseconds)

	Avg	Min	Max
SIM-BRA	12	10	18
SIM-LOU	11	9	20
DIFF-BRA	10	8	20
DIFF-LOU	10	9	14
UNIQ-BRA	14	8	28
UNIQ-LOU	15	9	17



(a) The parts of the railroad that are similar in 1962 and 2001, marked in red



(b) The parts of the railroad that are present in 1962 but are not present in 2001, marked in blue

Fig. 14. Examples of railroad system changes over time (Color figure online)

4 Related Work

Much work has been done on mapping vector data into RDF graphs. Kyzirakos et al. [6] developed a semi-automated tool for transforming geospatial data from their original formats into RDF using R2RML mapping. Usery et al. [12] presented a method for converting point and vector data to RDF for supporting query and analysis of geographic data. Our work differs by building linked geospatial data with a meaningful semantic relations between vector segments across map editions of the same region and thus for different points in time whereas prior approaches focus on the publication of data from raw files or relational databases.

Existing work on geographical data conflation [7,9] focuses on reconciling different sources for improving data accuracy and in combining incompatible geospatial data. This line of work does not consider the semantics and nature of linked data as we address it. Their work can be used within our framework to strengthen our geometric entity-matching tasks.

Bernard et al. [2] present a semantic matching algorithm for automatically detecting, describing and publishing descriptions of changes occurring in region partitions, and their geometries, in the linked open data web. Their work focuses on Territorial Statistical Nomenclatures (TSNs) which are used for the collection of regional statistics. Their semantic matching algorithm is bounded to an ontology where geometric changes are described either as deformed, expanded or contracted with no regard to composition or decomposition of geometries as we do. Our approach goes further since it provides a richer semantic capability, decomposition of geometries and data interlinking.

Prudhomme et al. [8] present an automatic approach for geospatial data integration in the web. Although this line of work tackles the problem of entity resolution for geospatial data, it is difficult to compare our approach to theirs because they tackle the task of ontology-matching (by maximizing semantic information from files using natural language processing) while ours addresses a complementary problem to that theirs. Our primary goal is the completeness and tractability of building the linked geodata, using semantics to provide a model that is linked to other online sources such as LinkedGeoData.

5 Discussion and Future Work

With the increasing availability of digitized geospatial data from historical map archives, better techniques are required to enable end-users and non-experts to easily understand and analyze geographic information across time and space. Existing techniques rely on human interaction and expert domain knowledge. In this paper, we addressed this issue and presented an unsupervised, effective approach to integrate, relate and interlink geospatial data from digitized resources and publish it as semantic-rich, structured linked data that follows the Linked Open Data principles.

The evaluation we presented in Sect. 3 shows that our approach is feasible and effective in terms of processing time, completeness and robustness. The

segmentation process runs only once for newly added resource, and does not require re-generation of "old" data since our approach is incremental. In case a new map edition emerges for the same region, we only need to process the newly added segments. Thus, data that has been previously published will continue to exist with a proper URI and will be preserved over time.

In a scenario that includes contemporary maps that change very quickly, we expect our method to require longer computation time, but would still be tractable with respect to the changes happening in the map geometries. As we mentioned in Sect. 2.3, the breakdown of segments depends on the complexity of the actual changes in the original topographic map. Further, the quality and level of detail of the original vector data play a significant role in the final RDF model as we mentioned in Sect. 3.1. In our ongoing work on this topic, we are looking into techniques to normalize and denoise the original vector data for the purpose of higher quality output.

Our approach has several limitations, one of them is in a form of a hyper-parameter that governs the buffer size we use in the process of the partitioning of segments to smaller parts. We currently set this parameter manually but we believe such parameter can be learned from the data or estimated using some heuristics. Another limitation in the system is the usage of a single external knowledge base. We will address this issue by expanding the ability to utilize additional knowledge bases to enrich our linked data with more semantics from Linked Open Vocabularies [13].

Although we only evaluated railroad data, it can be easily extended for highways, wetlands, forests and additional natural or man-made features with minor adjustments of their geometries and the filtering term that is used in the geocoding API (e.g., railroads). Since the topological relations of the geometries are expressed (via GeoSPARQL) and computed (via PostGIS) according to the DE-9IM model [10], a 2D model, it allows us to apply it over polygon geometries in addition to line geometries. For continuous 2D surfaces, such as wetlands and forests, that are expressed in polygon geometries, we can handle their boundaries similarly. We already started exploring this line of work.

In future work, we also plan to investigate the possibility of using multiple machines for faster processing. This is possible since there are computations in the segment-partitioning algorithm that are independent of each other and can be executed in parallel in the same iteration over a single map edition (lines 7 and 9 for different ks in Algorithm 1). This will enable a faster processing time and strengthen the effectiveness of our solution.

Resources. The source code, original datasets and the resulting RDF, can be found here: https://github.com/usc-isi-i2/linked-maps.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant Nos. IIS 1564164 (to the University of Southern California) and IIS 1563933 (to the University of Colorado at Boulder).

References

1. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: adding a spatial dimension to the web of data. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_46
2. Bernard, C., Plumejeaud-Perreau, C., Villanova-Oliver, M., Gensel, J., Dao, H.: An ontology-based algorithm for managing the evolution of multi-level territorial partitions. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 456–459. ACM (2018)
3. Chiang, Y.-Y., Duan, W., Leyk, S., Uhl, J.H., Knoblock, C.A.: Using Historical Maps in Scientific Studies: Applications, Challenges, and Best Practices. SG. Springer, Cham (2020). https://doi.org/10.1007/978-3-319-66908-3
4. Chiang, Y.Y., Leyk, S., Knoblock, C.A.: A survey of digital map processing techniques. ACM Comput. Surv. (CSUR) **47**(1), 1–44 (2014)
5. Duan, W., Chiang, Y., Knoblock, C.A., Leyk, S., Uhl, J.: Automatic generation of precisely delineated geographic features from georeferenced historical maps using deep learning. In: Proceedings of the AutoCarto (2018)
6. Kyzirakos, K., Vlachopoulos, I., Savva, D., Manegold, S., Koubarakis, M.: GeoTriples: a tool for publishing geospatial data as RDF graphs using R2RML mappings. In: Terra Cognita, 6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web, in Conjuction with ISWC, pp. 33–44 (2014)
7. Li, L., Goodchild, M.F.: An optimisation model for linear feature matching in geographical data conflation. Int. J. Image Data Fusion **2**(4), 309–328 (2011)
8. Prudhomme, C., Homburg, T., Ponciano, J.J., Boochs, F., Cruz, C., Roxin, A.M.: Interpretation and automatic integration of geospatial data into the semantic web. Computing **102**, 1–27 (2019)
9. Ruiz, J.J., Ariza, F.J., Urena, M.A., Blázquez, E.B.: Digital map conflation: a review of the process and a proposal for classification. Int. J. Geogr. Inf. Sci. **25**(9), 1439–1466 (2011)
10. Strobl, C.: Dimensionally extended nine-intersection model (DE-9IM). In: Shekhar, S., Xiong, H. (eds.) Encyclopedia of GIS, pp. 240–245. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-35973-1_298
11. Uhl, J.H., Leyk, S., Chiang, Y.Y., Duan, W., Knoblock, C.A.: Automated extraction of human settlement patterns from historical topographic map series using weakly supervised convolutional neural networks. IEEE Access (2019)
12. Usery, E.L., Varanka, D.: Design and development of linked data from the national map. Semant. Web **3**(4), 371–384 (2012)
13. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked open vocabularies (LOV): a gateway to reusable semantic vocabularies on the web. Semant. Web **8**(3), 437–452 (2017)