

Building Spatio-Temporal Knowledge Graphs from Vectorized Topographic Historical Maps

Basel Shbita [a,*], Craig A. Knoblock [a], Weiwei Duan [b], Yao-Yi Chiang [b], Johannes H. Uhl [c] and Stefan Leyk [c]

[a] *Information Sciences Institute, University of Southern California, CA, USA*
E-mails: shbita@isi.edu, knoblock@isi.edu
[b] *Spatial Sciences Institute, University of Southern California, CA, USA*
E-mails: weiweidu@usc.edu, yaoyic@usc.edu
[c] *Department of Geography, University of Colorado Boulder, CO, USA*
E-mails: johannes.uhl@colorado.edu, stefan.leyk@colorado.edu

Abstract.
Historical maps provide rich information for researchers in many areas, including 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 changes in transportation networks or the decline of wetlands or forest areas. Analyzing changes over time in such maps can be labor-intensive for a scientist, even after the geographic features have been digitized and converted to a vector format. Knowledge Graphs (KGs) are the appropriate representations to store and link such data and support semantic and temporal querying to facilitate change analysis. KGs combine expressivity, interoperability, and standardization in the Semantic Web stack, thus providing a strong foundation for querying and analysis. In this paper, we present an automatic unsupervised approach to convert vector geographic features extracted from multiple historical maps into contextualized spatio-temporal KGs. The resulting graphs can be easily queried and visualized to understand the changes in different regions over time. We evaluate our technique on railroad networks and wetland areas extracted from the United States Geological Survey (USGS) historical topographic maps for several regions over multiple map sheets and editions. We also demonstrate how the automatically constructed linked data (i.e., KGs) enable effective querying and visualization of changes over different points in time.

Keywords: Spatio-Temporal Knowledge Graphs, Knowledge Graphs, Linked Spatio-Temporal Data, Linked Data, Semantic Web, Data Integration, Historical Maps, Digital Humanities

1. Introduction

Historical map archives contain valuable geographic information on both natural and human-made features across time and space [1] and are increasingly available in systematically acquired digital data formats [2, 3]. The spatio-temporal data extracted from these documents are important since they can convey when, where and what changes took place [4]. For example, this type of data enables the detection of long-term changes in railroad networks or studying the evolution of wetlands within the same region over time and thus can support decision-making related to the development of transportation infrastructure or questions related to land conservation, landscape ecology, or long-term land development and human settlement. Many applications assessing geographic changes over time typically involve searching, discovering, and manually identifying, digitizing, and integrating relevant data. This is a difficult and laborious task that requires domain knowledge and familiarity with various data sources, data formats and working environments, and the task is often susceptible to human error [5].

Generally, there are two types of geospatial data, namely, raster data and vector data. Recent technolog-

*Corresponding author. E-mail: shbita@isi.edu.

ical advances facilitate the efficient extraction of vectorized information from scanned historical maps [1, 2, 4, 6, 7]. Vector data provide a compact way to represent real-world features within Geographic Information Systems (GIS). Every geographic feature can be represented using one of three types of geometries: (i) points (depict discrete locations such as addresses, wells, etc.), (ii) lines (used for linear features such as roads, rivers, etc.) or (iii) polygons (describe enclosed areas like waterbodies, islands, etc.). This paper tackles the core problem of detecting how geometries change over time, focusing on linear and polygonal features.

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 [8, 9]. In addition, the need for tracing geographic features over time or across documents has emerged for different applications [4]. Furthermore, growing attention has been paid to the integration and contextualization of the extracted data with other datasets in a GIS [10, 11].

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. We need to tackle the following challenges:

1. **Entity generation and interlinking.** Generate and interlink the geospatial entities ("building-block" geometries originating from the vector data) that constitute the desired features (i.e., railroad lines or wetland areas) and can exist across map editions
2. **External geo-entity linking.** Contextualize and link the generated entities to external resources to enable data augmentation and allow users to uncover additional information that does not exist in the original map sheets
3. **Representation.** Represent the resulting data in a structured and semantic output that can be easily interpreted by humans and machines, and adheres to the principles of the Semantic Web

Previous work on creating linked data from historical geospatial information has focused on the problem of transforming a single instance of a map sheet or a formatted geospatial data source into Resource Description Framework (RDF) graphs [12–14]. However, this line of work does not address the issue of entity interlinking that is essential for building linked geospatial data for the task of change analysis with a semantic relationship between geospatial entities across map editions of the same region, which could be part of a large collection of topographic map sheets. Similar work is concerned with only a specific type of geometry, such as points, as in [10, 15], or is limited to a specific physical feature (i.e. flooded areas [16] or wildfires [17]). Our work does not impose such limitations.

Our approach is not only helpful in making the RDF 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], which can only handle queries related to geospatial entities and relationships within local databases, linked data can utilize many widely available knowledge sources in the semantic web, such as OpenStreetMap (OSM) [18], GeoNames [19], and LinkedGeoData [20], to enable rich semantic queries.

Providing contextual knowledge can help explain or reveal interactions of topographic changes to further spatio-temporal processes. For example, the external-linking enables augmentation of the contained geographic information with data from external KBs, such as historical population estimates. Once we convert the map data into linked data, we can execute SPARQL queries to identify the changes in map features over time and thus accelerate and improve spatio-temporal analysis tasks. Using a semantic representation that includes geospatial attributes, we can 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.

This paper is based on our earlier conference and workshop papers [21, 22]. The paper expands our methods to support polygon-based geographic features, elaborates on the details of the algorithms, and provides a more extensive evaluation of the methods. In addition, we use OpenStreetMap instead of LinkedGeoData as an external KB to obtain the most up-to-date information and provide a solution that does not depend on time-sensitive third-party data dumps.

1.1. Problem definition

The task we address here is as follows: Given geographic vector data extracted from multiple map edi-

[1]https://postgis.net/

Fig. 1. New Albany (OH) and Chicago (IL) railroad system maps in 1886 (left) and 1904 (right)

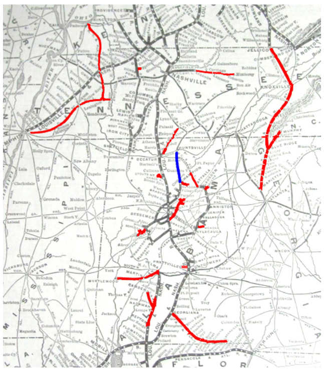


Fig. 2. Visual representation of the change in the New Albany (OH) and Chicago (IL) railroad system between the years 1886 and 1904; additions are in red, removals are in blue

tions of the same region, we aim to automatically construct a knowledge graph depicting all the geographic features that represent the original data, their relations (interlinks), 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 (e.g., population data, elevation data, etc.) related to the region originating from an external knowledge base.

As an example, consider the maps in Figure 1 where changes in the New Albany (OH) and Chicago (IL) railroad system have occurred between 1886 and 1904. Figure 2 shows the railroad line changes between the different map editions. Line segments that have been added are marked in red and line 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 approaches such as in Duan et al. [6]), our task in such a setting would be to construct a knowledge graph describing the shared line segments that are shared across these maps or unique in individual maps with a conventional semantic representation for the railroad line segments in each map edition. This would include objects from a list of common geographic features (points, lines, or polygons), their geolocational information, and their temporal coverage to allow easy analysis and visualization.

1.2. Contribution

The overall contribution of this paper is a fully automatic and unsupervised end-to-end approach for building a contextualized spatio-temporal knowledge graph from a set of vectorized geographic features extracted from topographic historical maps. We tackle the core challenges we mentioned earlier by presenting:

1. An algorithm to identify and partition the original vector data into interlinked geospatial entities (i.e., building-block geometries) that constitute the desired geographic features across map editions (entity generation and interlinking task)
2. A method to identify and retrieve relevant geospatial entities from a publicly available knowledge base (external geo-entity linking task)
3. A semantic model to describe the resulting spatio-temporal data in a structured and semantic output that can be easily interpreted by humans and machines in a form of a knowledge graph that adheres to linked data principles (representation task)

We also present a thorough evaluation for each of the above and apply our method to five relevant datasets that span two types of geographic features: railroads (line-based geometry) and wetlands (polygon-based geometry), each resulting in an integrated knowledge graph. Finally, we make the source code, the original datasets, and the resulting linked data publicly available[2].

1.3. Structure of paper

The rest of the paper is organized as follows. Section 2 presents our proposed approach and methods. Section 3 presents an evaluation of our approach by automatically building a knowledge graph for (i) a series of railroad networks from historical maps covering

[2]https://github.com/usc-isi-i2/linked-maps

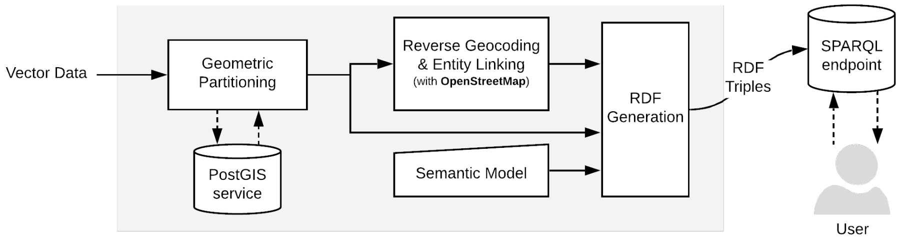


Fig. 3. Pipeline for constructing spatio-temporal linked data from vector data

two different regions from different points in time, and (ii) a series of wetland data from maps covering three different regions from different periods. Section 4 reviews the related work. Section 5 concludes, discusses, and presents future work.

2. Approach to Building Spatio-temporal Knowledge Graphs

2.1. Preliminaries

Before we discuss our proposed method, it is essential to define certain preliminaries and the terminology used in this paper. A geographic feature refers to a collection of geometries (points, lines, or polygons), which together represent an entity or phenomena on Earth. In this paper, a geographic feature is represented using a collection of "building-block" geometries. These building-blocks can be either lines or areas in the case of linear geographic features or polygon geographic features, respectively. A building block can represent a part of a geographic feature shared across maps or unique within a map. For example, in Figure 4a, `A` and `B` are line building-blocks. Each building-block may be decomposed into smaller building-blocks, which may contain continuous or disconnected parts (lines or areas). For example, if a part of `A` and `B` represents the same entity, `A` and `B` are decomposed into three building blocks: `A′`, `B′`, and `AB`. The building block `AB` represents the shared geometry detected from `A` and `B`. `A′` and `B′` represent the unique geometry in their original map. Similarly, each color area in Figure 5 (`A′` in red, `B′` in blue, and `AB` in green) is an area building-block (giving a total of three building-blocks). Each building-block geometry is encoded as well-known text (WKT) representation (`MULTILINE` or `MULTIPOLYGON` textual format) and corresponds to a geospatial entity in our resulting KG.

2.2. Overview of the approach

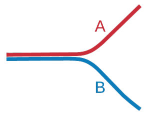
The unsupervised pipeline we propose for the construction of the knowledge graph consists of several major steps as illustrated in Figure 3. These steps can be summarized as follows:

1. Automatically partition the input geographic feature originating from the vector data into building-block geometries (i.e., geospatial entities) using a spatially-enabled database service (e.g., PostGIS) (see Section 2.3).
2. Perform external entity linking by utilizing a reverse-geocoding service to map the geospatial entities to existing instances in an open knowledge base (e.g., OpenStreetMap) (see Section 2.4)
3. Construct the knowledge graph by generating RDF triples following a pre-defined semantic model using the data we generated in the previous steps (see Sections 2.5 and 2.6)

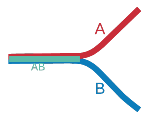
Once the RDF data is deployed, users can easily interact with the building-block geometries (geospatial entities), the geographic features and metadata to perform queries (Section 2.7). These allow end-users to visualize the data and support the development of spatio-temporal downstream applications.

2.3. Generating building-blocks and interlinking

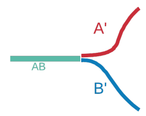
The first task in our pipeline is the generation of building-block geometries (also referred to as "geospatial entities" in the final resulting KG) that can represent the various geographic features (e.g., railroad networks or wetlands) across different maps (same region, different map editions) in a granular and efficient fashion. This task can be classified as a common entity matching/linking and entity "partitioning" task. Given two geometries from two map editions of the same region, we want to identify which parts of those geome-



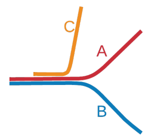
(a) Line segments A and B have common and distinct parts



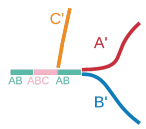
(b) Buffer out and find the common parts



(c) Partition line segments



(d) Line segment C is added



(e) Final line partitioning

Fig. 4. Illustration of the geometry partitioning to building-blocks for a line geometry: spatial buffers are used to identify the same line segments considering potential positional offsets of the data

tries coincide and thus represent the same parts of the feature. This allows us to generate building-block geometries that are more granular and can be used to represent the common and the distinct parts (changes) of the geographic features.

Consider a simplified example consisting of linear features from two map editions (Figure 4a), where line A is from an older map edition and line 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 features, we split each of these lines into several building-block lines based on the intersection of the lines, as shown in Figures 4b and 4c, resulting in building-block lines A', B' and AB. When a third source (another map edition also containing the feature), C, is added, a similar partitioning process is executed as shown in Figures 4d and 4e. Another example is seen in Figure 5, where we show an illustration of a geometry partitioning result for a polygon-based feature (i.e., wetland data). Similar to the line partitioning process described above, the polygon partitioning generates a collection of building-block areas (each block is shown in a different color in Figure 5). Note that similarly to line `AB` in Figure 4e, some building-blocks in Figure 5 contain disconnected geometries, representing the most granular building-blocks needed in order to represent the desired feature.

As we mentioned in Section 2.2, we use a spatially-enabled database service to simplify handling data manipulations of geospatial objects. 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 our task efficiently and enable an incremental addition of map sheets over time, we implemented Algorithm 1. The algorithm performs the partitioning task by employing several PostGIS Application Programming Interface (API) calls over the geometries of our lines or polygons in the database. In the case of line geometries, we buffer each line segment to create two-dimensional areas before applying any geospatial operation described below.

In detail, the procedure works as follows. The **for** loop in line 1 iterates over each of the map editions to extract the feature geometry (as seen in lines 2 and

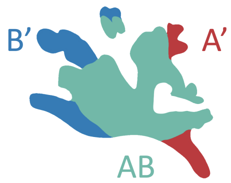


Fig. 5. Illustration of a geometry partitioning result for a polygon geometry: each color represents a different building-block. A single building-block may contain disconnected areas

Algorithm 1: The feature partitioning and interlinking algorithm

Data: a set $\mathcal{M}$ of feature geometries for different map editions of the same region (vector data)
Result: a directed acyclic graph $\mathcal{G}$ of building-block geometries (nodes) and their relations (edges)

```
1  foreach i ∈ M do
2      F_i = set of geometries in i;
3      G.add(i ↦ F_i);
4      L = list of current leaf nodes in G;
5      foreach k ∈ L do
6          F_k = set of geometries in k;
7          F_α = F_i ⋂ F_k;
8          G.add(α ↦ F_α) ; set i, k as direct
             predecessors of α;
9          F_γ = F_k \ F_α;
10         G.add(γ ↦ F_γ) ; set k as direct
             predecessor of γ;
11     F_δ = F_i \ (⋃_{j∈L} F_j);
12     G.add(δ ↦ F_δ) ; set i as direct predecessor
         of δ;
```

stored in $\mathcal{F}_i$) to create the initial "building-block" geometry (line 3, denoted as node i and added to graph $\mathcal{G}$, which eventually will hold our final set of building-blocks and record their relations in a data structure). 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 "leaf" nodes. These are nodes that represent the most fine-grained building-blocks computed so far. A and B in Figure 4a correspond to k and i respectively (in iteration 2 of the algorithm when executed over the data in the Figure 4). For each "leaf" node we execute the following steps:

1. **Geometry intersection.** k's geometry is stored in $\mathcal{F}_k$ (line 6) and then used to compute the matched geometry parts between i and k to generate the geometry $\mathcal{F}_\alpha$ (line 7) and create the new building-block α, a direct successor of nodes i and k (line 8). α (iteration 2) corresponds to AB in Figure 4c.
2. **Geometry difference (local "subtraction").** In line 9, we compute the geometry in k that is not in i, resulting in the geometry $\mathcal{F}_\gamma$ corresponding to the new building-block γ, now a direct successor of k (line 10). γ (iteration 2) corresponds to B' in Figure 4c.

Geometry union-difference (global "subtraction"). Once we finish going over the list of leaves, we compute the unique geometries that exist in the newly added building-block (the last added map edition in $\mathcal{M}$) by subtracting the union of the geometries of the leaf node intersections (with previous processed maps) from the original map block i (as described in line 11), resulting in the geometry $\mathcal{F}_\delta$ corresponding to the new building-block δ, now a direct successor of node i (line 12). δ (iteration 2) corresponds to A' in Figure 4c.

The relations between the nodes in graph $\mathcal{G}$ carry a semantic meaning between the different building-blocks (a node is contained in its predecessors and contains its successors) and will play a critical role in the RDF generation and query mechanism since they represent the relations between the building-blocks across different points in time of the same region.

2.4. reverse-geocoding and geo-entity linking

Volunteered geographic information platforms [23] are used for collaborative mapping activities with users contributing geographic data. OpenStreetMap is one of the most pervasive and representative examples of such a platform and operates with a certain, albeit somewhat implicit, ontological perspective of place and geography more broadly. OSM suggests a hierarchical set of tags[3] that users can choose to attach to its basic data structures to organize their map data. These tags correspond to geographic feature types that we will query (i.e., wetland, railroad, etc.).

Additionally, a growing number of OSM entities are being linked to corresponding Wikipedia articles, Wikidata [24] entities and feature identifiers in the USGS Geographic Names Information Service (GNIS) database. GNIS is the U.S. federal government's authoritative gazetteer. It contains millions of names of geographic features in the United States and Antarctica.

Our proposed method for the enrichment of the generated geospatial entities (i.e., building-block geometries) with an external resource is built upon a simple geo-entity linking mechanism with OSM. This is again a task of entity matching/linking; this time it is with an entity in an external knowledge base.

The method is based on reverse-geocoding, which is the process of mapping the latitude and longitude measures of a point or a bounding box to an address or a geospatial entity. Examples of these services include the GeoNames reverse-geocoding web service[4] and OSM's API.[5] These services support the identification of nearby street addresses, places, areal subdivisions, etc., for a given location.

The geo-entity linking process is depicted in Algorithm 2 and illustrated in Figure 6. We start with individual building-block geometries of known type (T in Algorithm 2). In the case of the data we present later in the evaluation in Section 3, we start with the building-blocks of geometries of **railroads** or **wetlands** (seen in blue in Figure 6), so we know the feature type we are searching for. Each input building-block geometry, s, is an individual node in graph $\mathcal{G}$ from Section 2.3. We first generate a global bounding box for 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 building-block. As a heuristic, we randomly sample a small number of coordinate pairs (`Points`), corresponding to the number of entities composing s (N ranges from 10 to 85 in our datasets, as presented in Section 3.1); thus, we gain more confidence in the detected instances, as seen in

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

Algorithm 2: The geo-entity linking algorithm

Data: building-block geometry s, number of samples N, feature type T
Result: list $\mathcal{L}$ of instances on OpenStreetMap in s

1 B_s = bounding box wrapping s;
2 $\mathcal{L}$ = reverse-geocoding(B_s, T); // returns OpenStreetMap instances of type T in B_s
3 **for** 1...N **do**
4   e = randomly sample a `Point` in s;
5   E = reverse-geocoding(e, T);
6     $\mathcal{L}$.add(E);
7 remove instances with a single appearance in $\mathcal{L}$;
8 return $\mathcal{L}$;

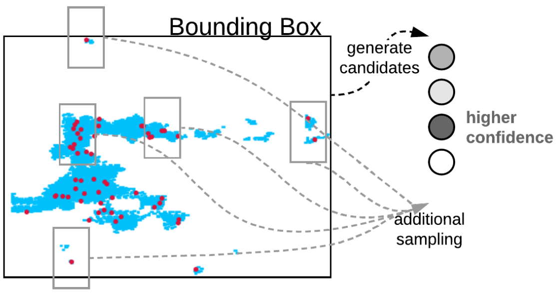


Fig. 6. Our method for acquiring external knowledge base instances

lines 4-6 in Algorithm 2 and in red in Figure 6. Finally, we reduce the list by removing the matching candidates in the external KB that have a single appearance, thus filtering out entities that are not likely part of the enclosed geometry of our geospatial entity. Each one of the resulting instances is used in later stages to enrich the knowledge graph we construct with additional semantics and metadata from the external knowledge base.

Figure 7 shows an example of a scanned topographic map (seen in the background), which we used to extract its corresponding wetland vector data, alongside two OSM instances detected using our geo-entity linking method. The labels in Figure 7 (i.e., `Four Mile Cove Ecological Preserve` and `Six Mile Cypress Slough Preserve`) correspond to the `name` attribute (i.e., "human label") of each entity. These labels are part of a set of attributes we use to augment our resulting data with information that did not exist in the original dataset.

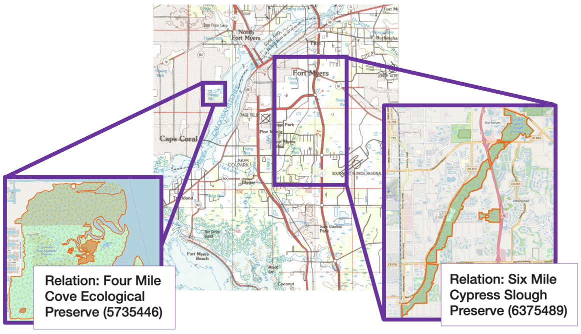


Fig. 7. An example of two OSM instances (enclosed in purple) and their `name` labels detected using our geo-entity linking method over a scanned topographic map (seen in the back)

2.5. *Semantic model*

As a generic model or framework, RDF can be used to publish geographic information. Its strengths include its structural flexibility, particularly suited for rich and varied forms for metadata required for different purposes. However, it has no specific features for encoding geometry, which is central to geographic information. The OGC GeoSPARQL [25] 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. To provide a representation with useful semantic meaning and universal conventions for our resulting data, we define a semantic model that builds on GeoSPARQL.

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

Figure 8 shows the semantic model we describe in this section. 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 building-block extracted from the original vector data.

By aggregating a collection of instances of the class `geo:Feature` with a property of type `geo:sfWithin` we can construct a full representation for the geometry of a specific geographic feature in a given point in time. Similarly, we can denote the decomposition to smaller elements using the property `geo:sfContains`).

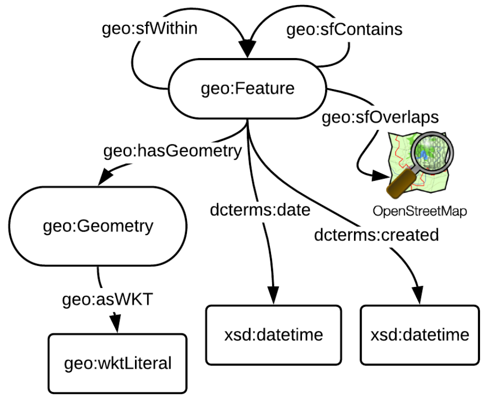


Fig. 8. Our semantic model

The use of these 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 OSM 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 point in time of the building-block, and `dcterms:created`, to denote the time in which this building-block was generated to record provenance data. `dcterms` stands for the Dublin Core Metadata Initiative[6] metadata model, as recommended by the World Wide Web Consortium (W3C).[7]

Complex geometries are not human-readable as they consist of hundreds or thousands of coordinate pairs. Therefore, we use dereferenceable URIs to represent the geospatial entity instead. Using a named node in this capacity means that each entity has its own URI as opposed to the common blank-node approach often used with linked geospatial entities. Each URI is generated using a hash function (the MD5 message-digest algorithm, arbitrarily chosen) on a plain-text concatenation of the feature type, geometry, and temporal extent, thus providing a unique URI given its attributes. Each building-block instance (geospatial entity) 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.

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.

Figure 9 shows how the spatio-temporal data, resulting from the previous steps, is mapped into the semantic model (from Figure 8) to generate the final RDF graph. The first column, titled `gid`, corresponds to the local URI of a specific node (building-block geometry). The columns titled `predecessor_id` and `successor_id` correspond to the local URIs of the nodes composed-of and composing the specified `gid` node, respectively. All the three node entities are of type `geo:Feature`. The data in the `wkt` column contains the geometry WKT representation. It is linked to the building block node via an entity of type `geo:Geometry`, as we described above. The rest of the attributes (`year`, `time_generated`, and `OSM_uri`) are stored as literals, following the semantic model we presented in Figure 8.

2.6. Incremental linked data

Linked Data technologies can effectively maximize the value extracted from open, crowdsourced, and proprietary big data sources. Following the data extraction and acquisition tasks described in Sections 2.3 and 2.4, and the semantic model described in Section 2.5, 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. In order to encourage reuse and application of our data in a manageable manner, we need to make sure that the linked data publication process is robust and maintainable.

This hierarchical structure of our directed acyclic graph $\mathcal{G}$ (introduced in Algorithm 1) and its metadata management allows us to avoid an update across all the existing published geographic vector data (in linked data) and instead handle the computations incrementally once a new representation of the feature from a subsequent map edition is introduced.

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

1. Generating URIs as names for things, without the need to modify any of the previously published

[6]https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[7]https://www.w3.org/

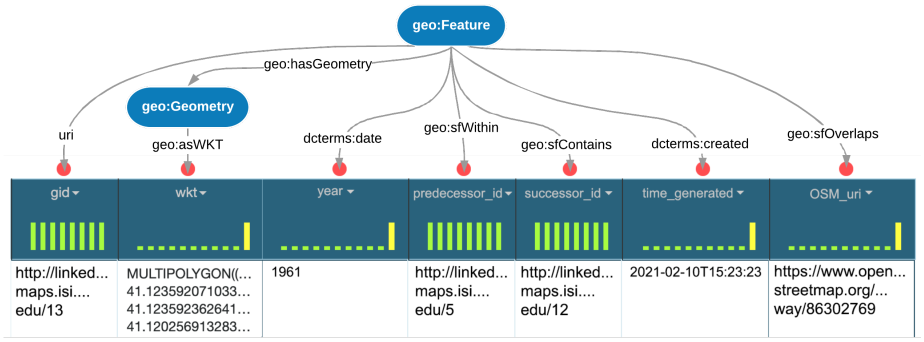


Fig. 9. Mapping of the generated spatio-temporal data into the semantic model

URIs once further vector data from the same region is available and processed
2. Maintaining existing relations (predicates) between instances (additional relations may be added, but they do not break older ones)
3. Generating data as a machine-readable structured data
4. Using standard namespaces and semantics (e.g., GeoSPARQL)
5. Linking to additional resources on the web (i.e., OpenStreetMap)

2.7. *Querying*

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

In order to elaborate on 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 10 shows a query (i.e., the "skeleton" query) that retrieves all the leaf node building-blocks (i.e., the most granular building-blocks). As shown Figure 10, 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 Figures 18, 19, 20, and 21). Line 5 restricts the queried building-blocks (`geo:Features`) to leaf nodes only (in graph $\mathcal{G}$), thus retrieving the most granular building-blocks. This is done by discarding all the nodes that hold a predicate of type `geo:sfContains`, which means that we retrieve only leaf nodes.

This is important due to the incremental nature (and the way graph $\mathcal{G}$ "grows"): as we mentioned previously, every time we add a new representation of the feature from a subsequent map edition, we decompose the existing leaf nodes (most granular building-blocks) to a new layer of leaf blocks (newer, smaller and more granular building-blocks, if subject to decomposition) and its metadata migrates to the lowest level of nodes (new 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") 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. 10. Our SPARQL query "skeleton"

If, for example, we are interested to see the entire geographic feature in a specific point in 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 feature parts from a specific time or finding the feature 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 evaluate and analyze our methods using qualitative and quantitative methods over two types of geographic features: railroads (line-based geometry) and wetlands (polygon-based geometry). In this section,

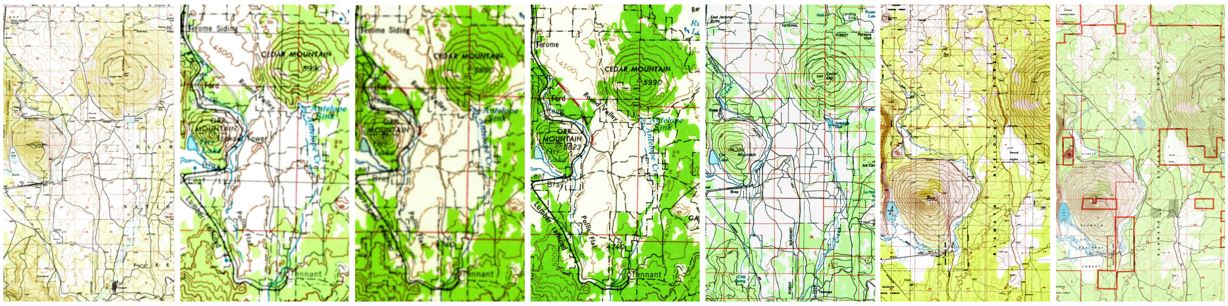


Fig. 11. Historical maps of Bray, California from 1950, 1954, 1958, 1962, 1984, 1988 and 2001 (left to right, respectively)

we present the results, measures, and outcomes of our pipeline when executed on the following datasets:

Railroad data We tested two datasets of vector railroad data (encoded as `MULTILINE`s) extracted from the USGS historical topographic map archive,[8] using the extraction methods of Duan et al. [6]. Each of which covers a different region and is available for different points in time. The railroad data originates from a collection of historical maps for:

1. Bray, California (denoted as **CA**) from the years 1950, 1954, 1958, 1962, 1984, 1988, and 2001 (the original raster maps are shown in Figure 11).
2. Louisville, Colorado (denoted as **CO**) from the years 1942, 1950, 1957 and 1965.

Wetland data We tested three datasets of vector wetland data (encoded as `MULTIPOLYGON`s) that were similarly extracted from the USGS historical topographic map. Again, each of these map sheets covers a different region and spans different points in time. The wetland data originates from a collection of historical maps for:

1. Bieber, California (denoted as **CA**) from the years 1961, 1990, 1993, and 2018 (the original raster maps are shown in Figure 12)..
2. Palm Beach, Florida (denoted as **FL**) from the years 1956, 1987 and 2020.
3. Duncanville, Texas (denoted as **TX**) from the years 1959, 1995 and 2020.

Our primary goal in this section is to show that our proposal provides a complete, robust, tractable, and efficient solution for the production of linked data from vectorized historical maps.

[8] https://viewer.nationalmap.gov, http://historicalmaps.arcgis.com/usgs/

Table 1
Partitioning statistics for **CA railroads**

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

Table 2
Partitioning statistics for **CO railroads**

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

3.1. *Evaluation on the feature partitioning*

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 and feature type (executed on a 16 GB RAM machine @ 2.9 GHz Quad-Core Intel Core i7). The number of vector features in the geographic feature 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 sheet feature geometry (column '`# nodes`') are depicted in Tables 1 (CA) and 2 (CO) for the railroad data, and in Tables 3 (CA), 4 (FL) and 5 (TX) for the wetland data.

As seen in Tables 1, 2, 3, 4 and 5, we observe that for both types of geographic features, the building-block geometries extracted from these maps vary in terms of "quality". That is, they have a different number of vector lines that describe the geographic feature and each

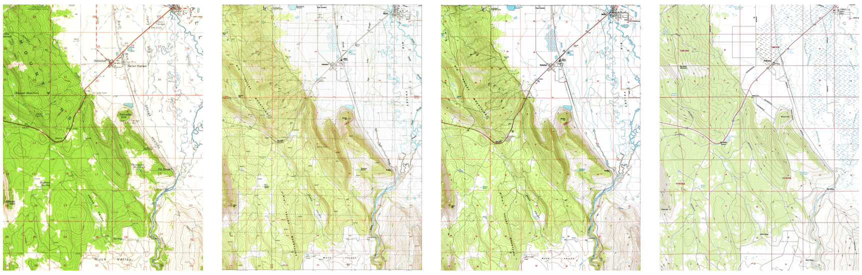


Fig. 12. Historical maps of Bieber, California from 1961, 1990, 1993 and 2018 (left to right, respectively)

one has a different areal coverage (the bounding box area for each feature geometry is reported in Table 6). This is caused by the vector extraction process and is not within 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.

First, we notice that the growth of the number of nodes in graph $\mathcal{G}$ is tractable due to the way the given geographic features change in practice and the fact that each sub-step runs only once when a new set of geometries is inserted from a new map edition. Further, the runtime of each sub-step is also tractable. As expected, the first two map editions (for all areas) generate results within less than a minute for railroads and less than three minutes for wetlands, requiring at most three computations: one geometry intersection between two building-block geometries and two additional subtractions: a local and a global one (as explained in Section 2.3). The following runtimes show that our computation cost is not exponential in practice. By inspecting Tables 1, 2, 3, 4 and 5, we observe that the partitioning runtime depends mostly 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 geospatial data they need to compose. With the addition of new map sheet feature geometries, 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.

Table 3
Partitioning statistics for **CA wetlands**

Year	# vecs	Runtime (s)	# nodes
1961	12	<1	1
1993	17	<1	5
1990	27	6	11
2018	9	6	24

Table 4
Partitioning statistics for **FL wetlands**

Year	# vecs	Runtime (s)	# nodes
1987	184	<1	1
1956	531	180	5
2020	5322	1139	13

Table 5
Partitioning statistics for **TX wetlands**

Year	# vecs	Runtime (s)	# nodes
1959	8	<1	1
1995	6	<1	5
2020	1	1	10

Additionally, by comparing the first three rows in Tables 1 and 4, we notice that the computation time over a polygon geometry is significantly slower than that of a buffer-padded line geometry. This is despite the railroads having a large number of feature vectors. Further, by examining Tables 3, 4 and 5, we notice that a bigger number of feature vectors in the case of a polygon geometry causes a significant increase in processing time. These observations are predictable, as the computation time is expected to grow when dealing with more complex geometries like polygons covering larger areas that may include several interior rings in each enclosed exterior (similar to "holes in a cheese"), compared to the simple buffer-padded line geometries (sort of a "thin rectangle").

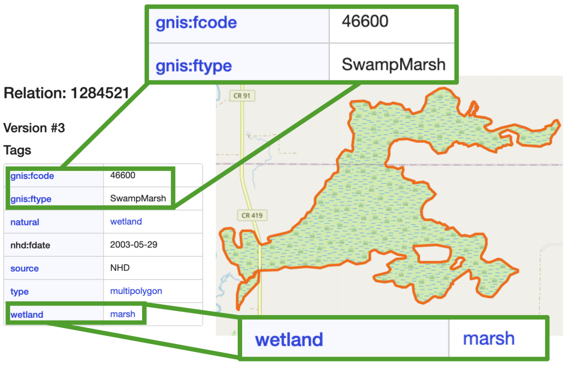


Fig. 13. Screenshot of a wetland instance on *OpenStreetMap* matching an active area corresponding to an instance we generated from the CA wetland data

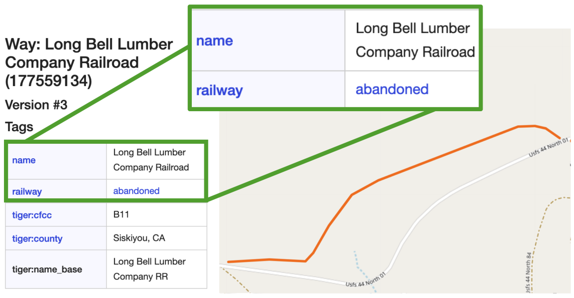


Fig. 14. Screenshot of a railroad instance on *OpenStreetMap* matching an abandoned rail segment corresponding to an instance we generated from the CA railroad data

Table 6

Geo-entity linking results; Area is in square kilometers

		Area	Precision	Recall	F_1
Railroads	CA-baseline	420.39	0.193	1.000	0.323
	CA		0.800	0.750	**0.774**
	CO-baseline	132.01	0.455	1.000	0.625
	CO		0.833	1.000	**0.909**
Wetlands	CA-baseline	224.05	0.556	1.000	0.714
	CA		1.000	1.000	**1.000**
	FL-baseline	27493.98	0.263	1.000	**0.417**
	FL		0.758	0.272	0.400
	TX*	16.62	-	-	-

3.2. *Evaluation on geo-entity linking*

In the process of linking our data to OpenStreetMap, we are interested in the evaluation of the running time and correctness (precision, recall and F_1) of this task.

The running time is linearly dependent on the number of nodes in graph $\mathcal{G}$, the areal coverage of the corresponding geometry, the number of samples using the OpenStreetMap API, and the availability of the API. The API response time averages 3 seconds for each sample. For the railroad data, the execution time for the set of maps from the CA region took approximately an hour (85 nodes) and only a few minutes for CO (10 nodes). This is not surprising as the CA region covers a bigger area and a larger number of nodes. We observe similar behavior in the wetland data. The FL execution time took approximately 2 hours (13 nodes), as it has the largest areal coverage (as seen in Table 6), while the other regions (CA, TX) took approximately 30 minutes to finish. This provides a feasible solution to a process that runs only once for a given set of geometries from different 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 building-block geometry. The measure we present here is in terms of entity (instance) coverage. Precision and recall are calculated according to the labeled (type) instances that are available on OpenStreetMap and make up the inspected geographic feature (i.e., railroad or wetland). Figures 13 and 14 show an example of such instances, with their corresponding tags in OSM's graphic interface. Figure 13 shows a wetland instance on OSM marked as a `SwampMarsh` (with its corresponding GNIS code) and matching an active wetland area in our data (a common building-block from all the editions of the CA wetland data). Figure 14 shows a railroad instance on OSM marked as `abandoned` and matching an abandoned railroad segment in our data (a unique building-block from the 1950 edition of the CA railroad data). This shows our ability to enrich and link our graph to external resources on the web. Moreover, it is worth mentioning that in terms of detecting which railroad or wetland it is ("human label"), we are able to achieve 100% accuracy by taking a majority vote (in cases where such label is present on OSM, as seen in Figure 14, as some instances do not have such designated "human label", as seen in Figure 13).

We have set up a baseline for comparison with our geo-entity 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 removal steps we have described in Section 2.4.

The precision, recall, and F_1 scores of each method over each dataset are shown in Table 6. For each geographic feature and each region, we report the baseline result and our method's result. We also present the

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

Fig. 15. Query similar feature geometries in both 1962 and 2001

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

Fig. 16. Query feature geometries present in 1962 but not in 2001

bounding box area for each dataset (in square kilometers), as it is an important factor in the geo-entity linking evaluation. The bigger the area, the more sampling points we require. The first 4 rows correspond to the railroad data (in CA and CO). The rest corresponds to the wetland data. Note, that results for the TX wetlands have been omitted in this part of the evaluation due to a complete absence of labeled data in OSM covering that area. We will briefly discuss this in Section 5.

Due to the varying geometries, areal coverage, and available data in the external knowledge base for each region, and as expected, our measure shows different scores for each dataset. In 3 out of 4 datasets, our method achieves much higher F_1 scores than the baseline (0.774 and 0.909 compared to 0.323 and 0.625 respectively in the railroad data; and 1.000 compared to 0.714 in the CA wetland data) and achieves an acceptable score for this task. In the FL wetland dataset, we achieve lower F_1 scores for both methods (baseline and ours). This is not surprising as the area of coverage is significantly bigger than in all other datasets, requiring us to generate a bigger number of samples in order to capture all the relevant instances on OSM. We will expand on this issue in Section 5. Nonetheless, further examination of the FL wetland results shows that the low F_1 score of the baseline is due to the fact that it only considers the global bounding box (thus the high recall, but low precision). On the other hand, our method achieves a higher precision score but a much lower recall, compared to the baseline. This is a crucial point in geographic applications, as many systems consider the precision to be more important than recall due to a low false-positive tolerance [26].

3.3. Evaluation on querying the resulting data

We execute several query examples over the knowledge graph we constructed in order to measure our

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

Fig. 17. Query unique feature geometries from 1958

model in terms of query time, validity, and effectiveness. For the generated railroad data, we had a total of 914 triples for the CA dataset and 96 triples for the CO dataset. For the wetland data, we had a total of 270 triples for the CA dataset, 149 for the FL dataset, and 85 for the TX dataset.

The generated RDF triples would be appropriate to use with any Triplestore. We hosted our triples in Apache Jena.[9] Jena is relatively lightweight, easy to use, and provides a programmatic environment.

Table 7 shows the query-time performance results (average, minimum and maximum). In the first type of query we want to identify the feature parts that remain unchanged in two different map editions (different time periods) for each region (e.g., Figure 15). Each row with a label starting with `SIM-` in Table 7 corresponds to this type of query (the label suffix determines the tested region). We executed a hundred identical queries for each feature type and each area across different points in time 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 feature that were removed or abandoned between two different map editions for each region (i.e., Figure 16). Each row with a label starting with `DIFF-` in Table 7 corresponds to this type of query.

The third type of query retrieves the parts of the feature that are unique to a specific edition of the map (i.e., Figure 17). Each row with a label starting with `UNIQ-` in Table 7 corresponds to this type of query.

Looking at Table 7, we notice that the average query times are all in the range of 10-48(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. The query time results corresponding to the wetland data are slightly slower, but not significantly, comparing to the railroad data. This may be explained by the longer literal encoding of the WKT geometry for polygons, thus slower retrieval time comparing the the line encoding.

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

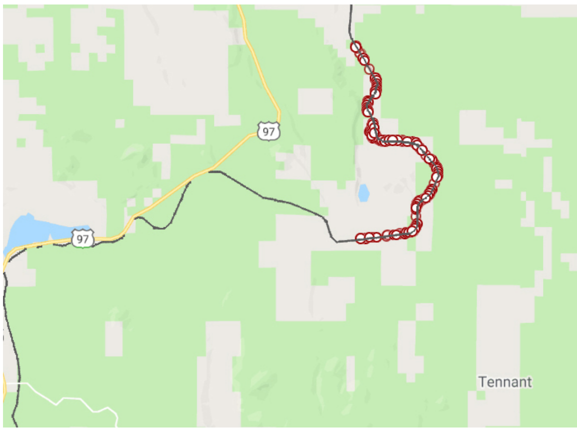


Fig. 18. Example of railroad system changes over time: the parts of the railroad that are similar in 1962 and 2001, marked in red

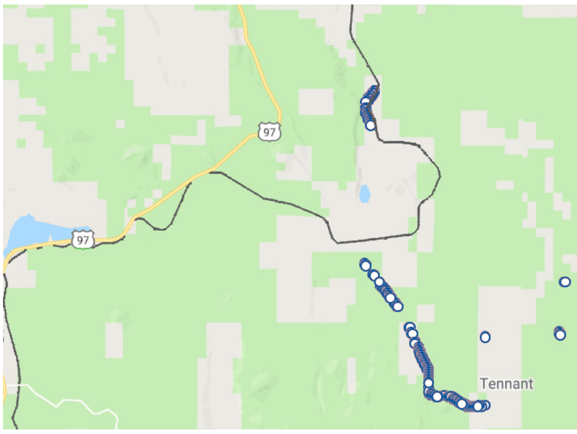


Fig. 19. Example of railroad system changes over time: the parts of the railroad that are present in 1962 but are not present in 2001, marked in blue

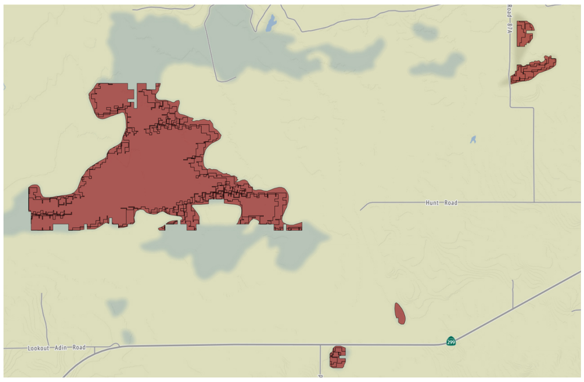


Fig. 20. Example of wetland changes over time: the parts of the Big Swamp in Bieber (CA) that are similar in 1961 and 2018, marked in red

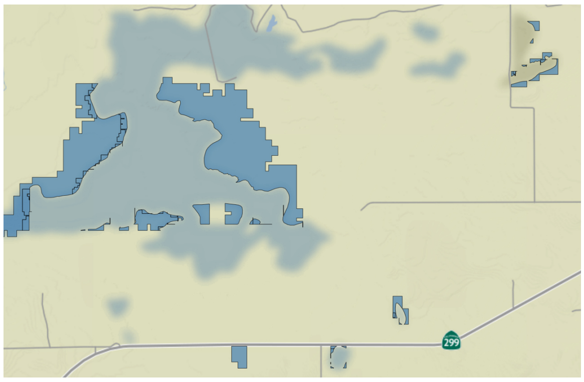


Fig. 21. Example of wetland changes over time: the parts of the Big Swamp in Bieber (CA) that are present in 1961 but are not present in 2018, marked in dark blue, emphasizing its decline throughout time

In order to evaluate the validity of our graph we observe the visualized results of the query presented in Figure 15 when executed over the CA railroad data, which are shown in Figure 18. The figure shows in red the unchanged building-block geometries between the years 1962 and 2001. We notice that the geometries we retrieve qualitatively match what we observe in the original vector data (the line marked in black over the maps in Figures 18 and 19 represents the current railway, which has not changed since 2001). The results of the query presented in Figure 16 are shown in Figure 19, again, when executed over the CA railroad data. Figure 19 shows in blue the parts of the railroad that were abandoned between 1962 to 2001. Comparably, we perform a similar type of query and visualization for the CA wetland data, between the years 1961 and 2018. Figure 20 shows the similar parts in both editions (in red). Figure 21 shows the parts of the wetland (swamp) that were present in 1961 but are not present in 2018 (in dark blue). Again, this matches our qualitative evaluation based on the original vector files (the light blue marks that are part of the map's background depict the current swamp, thus validating our results qualitatively).

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 effective and contextualized open KGs and linked data from historical and contemporary geospatial data.

Table 7
Query time statistics (in milliseconds)

		avg	min	max
Railroads	SIM-CA	12	10	18
	SIM-CO	11	9	20
	DIFF-CA	10	8	20
	DIFF-CO	10	9	14
	UNIQ-CA	14	8	28
	UNIQ-CO	15	9	17
Wetlands	SIM-CA	22	18	34
	SIM-FL	35	18	55
	SIM-TX	21	12	44
	DIFF-CA	25	16	43
	DIFF-FL	32	18	60
	DIFF-TX	21	11	30
	UNIQ-CA	24	18	44
	UNIQ-FL	48	38	73
	UNIQ-TX	14	12	40

4. Related Work

Much work has been done on mapping geospatial data into RDF graphs. Kyzirakos et al. [13] developed a semi-automated tool for transforming geospatial data from their original formats into RDF using R2RML mapping. Usery et al. [14] presented a method for converting point and other vector data types to RDF for supporting queries and analyses of geographic data. The transformation process presented in these papers does not address linking the data across multiple sources or linking the source data with additional knowledge bases on the semantic web as described in this paper.

Annotating geospatial data with external data on the web is used for contextualization and the retrieval of relevant information that cannot be found in the source data. This line of research has been addressed in different studies. Vaisman et al. [15] studied the problem of capturing spatio-temporal data from different data sources, integrating these data and storing them in a geospatial RDF data store. Eventually, these data were enriched with external data from LinkedGeoData [20], GeoNames [19], and DBpedia [27]. Smeros et al. [28] focus on the problem of finding semantically related entities lying in different knowledge bases. According to them, most approaches on geo-entity linking focus on the search for equivalence between entities (same labels, same names, or same types), leaving other types of relationships (e.g. spatial, topological, or temporal relations) unexploited. They propose to use spatio-temporal links to improve the process. However, both of these papers do not address linking unlabeled geospatial entities (containing only geometry data, without any additional attributes), as we do in this paper.

Current work on geospatial change analysis spans the construction of geospatial semantic graphs to enable easier search, monitoring and information retrieval mechanisms. Perez et al. [29] computed vegetation indexes from satellite image processing and exposed as the data as RDF triples using GeoSPARQL [25]. The changes in these indexes are used to support forest monitoring. Similar to that approach, Kauppinen et al. [30] collected statistical data from open data sources to monitor the deforestation of the Amazon rainforest by building temporal data series translated into RDF. Kyzirakos et al. [17] used open knowledge bases to identify hot spots threatened by wildfire. However, this line of work does not address an incremental process of geospatial change over time. In this paper, we incorporate a temporal dimension to the geospatial semantic graphs and present an unsupervised pipeline for an automatic incremental geographic feature analysis over time.

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 automatic unsupervised approach to integrate, relate and interlink geospatial data from digitized resources and publish it as semantic-rich, structured linked spatial data in a form of a knowledge graph that follows the Linked Open Data principles.

The evaluation we presented in Section 3 shows that our approach is feasible and effective in terms of processing time, completeness and robustness. The partitioning process runs only once for newly added resources, 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 geometry. 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, due to an increased number of computations, but would still be tractable with respect to the changes happening in the map geometries. As we mentioned in Section 2.3, the breakdown of the building-block geometries 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 have observed in Section 3.1.

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 the geometries to smaller building-block geometries in the case of line geometry. We currently set this parameter manually but we believe such parameter can be learned from the data or estimated using some heuristics with respect to additional attributes such as the area of the geometric object and the quality of the original vector data extraction.

As we have observed in the geo-entity linking evaluation for the wetland data in TX, in Section 3.2, it is not always possible to provide complete solutions to some problems. Given the large volume, and openness of the OSM schema, there are peculiarities about geographic information that present particular challenges with respect to the semantic web. One challenge is the vagueness that exists in geographic categories. For example, what makes a wetland a wetland and not mud? Again, it is worth emphasizing that these peculiarities are not unique to geography. OSM is open to non-expert users of geographic data and thus, the tagging attitude is rather intuitive than based on scientific methodologies and knowledge. We believe that these types of problems can be solved by the downstream application by adjusting the pipeline according to the user's needs.

Moreover, and as we have seen in Section 3.2, performing geo-entity linking over very large areas (such as the case of the FL wetland data), can cause poor performance. In order to achieve better recall, we need to increase the number of the reverse-geocoding calls and perform adjustable bulk query calls instead of an unbounded number of single API calls (which we avoided due to the policies in using the OSM API). An additional solution for this issue can come in the form of a caching mechanism and by using multiple machines for faster parallel access.

We are currently looking into expanding the ability to utilize additional knowledge bases such as Yago2Geo [31]. This knowledge base gathers geospatial entities with their spatial data from different sources (including OSM). 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 Algorithm 1 that are independent of each other and can be executed in parallel in the same iteration over a single map edition. This will enable a faster processing time and strengthen the effectiveness of our solution.

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) and Nvidia Corporation.

References

[1] Y. Chiang, S. Leyk and C.A. Knoblock, A survey of digital map processing techniques, *ACM Computing Surveys (CSUR)* **47**(1) (2014), 1–44.
[2] J.H. Uhl, S. Leyk, Y. Chiang, W. Duan and C.A. Knoblock, Automated extraction of human settlement patterns from historical topographic map series using weakly supervised convolutional neural networks, *IEEE Access* (2019).
[3] J.H. Uhl and W. Duan, Automating information extraction from large historical topographic map archives: New opportunities and challenges, in: *Werner, M. and Chiang, Y.-Y. (Eds.): Handbook of Big Geospatial Data*, Springer, 2021. doi:10.1007/978-3-030-55462-0.
[4] Y. Chiang, W. Duan, S. Leyk, J.H. Uhl and C.A. Knoblock, *Using historical maps in scientific studies: Applications, challenges, and best practices*, Springer, 2020.
[5] S. Leyk, R. Boesch and R. Weibel, A conceptual framework for uncertainty investigation in map-based land cover change modelling, *Transactions in GIS* **9**(3) (2005), 291–322.
[6] W. Duan, Y. Chiang, C.A. Knoblock, S. Leyk and J.H. Uhl, Automatic generation of precisely delineated geographic features from georeferenced historical maps using deep learning, in: *Proceedings of the AutoCarto*, 2018.
[7] N.I. Maduekwe, A GIS-Based Methodology for Extracting Historical Land Cover Data from Topographical Maps: Illustration with the Nigerian Topographical Map Series, *KN-Journal of Cartography and Geographic Information* (2021), 1–16.
[8] S. Athanasiou, D. Hladky, G. Giannopoulos, R.G. Rojas and J. Lehmann, GeoKnow: Making the web an exploratory place for geospatial knowledge, *ERCIM News* **96**(12–13) (2014), 119–120.
[9] C. Bone, A. Ager, K. Bunzel and L. Tierney, A geospatial search engine for discovering multi-format geospatial data across the web, *International Journal of Digital Earth* **9**(1) (2016), 47–62.

[10] S. Athanasiou, G. Giannopoulos, D. Graux, N. Karagiannakis, J. Lehmann, A.N. Ngomo, K. Patroumpas, M.A. Sherif and D. Skoutas.

[11] M. Alirezaie, M. Längkvist, M. Sioutis and A. Loutfi, Semantic referee: A neural-symbolic framework for enhancing geospatial semantic segmentation, *Semantic Web* **10**(5) (2019), 863–880.

[12] C. Bernard, C. Plumejeaud-Perreau, M. Villanova-Oliver, J. Gensel and H. Dao, 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*, ACM, 2018, pp. 456–459.

[13] K. Kyzirakos, I. Vlachopoulos, D. Savva, S. Manegold and M. Koubarakis, 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*, 2014, pp. 33–44.

[14] E.L. Usery and D. Varanka, Design and development of linked data from the national map, *Semantic Web* **3**(4) (2012), 371–384.

[15] A. Vaisman and K. Chentout, Mapping spatiotemporal data to RDF: A SPARQL endpoint for brussels, *ISPRS International Journal of Geo-Information* **8**(8) (2019), 353.

[16] K.R. Kurte and S.S. Durbha, Spatio-Temporal Ontology for Change Analysis of Flood Affected Areas Using Remote Sensing Images., in: *JOWO@ FOIS*, 2016.

[17] K. Kyzirakos, M. Karpathiotakis, G. Garbis, C. Nikolaou, K. Bereta, I. Papoutsis, T. Herekakis, D. Michail, M. Koubarakis and C. Kontoes, Wildfire monitoring using satellite images, ontologies and linked geospatial data, *Journal of web semantics* **24** (2014), 18–26.

[18] M. Haklay and P. Weber, Openstreetmap: User-generated street maps, *IEEE Pervasive computing* **7**(4) (2008), 12–18.

[19] M. Wick and B. Vatant, The geonames geographical database, *Available from World Wide Web: http://geonames. org* (2012).

[20] S. Auer, J. Lehmann and S. Hellmann, Linkedgeodata: Adding a spatial dimension to the web of data, in: *International Semantic Web Conference*, Springer, 2009.

[21] B. Shbita, C.A. Knoblock, W. Duan, Y. Chiang, J.H. Uhl and S. Leyk, Building Linked Spatio-Temporal Data from Vectorized Historical Maps, in: *European Semantic Web Conference*, Springer, 2020, pp. 409–426.

[22] C. Lin, H. Su, C.A. Knoblock, Y. Chiang, W. Duan, S. Leyk and J.H. Uhl, Building Linked Data from Historical Maps., in: *SemSci@ ISWC*, 2018, pp. 59–67.

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

[24] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85.

[25] R. Battle and D. Kolas, Geosparql: enabling a geospatial semantic web, *Semantic Web Journal* **3**(4) (2011), 355–370.

[26] G. Nagy and S. Wagle, Geographic data processing, *ACM Computing Surveys (CSUR)* **11**(2) (1979), 139–181.

[27] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, Dbpedia: A nucleus for a web of open data, in: *The semantic web*, Springer, 2007, pp. 722–735.

[28] P. Smeros and M. Koubarakis, Discovering spatial and temporal links among RDF data, in: *LDOW@ WWW*, 2016.

[29] A.J. Pérez-Luque, R. Pérez-Pérez, F.V. Bonet-García and P.J. Magaña, An ontological system based on MODIS images to assess ecosystem functioning of Natura 2000 habitats: A case study for Quercus pyrenaica forests, *International Journal of Applied Earth Observation and Geoinformation* **37** (2015), 142–151.

[30] T. Kauppinen, G.M. de Espindola, J. Jones, A. Sánchez, B. Gräler and T. Bartoschek, Linked brazilian amazon rainforest data, *Semantic Web* **5**(2) (2014), 151–155.

[31] N. Karalis, G. Mandilaras and M. Koubarakis, Extending the YAGO2 knowledge graph with precise geospatial knowledge, in: *International Semantic Web Conference*, Springer, 2019, pp. 181–197.