A GEOSPATIAL KNOWLEDGE GRAPH PROTOTYPE FOR NATIONAL TOPOGRAPHIC MAPPING

D. E. Varanka

U.S. Geological Survey, Rolla, Missouri, USA – dvaranka@usgs.gov

Commission IV, WG IV/4

KEY WORDS: National Topographic Mapping, Geospatial Knowledge Graph, GeoSPARQL, Linked Open Data, Map Portrayal

ABSTRACT:

Knowledge graphs are a form of database representation and handling that show the potential to better meet the challenges of data interoperability, semi-automated information reasoning, and information retrieval. Geospatial knowledge graphs (GKG) have at their core specialized forms of applied ontology that provide coherent spatial context to a domain of information including non-spatial attributes. This paper discusses research toward the development of a prototype GKG based on national topographic databases of geospatial feature instances, attributes, properties, metadata, and annotations. The challenges are to capture and represent geographic semantics inherent in the source data, to align such graph models with standards where possible, to test logical computations, and to visualize the data using a cartographic user interface. Data integration from outside sources was tested through SPARQL and GeoSPARQL queries. Called the MapKB, the approaches applied in this prototype use a number of software components to build a system architecture aligned with those objectives and are composed entirely of free and open-source software. The system and ontology design were validated through reasoning and competency questions. Technical aspects of the prototype software succeeded, but customization was found to be needed for user-based design.

1. INTRODUCTION

Spatial data infrastructures such as national topographic mapping prioritize data interoperability to serve their diverse communities. Geospatial knowledge graphs (GKG) are a form of database representation and handling that aim to meet the challenges of data interoperability, reasoning for faster knowledge creation, and user access that provides coherent spatial context to domains of information. The research aims to show interoperability that results from formalizing conceptual contexts balances the expanded semantic specifics from users and the complexity of multiple data formats. These are tested in methodological representation and manipulation capabilities of technical prototype software called the Map as Knowledge Base (MapKB).

Though broadly defined, topography as a subject consists of the widely recognized natural and human features of the landscape. Human perception and the biology of the brain supports fields and objects as broad categories of geographic information, these are mirrored in broad categories of geographic information system (GIS) vector and raster data and their semantic logic and assumptions. Data transformation from GIS to knowledge graphs draws on a sphere of shared conceptual knowledge for semantic interoperability. As a result, topographic map representation is widely used as base data consisting of cognitive and logical primitives, logic of geospatial field and object models, to be articulated and integrated with thematic representations. Semantic technology design and application draw on this foundational level of broader knowledge for data information interoperability. Base data consists of cognitive and logical primitives yet aligns with aspects of knowledge such as the ontology of force dynamics. Those broader relations support alignment with physical principles of scientific modelling.

This paper discusses the development of a prototype GKG used to test interoperability aspects of ontology creation, semantic and visualization approaches to information search and retrieval using GeoSPARQL (Open Geospatial Consortium, 2022). The challenges are to capture and represent geographic semantics inherent in the source data, to integrate data from outside sources through SPARQL Protocol and RDF Query Language (SPARQL) queries and to visualize the data using a cartographic user interface. The focus of the prototype system is semantic representation and does not address important questions for efficient storage and retrieval of very large databases. The MapKB system is publicly available as a set of Docker Containers, stand-alone, executable software packages (Bourquin, 2021).

2. APPROACH

2.1 Workflow

MapKB approaches use software components to build a system architecture aligned with available standardized vocabularies and is composed entirely of free and open-source software for geospatial data. The application was created in the context of The National Map of the U.S. Geological Survey (USGS). A workflow was created which leverages multiple open-source software (Figure 1).

The GKG ontology was semi-automatically transformed from source databases through the application of rules on schema attribute, domain, and metadata files to create classes, properties, and other triple resources of Resource Description Framework (RDF) and Web Ontology Language (OWL) using the open-source tool Protégé (Wagner and Varanka, 2020; Hayes and Patel-Schneider, 2014; Hitzler et al., 2012; Musen, 2015). OWL ontology triple resources are comparable to the table and key structure for a GIS relational database. Two predominant subclasses, topo:Feature and topo:Attribute, refers to the dynamic tables in the database that contain instance data in the former, and to data belonging to static tables in the database containing feature names, codes, resolution types, and more. These data are referenced across multiple GIS layer datasets so redundant information regarding each instance of these classes is removed and replaced with object properties, to reduce data storage requirements and use for inference. The ontology was implemented on the following subset of geometric GIS layers from The National Map: Boundaries, GNIS, Hydrography, Structures, and Transportation and limited to the general region of Washington, D.C., USA.

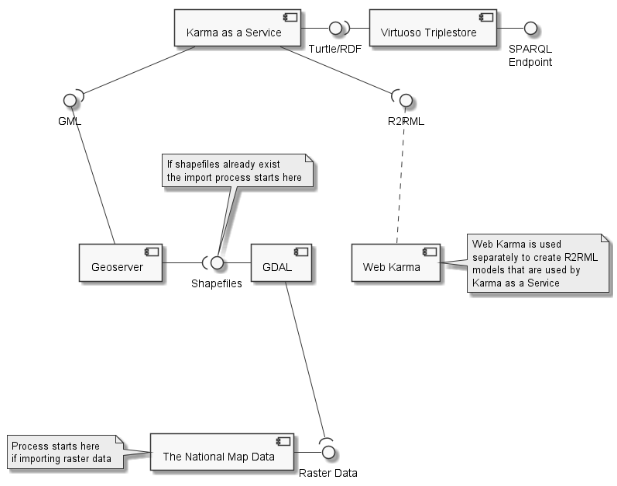

Karma as a Service
Turtle/RDF
Virtuoso Triplestore
SPARQL Endpoint
GML
R2RML
If shapefiles already exist the import process starts here
Geoserver
Shapefiles
GDAL
Web Karma
Web Karma is used separately to create R2RML models that are used by Karma as a Service
Process starts here if importing raster data
The National Map Data
Raster Data

Figure 1. Workflow diagram of populating a triplestore for querying data from The National Map of the USGS. Shaded boxes indicate optional steps and bolded boxes indicate user-performed steps.

The feature instance and related databases being converted to RDF were downloaded from The National Map or converted to ESRI Shapefile format using the GDAL tool and uploaded to a Geoserver instance (U.S. Geological Survey, 2022; Open Source Geospatial Foundation, 2019a; Open Source Geospatial Foundation, 2019b). Geoserver conforms to the Web Feature Service (WFS) Interface Standard to export GML data and can deliver data via a REST interface (OGC, 2019; W3C, 2012). Karma-As-A-Service directly imports data from Geoserver via Geoserver's REST interface to be run as an RDF Generation Representational State Transfer (REST) Service to batch convert data. The Web-Karma interface allows users to map GML data to an OWL ontology by generating a R2RML mapping file for the data to produce stored RDF (University of Southern California, 2016; Das et al., 2012). Once the R2RML file is generated by Web-Karma, associated Python functions were built. The created conversion model location is passed to Karma-as-a-Service to convert subsequent GML data to RDF and to implement custom Python functions. Functions were implemented for namespaces and feature geometries. The functions used to create URIs for entities within-dataset for the purpose of linking via object properties consist entirely of appending the data from the fields to the namespace for the data. The functions used to create the full data for geometries build the GML and Well-Known Text (WKT) representation out of the data served from Geoserver (Lott, 2015). WKT representations of geometries are not given from Geoserver so functions were created that use the coordinate strings from the GML representation to generate the WKT representation.

Ontology required post-processing to better align with RDF/OWL properties. The rdfs:subPropertyOf relationship was leveraged for inference. Data from TNM can be matched with the Open Geospatial Consortium GeoSPARQL v. 1 ontology. The topo:Feature class the geosparql:Feature class are approximately equivalent. The geometry field of unique geospatial data for all tables is converted to the class geosparql:Geometry. This equivalence relationship allows the ontology to leverage the feature geometries and the set of geometry properties. OWL ontologies created with metadata benefit from semantic information that define them and their entities. Multiple metadata properties provide sufficient context for alignment. Annotation properties created from metadata provide one approach for linking ontologies.

The converted triples and ontology are imported into OpenLink Virtuoso to be used as a triplestore and SPARQL endpoint to store, manage, and access the data to respond to queries by a user, a program, or other actor (OpenLink Software, 2021). The MapKB client user interface would consume data from the triplestore or could query the Virtuoso SPARQL endpoint directly. Once the SPARQL query is sent, the client connects through the SPARQL endpoint or through a provided Virtuoso connector to either query USGS dataset triplestores or link with LOD cloud data sources via the user interface. SPARQL can be used to update the triplestore. MapKB downloads and compiles the development version of Virtuoso when it is deployed, so although the software is contained within the container, it is not shipped as part of the image.

2.2 Ontology Pattern

An OWL ontology pattern (OP) based on the source data demonstrated inference to improve access to the data semantics. The OP inferred interchangeable feature types, codes, and geometries that align with the basic GeoSPARQL ontology. Real-world concepts of geospatial feature types are organized by class identification and description in the source GIS data as look-up tables called FTypes and FCodes. The look-up table approach requires users to manually find and read natural language descriptions that contain details about topographic feature types from separate metadata documents. Another primary function of these codes is for assigning GIS geometry types to feature types; feature classes as defined by the GIS software of the source data are generally geometry types, not real-world concepts. The point and line geometries must be separated from the feature type codes for the Open Geospatial Consortium (OGC) GeoSPARQL ontology and, the natural language semantics must be formalized for ontology subgraphs.

The OP enables the following rules:

1. All feature instances have a feature type classification based on the class definition
2. All feature classes and their members have an FCode and/or FType
3. All feature type classes and their codes are associated with one or more geometry types
4. All geometry instances are members of a geometry Feature Class

These rules are supported first by the asserted then by inferred triples below.

1. Feature type classes for data from The National Map use the Description label and prefix topo for the OP. These classes form subclasses under geosparql:Feature.

2. Feature classes have an object property called topo:fCode for the 5-digit code and an FCode; specific FCodes are modeled as classes. The properties topo:fCode, topo:fType and topo:subType are all members of the owl:SymmetricProperty class. FType is an Anonymous Ancestor, as named in Protégé, meaning an inferred subclass.
3. Geosparql:Geometry and topo:FeatureClass are essentially subclasses of each other (an OWL technique for equivalency constrained as necessary, but not sufficient). In this OP, a topo:subType (to more similarly match the source data) between geosparql:Geometry and topo:FeatureClass. This property asserts these classes are inverses.
4. Geometry instances are members of OGC Simple Features standard, included in the GeoSPARQL ontology.

Upon clicking on a geosparql:Feature subclass, the following inferred triples are produced.

1. fType only 73002 (the specific FType code)
2. hasGeometry some Geometry
3. name some Name
4. objectID exactly 1 ObjectID

The OP based on the triplestore graphs demonstrated the correct functioning of those segments of the core MapKB ontology. The next stage of the study tested the ontology as it enabled SPARQL queries using the MapKB user interface (UI).

2.3 MapKB User Interface

A cartographic UI was created using Leaflet, an open-source JavaScript library for interactive maps for the visualization and interaction of users with the triplestore graphs and SPARQL endpoint. (Agafonkin, 2022). The general guidelines given by the information search process model (Kuhlthau, 2004) serves to guide anticipated exploration and functionality needs. Stages of the model a user's holistic seeking experience involving affect, cognition, and physical actions to the experience. The user may begin by selecting namespace labels by menu search options for typically retrieving initial results (Figure 2). Multiple graphs can be visualized at once. After evaluation of the results, further optional actions can follow.

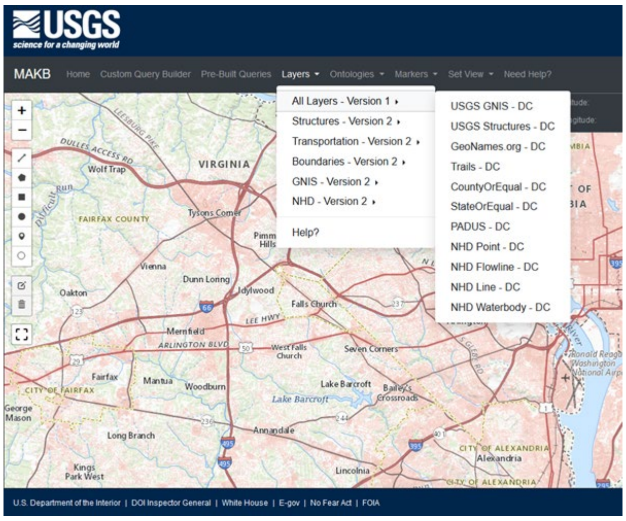
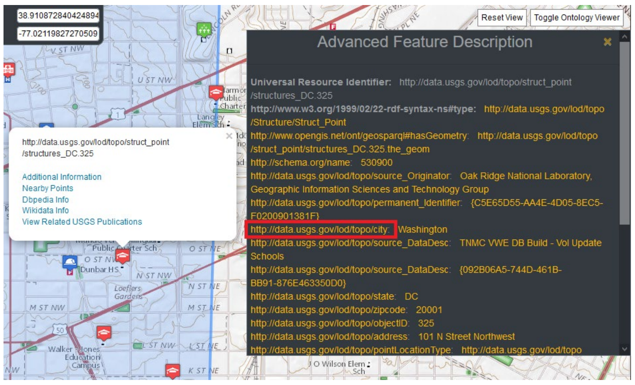


Figure 2. Initial list of available datasets

Other queries were performed on the initial results appearing on a map or table. The concept map for the features retrieved from source GIS layers was toggled back and forth from the viewer for information clarification. The interface responds to SPARQL and GeoSPARQL query language implemented to visualize the contents of the data graphs. An advanced feature description function retrieves related properties to support browsable graph search. The RDF triples associated with each entity can be found by clicking on the entity or on a link called 'Additional Information' The initial available properties for the selected features are listed in a drop-down menu and linked to the next set of related triples (Figure 3).



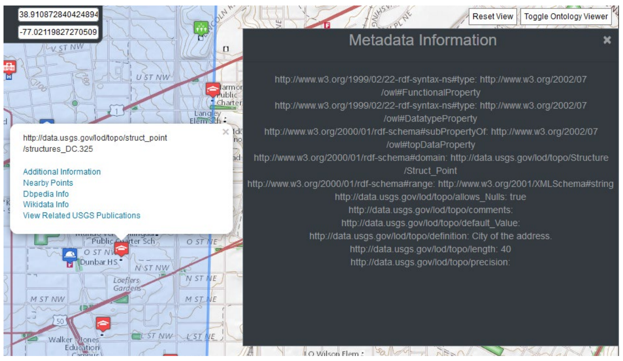
Advanced Feature Description

http://data.usgs.gov/lod/topo/struct_point
/structures_DC.325

Universal Resource Identifier: http://data.usgs.gov/lod/topo/struct_point /structures_DC.325
http://www.w3.org/1999/02/22-rdf-syntax-ns#type: http://data.usgs.gov/lod/topo /Structure/Struct_Point
http://www.opengis.net/ont/geosparql#hasGeometry: http://data.usgs.gov/lod/topo /struct_point/structures_DC.325.the_geom
http://schema.org/name: 530900
http://data.usgs.gov/lod/topo/source_Originator: Oak Ridge National Laboratory, Geographic Information Sciences and Technology Group
http://data.usgs.gov/lod/topo/permanent_Identifier: {C5E65D55-AA4E-4D05-8EC5-F0200901381F}
http://data.usgs.gov/lod/topo/city: Washington
http://data.usgs.gov/lod/topo/source_DataDesc: TNMC VWE DB Build - Vol Update Schools
http://data.usgs.gov/lod/topo/source_DataDesc: {092B06A5-744D-461B-BB91-876E463350D0}
http://data.usgs.gov/lod/topo/state: DC
http://data.usgs.gov/lod/topo/zipcode: 20001
http://data.usgs.gov/lod/topo/objectID: 325
http://data.usgs.gov/lod/topo/address: 101 N Street Northwest
http://data.usgs.gov/lod/topo/pointLocationType: http://data.usgs.gov/lod/topo

Figure 3. Properties available for selected feature-level data by browsing the graph. Links to additional information from Linked Open Data (LOD) and related USGS publications are shown on the left.

Geospatial metadata can be difficult to find because not all are combined with the data, may reside in different databases, or are constrained by the collecting agency's standards. Thus, across the different published products, there were a variety of attributes with similar names but different associated metadata. Within the GKG, annotation properties are akin to the metadata within a relational database; they describe the restrictions placed on attributes and allow users to understand the contents of that attribute. Such properties are critical for automated GKG information integration and alignment. The machine could use annotation properties for data properties to draw a comparison between two or more instances. The method applied for this study for a machine to accurately analyze datatype and object properties to align different graphs was to convert metadata to annotation properties for the ontology using predominant sources, such as table data models, data dictionary reports, and rules for the creation of data that cross-referenced resources variously formatted as text, UML, and XML files (Figure 4).



Metadata Information

http://data.usgs.gov/lod/topo/struct_point
/structures_DC.325

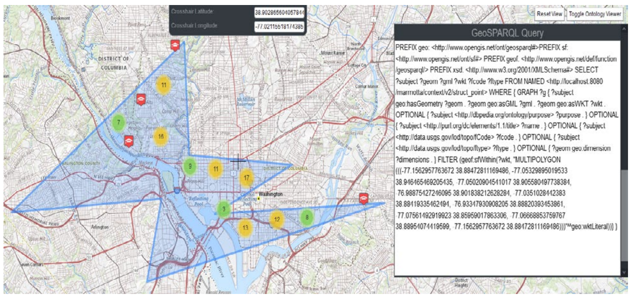
http://www.w3.org/1999/02/22-rdf-syntax-ns#type: http://www.w3.org/2002/07 /owl#FunctionalProperty
http://www.w3.org/1999/02/22-rdf-syntax-ns#type: http://www.w3.org/2002/07 /owl#DatatypeProperty
http://www.w3.org/2000/01/rdf-schema#subPropertyOf: http://www.w3.org/2002/07 /owl#topDataProperty
http://www.w3.org/2000/01/rdf-schema#domain: http://data.usgs.gov/lod/topo/Structure /Struct_Point
http://www.w3.org/2000/01/rdf-schema#range: http://www.w3.org/2001/XMLSchema#string
http://data.usgs.gov/lod/topo/allows_Nulls: true
http://data.usgs.gov/lod/topo/comments:
http://data.usgs.gov/lod/topo/default_Value:
http://data.usgs.gov/lod/topo/definition: City of the address.
http://data.usgs.gov/lod/topo/length: 40
http://data.usgs.gov/lod/topo/precision:

Figure 4: The project UI showing how the metadata associated with an entity were displayed as RDF triples.

A challenge of using a manual approach to creating an OWL ontology from pre-existing data includes the reuse of attribute labels. Common data properties for geospatial data include human-readable names and coordinate information, but standard namespaces for these attributes vary. Across the different products published, there are a variety of attributes with similar names but different associated metadata. Labelling an instance with a class in an OWL ontology allows that instance to inherit the attributes, additional properties, and rules of the table with which it was created. Commonly used OWL annotation properties can more quickly find similarities that enable users to determine relations among resources.

2.4 Query and Retrieval

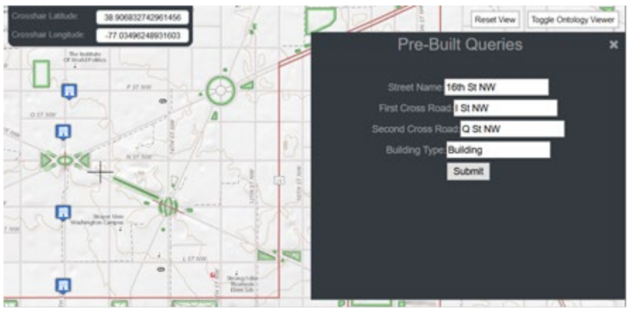
The interface provides a SPARQL and GeoSPARQL query window that allows a user to create or input their choice of pre-built queries. A custom Query Builder allows users to generate queries based on different parameters for the graphs currently in the triplestore. Pre-built GeoSPARQL queries support topological spatial relations that were designed to demonstrate some basic functions that allow a user to travel across different datasets without the need for background knowledge. Two supported functions are "Nearby Points" and "Entities Within". Both functions work by retrieving the entity's geometry as WKT, creating a buffer around that point, and returning all the entities whose geometry is contained within that buffer. The topology of the selected geometry objects was not structured. The GeoSPARQL Query with INPUT_NAMESPACE is replaced with the graph the user wants to search and INPUT_GEOMETRY is replaced with the WKT geometry the user is using to create the buffer. The "Entities Within" function pulls in the geometry of the selected entity using its Universal Resource Identifier (URI) and compares all the entities within a graph to that entity using the geof:sfWithin function. The GeoSPARQL Query for "Entities Within" INPUT_URI is replaced with the URI of the entity the user wants to find search inside of (Figure 5). The query searched a graph relating to structure points, using the GeoSPARQL function sfWithin coupled with the latitudes and longitudes of each user created point geometry, area in blue.



```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>PREFIX sf:
<http://www.opengis.net/ont/sf#> PREFIX geof: <http://www.opengis.net/def/function
/geosparql/> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> SELECT
?subject ?geom ?gml ?wkt ?fcode ?ftype FROM NAMED <http://localhost:8080
/marmotta/context/v2/struct_point> WHERE { GRAPH ?g { ?subject
geo:hasGeometry ?geom . ?geom geo:asGML ?gml . ?geom geo:asWKT ?wkt .
OPTIONAL { ?subject <http://dbpedia.org/ontology/purpose> ?purpose . } OPTIONAL
{ ?subject <http://purl.org/dc/elements/1.1/title> ?name . } OPTIONAL { ?subject
<http://data.usgs.gov/lod/topo/fCode> ?fcode . } OPTIONAL { ?subject
<http://data.usgs.gov/lod/topo/ftype> ?ftype . } OPTIONAL { ?geom geo:dimension
?dimensions . } FILTER (geof:sfWithin(?wkt, "MULTIPOLYGON
(((-77.1562957763672 38.88472811169486, -77.05329895019533
38.946465469205435, -77.05020904541017 38.905580497738384,
-76.98875427246095 38.901838212628284, -77.0351028442383
38.88419335462494, -76.93347930908205 38.88820393453861,
-77.07561492919923 38.85959017863306, -77.06668853759767
38.88954074419599, -77.1562957763672 38.88472811169486)))"^^geo:wktLiteral)) } }
```

Figure 5: The project UI showing generated output of a custom GeoSPARQL query from the query builder.

As an example of a problem where the parameters are not clearly defined, one pre-built query model was for finding buildings along a road between two intersections. The distance between a roadway and buildings along that roadway is not constant and can be different depending on the size of the roadway itself, the local zoning laws, and other factors, such as elevation, etc. The initial GeoSPARQL approach was to find starting and ending points based on the latitude and longitude of the inputted intersections and then draw a line between those two points. This approach would be fast and work if the roadway itself was a straight line, but most are not; roads have curves and change direction, so this initial approach failed. The problem was instead approached from a perspective of a person walking along the roadway, beginning with the initial starting point of the first intersection and iteratively moving in a straight line in small incremental steps. If the roadway was lost, the path was perpendicularly adjusted at the previous step until found again. This process continued until arriving at the ending intersection, creating a greedy algorithm such that the locally optimal solution is also the globally optimal solution. A polygon is superimposed over this path, whose orthogonal dimension was arbitrary to effectively create a shape. The problem was solved using the GeoSPARQL function sfTouches to find any objects whose latitude and longitude touched the created shape. Figure 6 illustrates this functionality.



Crosshair Latitude: 38.906832742961456
Crosshair Longitude: -77.03496248931603

Street Name: 16th St NW
First Cross Road: I St NW
Second Cross Road: Q St NW
Building Type: Building

Figure 6. Pre-Built query finding buildings depicted in blue between two road intersections.

Linked Open Data were retrieved using SPARQL endpoints to test linking triples. The LOD cloud provides a vast amount of data that could be related to any given entity within the MapKB system. An earlier version of this system mapped features to owl:sameAs based on geometric (location point) and semantic rules to link individuals between different datasets and ontologies (Wagner, Varanka, and Usery, 2020). However, most LOD providers create and publish their data using a custom ontology, so linkset relationships must be pre-defined and created, requiring advanced knowledge of the ontologies and their exact meanings to avoid errors because data can vary for the same instances. Entities linked by owl:sameAs may refer to the same thing, but their graphs may conflate conflicting context-dependent descriptions through inference potentially due to conflicting literal values for same or similar properties, e.g., five- vs nine-digit zip codes. This problem requires pre-computation on the publisher's end to determine whether two entities actually are the same, perhaps through advanced algorithms and mathematical comparison such as those developed automated ontology alignment. The lack of this data creates a hurdle for leveraging different LOD publishers to enhance instance data.

SPARQL and other REST-like interfaces were used to attempt to find relevant or similar entities on the fly to leverage the LOD cloud to enhance the information available to users. This increases the size of the knowledge graph available to traverse without creating links between separate graphs. MapKB supports links to two LOD publishers, DBPedia and Wikidata, and one non-LOD publisher demonstrating that additional information regarding an entity can be gathered without the use of the owl:sameAs relationship. One major benefit of geographical data is that this provides a good avenue outside of plain-text names and attributes to compare relevant entities. In all three links, geometry coordinates were used to find related information. The approach used within the MapKB interface to link to the LOD cloud was to pull in the data for the requested object and send a query to other SPARQL endpoints to attempt to find relationships on-the-fly. The graph patterns required significant research into the properties and format of the target datasets to produce. To link our data to Wikidata, just geographic coordinates were used. Due to the time constraint on their SPARQL endpoint, additional attributes such as the associated continent and country were used to reduce the target graph size so that the query could be completed in time. The graph pattern of the query used to link data from DBPedia with The National Map dataset is simpler than the one used to link with Wikidata and uses some popular ontologies such as rdf-schema and GeoSPARQL. Using different available ontologies enabled different levels of complexity to search. However, finding similar entities to most entities within our MapKB datasets failed due to the immaturity of the comparison operators. There is a need to find those relationships without the existence of a 'seed' triple.

3. RESULTS

The MapKB development investigated three main approaches for data interoperability with The National Map: the creation of an OWL ontology transformed from existing data, a cartographic UI for graph-based data, and query processing for SPARQL endpoints. The data transformation process to improve technical and semantics interoperability aimed to move the schematic design of the ontology to more closely resemble knowledge from information. As a conceptual and logical system for organizing knowledge, ontology design raised questions and discussion within the data publishing enterprise about the product semantics before it was applied as a graph database. Questions that were encountered during the process of transforming to a graph data model eliminated legacy constraint of the source technology. Specific questions included redundant attribute fields, the creation of object properties to support automated data linking, and conformance to OWL.

The MapKB functionality was validated by testing the ontology axioms through the application of OntoDebug (2022) plug-in reasoners; the inferenced information accessed through an OP; and by successful information query and retrieval approaches using SPARQL and GeoSPARQL. The post-processing stage of the ontology creation demonstrated the inclusion of metadata from spatial resources supported OP and LOD alignment.

An ontology pattern of aligning feature classes represented as codes and geometries of The National Map matched to the GeoSPARQL ontology feature and geometry classes. The ontology for feature interoperability provided inferred information for competency questions such as "What type of feature is classified as FCode 73002," or "How are streams represented geometrically?" OP created as a subgraph of the core ontology was supported by the inclusion of metadata using annotation properties. Multiple GeoSPARQL queries executing topological relations on features were successfully demonstrated with a pre-built query window to find specified buildings on a road section between two cross streets. The success of such a query can depend on the shape of the road, building distance from the roadway, and other factors. The queries required a change in viewpoint from machine computation to landscape cognition involving other related semantic factors, and then were followed by customized GeoSPARQL function computation.

The semantic specification of topographic features and relations adaption with other LOD entities were tested. Basic on-the-fly entity resolution was attempted by comparing the data of a given entity to entities located in the LOD cloud to learn more about an entity. The GKG alignment with LOD used some specific widely used vocabularies to be reused between graphs, and problems encountered could be resolved by designing a better metadata annotation approach for structural alignment in addition to syntax matching.

4. CONCLUSIONS

The MapKB project tested key challenges for GKG applications for spatial data infrastructure interoperability including data transformation, ontology design, spatial and non-spatial information search and retrieval, and multi-modality cartographic visualization. RDF and OWL vocabulary were sufficiently expressive to demonstrate linking and reasoning successes.

Challenges in completing the resulting ontology from automated data transformation for knowledge representation still involved primarily cognitive activities. Entity relationships are not sufficiently defined between datasets in the LOD cloud. More work needs to be done to perform accurate on-the-fly entity resolution for geospatial properties. Although initial tests of GeoSPARQL techniques were successful, the full capabilities of SPARQL as a rule-based reasoning tool would need further research for queries that can leverage the full capabilities of GKG and for their semantic portrayal, including the need for cartographic visual language of underlying graph structures.

ACKNOWLEDGEMENTS

Programming and other technical support for the MapKB project was completed by Matthew E. Wagner, Jacques Bourquin, and Tanner Fry and is gratefully acknowledged.

REFERENCES

Agafonkin, V., 2022. Leaflet, an open-source JavaScript library for mobile-friendly interactive maps. https://leafletjs.org (18 April 2022).

Bourquin, J., 2021. Map as a Knowledge Base (MapKB). https://code.usgs.gov/makb (25 May 2022).

Das, S., Sundara, S., and Cyganiak, R., eds., 2012. R2RML—RDB to RDF mapping language. W3C https://www.w3.org/TR/r2rml/ (25 May 2022).

Hayes, P.J. and Patel-Schneider, P.F., eds., 2014. RDF 1.1 Semantics. W3C. https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/ (11 February 2022).

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., and Rudolph, S., 2012. OWL 2 Web Ontology Language Primer (Second edition). W3C. https://www.w3.org/TR/owl2-primer/ (11 February 2022).

Kuhlthau, C., 2004: *Seeking Meaning: a process approach to library and information services*. Libraries Unlimited.

Lott, R., ed., 2015. Geographic information—Well-known text representation of coordinate reference systems: Open Geospatial Consortium document 12–063r5. http://docs.opengeospatial.org/is/12- 063r5/12-063r5.html (2 August 2022).

Musen, M.A., 2015. The Protégé project: A look back and a look forward. *AI Matters*. *Association of Computing Machinery Specific Interest Group in Artificial Intelligence* 1(4), 4-12.

OntoDebug. OntoDebug - Interactive Ontology Debugging in Protégé (aau.at) (23 May 2022).

Open Geospatial Consortium, 2022. OGC GeoSPARQL. https://github.com/opengeospatial/ogc-geosparql (11 February 2022).

Open Source Geospatial Foundation, 2019a. GeoServer. http://geoserver.org/ (2 August 2022).

Open Source Geospatial Foundation, 2019b. GDAL. https://gdal.org/ (2 August 2020).

OpenLink Software, 2021. Virtuoso Open-Source Edition. https://virtuoso.openlinksw.com (25 May 2022)

University of Southern California, 2016. Karma—A data integration tool. University of Southern California. http://usc-isi-i2.github.io/karma/ (25 May 2022).

U.S. Geological Survey, 2022, TNM download (v2.0): U.S. Geological Survey. https://apps.nationalmap.gov/downloader/ (8 July 2022).

Wagner, M.E., Varanka, D.E., 2020: Creating Annotation Property for OWL Ontologies Generated from Relational Databases. V. Villazón-Terrazas et al. (Eds.): Knowledge Graphs and Semantic Web 2020, Communications in Computer and Information Science CCIS 1232, 1-16.

Wagner, M., Varanka, D.E., and Usery, E.L., 2020, A system design for implementing advanced feature descriptions for a map knowledge base: U.S. Geological Survey Scientific Investigations Report 2019–5148, 25 p., https://doi.org/ 10.3133/sir20195148.