A Semantic Approach to Retrieving, Linking, and Integrating Heterogeneous Geospatial Data

Ying Zhang
North China Electric Power University
Beijing, China
yingzhang@ncepu.edu.cn

Yao-Yi Chiang
University of Southern California
California, USA
yaoyichi@isi.edu

Pedro Szekely
University of Southern California
California, USA
pszekely@isi.edu

Craig A. Knoblock
University of Southern California
California, USA
knoblock@isi.edu

ABSTRACT

There is a tremendous amount of geospatial data available, and there are numerous methods for extracting, processing and integrating geospatial sources. However, end-users' ability to retrieve, combine, and integrate heterogeneous geospatial data is limited. This paper presents a new semantic approach that allows users to easily extract, link, and integrate geospatial data from various sources by demonstration in an interactive interface, which is implemented in a tool called Karma. First, we encapsulate the retrieval algorithms as web services and invoke the services to extract geospatial data from various sources. Then we model and publish the extracted geospatial data to RDF for eliminating the data heterogeneity. Finally, we link the geospatial data (in RDF) from different sources using a semantic matching algorithm and integrate them using SPARQL queries. This approach empowers end users to rapidly extract geospatial data from diverse sources, to easily eliminate heterogeneity and to semantically link and integrate sources.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods— *Semantic Networks*; H.2.8 [**Database Management**]: Database applications—*Spatial databases and GIS*; H.3.5 [**Information Storage and Retrieval**]: On-line Information Services; H.5.2 [**Information Inter- faces and Presentation**]: User Interfaces—*Theory and methods*.

General Terms

Algorithms, Performance, Experimentation

Keywords

Geospatial data, extraction, semantic modeling, ontology, source linking, information integration

1. INTRODUCTION

There is a great variety of geospatial data becoming available every day and access to these data is important for end-users. Although large amounts of data are produced, it is still difficult to search for geospatial data sources over the Web, and the ability of end-users to retrieve, combine and integrate geospatial data is limited.

Typically there are three steps in the process of obtaining complete information from various sources. The first step is to extract geospatial data from various sources; however, because there is no semantic meaning in the extracted geospatial data, users can not fully understand and use them very well. Second, once the data is extracted, the next step is to find the links between different sources. In previous work, this has been done by defining a set of semantic rules for performing the linking [1-5]. However, the geospatial data has some specific geospatial relationships such as overlaps, contains and distance, which cannot be solved by semantic web techniques. The third step is to merge the linked records by eliminating data redundance and combining complementary properties. Performing union work efficiently on all extracted geospatial data in repository is a problem. In this paper, we union the attributes of the same geospatial entity from multiple sources.

This paper provides an easy way to fully exploit the growing amount of heterogeneous geospatial data to users without adequate background knowledge. The workflow for solving this task is shown in Figure 1. The three processes in the presented approach are modeling, linking and integrating. Each process is used to solve specific problems for each corresponding step as mentioned afore. We will discuss each process in detail in the following sections.

Our approach is based on the prior work about Karma [6-7], which is an information integration tool that can partly support the modeling process. In addition, we take advantage of Karma's visulization function to display the extracted geographic data in the Web browser using the Google Earth plug-in.

In the next section we present a motivating example of extracting, modeling, linking and integrating geospatial data from different sources. Sections 3 through 5 elaborate the detail processes of our approach. We then describe the related work and conclude with a discussion of our contribution.

2. MOTIVATING EXAMPLE

In this section we describe the example that we use to motivate the remainder of the paper. We take Wikimapia and OpenStreetMap as the example sources in this paper. Wikimapia provides names, latitudes, longitudes and polygon outlines for building entities, while OpenStreetMap gives elevation and address information, such as state and county name, in addition to the building names, longitudes, latitudes and polygons. For the same building, the extracted name from Wikimapia might be totally different than that provided by OpenStreetMap. Moreover, for the same building, the location information might be different. For instance, we can have

AIIP'13, August 04 - 05, 2013, Beijing, China.
Copyright 2013 ACM 978-1-4503-2346-8/13/08 …$15.00.

the building which is named “Wallis Annenberg Building for Science Learning and Innovation” by OpenStreetMap with coordinate “Point(-118.283789, 34.017568)”. In contrast, the same building is named “Science Center School” in Wikimapia, and its coordinate is “Point(-118.2837975,34.017478)”. These problems might puzzle end-users. Without Karma, users need to union by checking each entity from different sources manually. This paper addresses this problem and provides users with complete and integrated information.

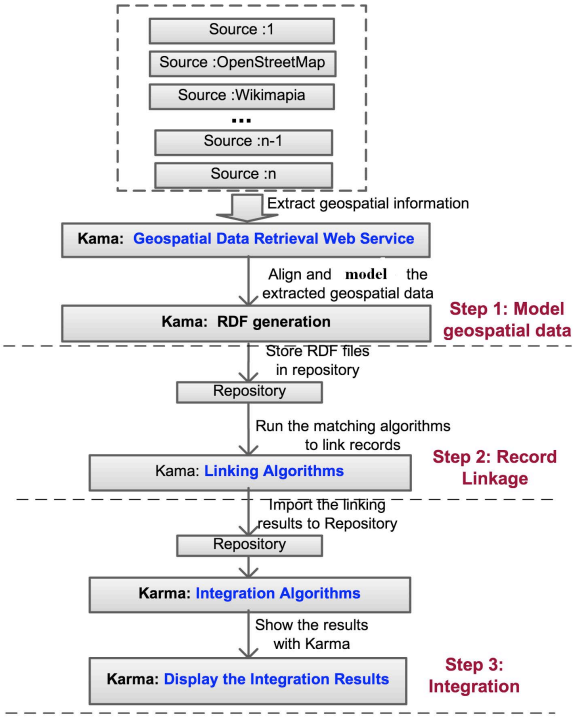
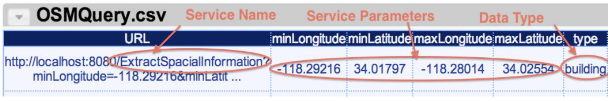

Source :1
Source :OpenStreetMap
Source :Wikimapia
...
Source :n-1
Source :n
Extract geospatial information
Kama: Geospatial Data Retrieval Web Service
Align and model the extracted geospatial data
Kama: RDF generation
Step 1: Model geospatial data
Store RDF files in repository
Repository
Run the matching algorithms to link records
Kama: Linking Algorithms
Step 2: Record Linkage
Import the linking results to Repository
Repository
Karma: Integration Algorithms
Show the results with Karma
Karma: Display the Integration Results
Step 3: Integration

Figure 1. The overview of geospatial data retrieval, linkage and integration.

3. Model the Geospatial Data

In this section, we first extract geospatial data. We encapsulate the retrieval algorithms as Web Services, and provide end-users with a uniform interaction paradigm to access services. Next we map the data to a generic geospatial ontology with Karma, and the generated RDF data is used to support linking and integrating work.

3.1 Extract the Geospatial Data



OSMQuery.csv

URL	minLongitude	minLatitude	maxLongitude	maxLatitude	type
http://localhost:8080/ExtractSpacialInformation?minLongitude=-118.29216&minLatit ...	-118.29216	34.01797	-118.28014	34.02554	building

Figure 2. Queries with a uniform interaction paradigm

The previous work on Karma presented an approach that allows domain experts to create their own services with invocation URLs. The domain experts can also rapidly create semantic models of services and use them to produce linked data [8]. In this paper, we encapsulate the retrieval algorithms as a Web Service, and embed all the inputs in a URL. Since users usually search geospatial data by limiting a predefined bounding box with a specified type, the invocation URL consists of three kinds of information: service name, bounding box and the required geospatial data. In the example shown in Figure 2, we extract building information around USC. “ExtractSpatcialInformation” in URL is a service name, four parameters consist of a bounding box of USC campus: minimum longitude, minimum latitude, maximum longitude and maximum latitude, and “building” represents the required geospatial data type. After uploading this query, Karma can invoke corresponding Web Service, and produce a worksheet that contains the extracted output.

3.2 Generic Geospatial Ontology

Because the format and content of various sources are different from each other, and there is no semantic meaning in the extracted data, it is difficult for users to understand and use the data. In order to model the semantics of geospatial sources, we built a generic geospatial ontology for aligning the extracted data.

Figure 3 shows the structure of the generated generic geospatial ontology. The Layer class includes RoadLayer and BuildingLayer subclasses. The subclasses and properties of RoadLayer class are used to model geospatial data of road type. Similarly, the building type of geospatial data is represented by the subclasses and properties of BuildingLayer class. The PointFeature class and PolygonFeature class are used to model the Building entity, while PolylineFeature class is exploited to describe the Road entity. SpatialReferenceSystem class is used to define a specific map projection for sources.

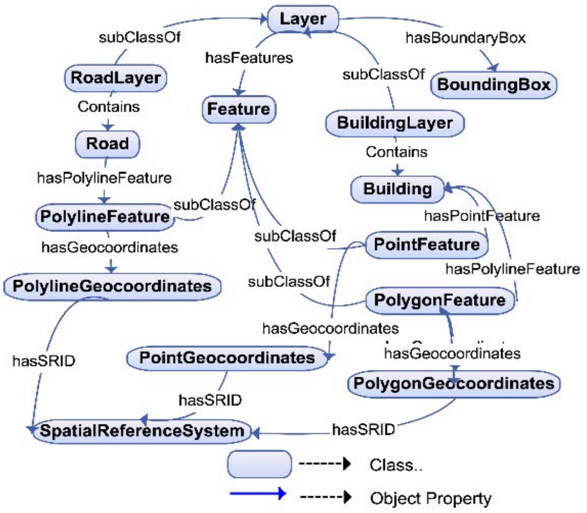
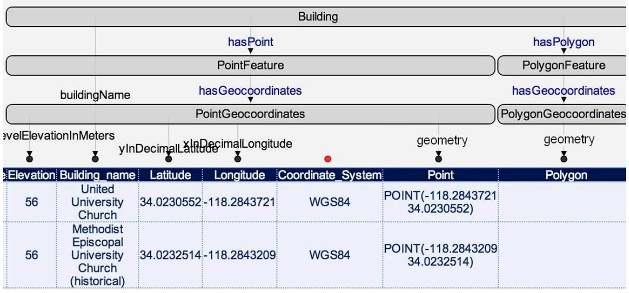


Figure 3. The generic geospatial ontology

3.3 Map the Extracted Geospatial Data to RDF

Based on the generic geospatial ontology, we can model the extracted geospatial data by mapping them to RDF data with Karma. The previous work about Karma discussed the detail about how to map structured data to RDF according to an ontology given by users [9-10]. Karma provides a visual interface where users can see the mappings. Users can also adjust them if necessary. The approach of Karma to mapping data to ontology involves two steps:

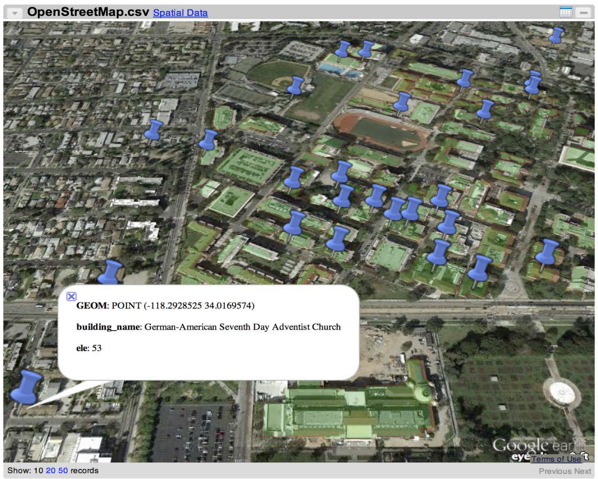
linking data columns to semantic types and specifying the relationships between semantic types. For the first step, the data columns (obtained through the invocation URL given by Figure 2) are linked with the ranges of data properties. For example, the bottom connections between columns and ontology as shown in Figure 4 are all data properties such as buildingName, xInDecimalLongitude, yInDecimalLatitude and geometry. In the second step, Karma uses paths of object properties to specify the relationships between semantic types. Figure 4 describes the object properties in blue. After publishing RDF, the structured geospatial data are mapped to RDF data, on which the linking process can be applied.



Elevation	Building_name	Latitude	Longitude	Coordinate_System	Point	Polygon
56	United University Church	34.0230552	-118.2843721	WGS84	POINT(-118.2843721 34.0230552)	
56	Methodist Episcopal University Church (historical)	34.0232514	-118.2843209	WGS84	POINT(-118.2843209 34.0232514)	

Figure 4. Mapping with the generic geospatial ontology.

3.4 Display on Google Earth



GEOM: POINT (-118.2928525 34.0169574)

building_name: German-American Seventh Day Adventist Church

ele: 53

Figure 5. Display on Google Earth.

Figure 5 illustrates the extracted geographic information for the building scenario visualized on Google Earth with Karma. Each building, shown as a pin located at a place which can be found by the given values of latitude and longitude, is represented by "point" in the source data. The green polygons also denote buildings when they are described by "polygon" in the source. In Figure 5, the bubble, which appears when users click on pin or polygon, displays the details about that building.

4. Geospatial Data Linking Process

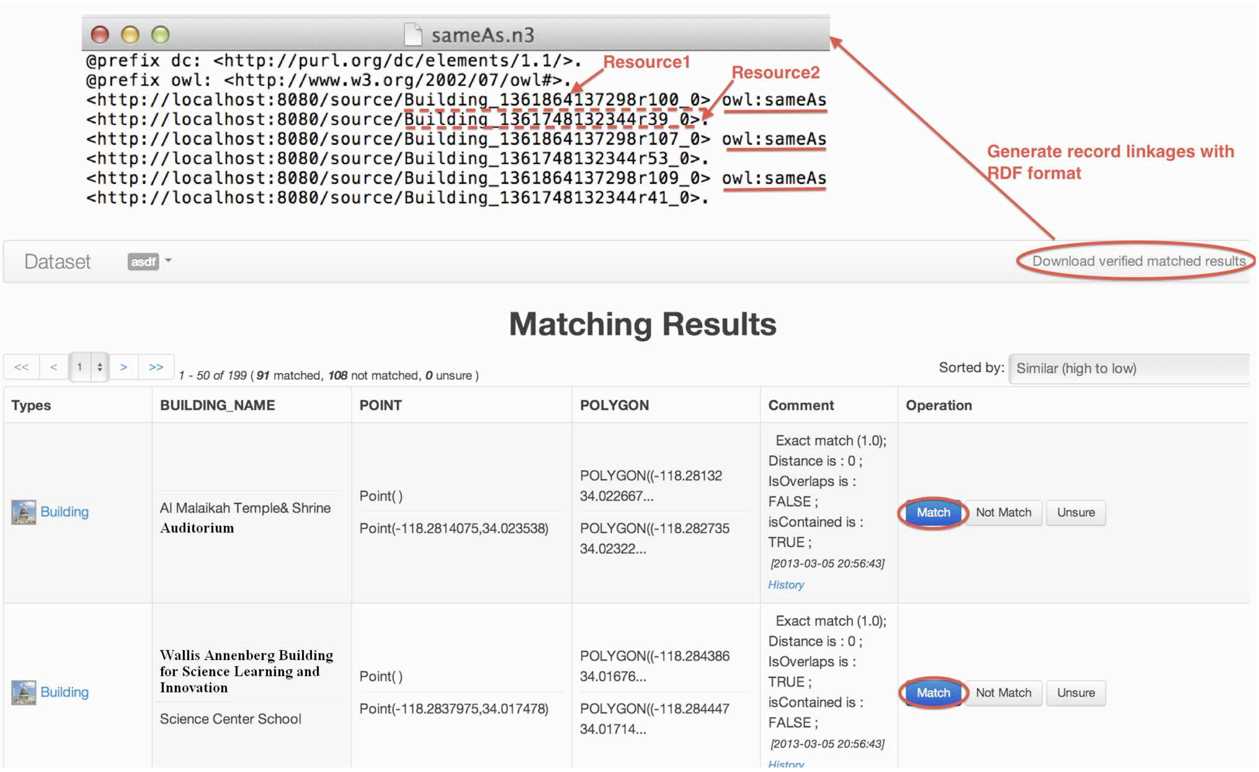
Because the extracted geospatial data varies from source to source, some of them are duplicated, while others are complementary to each other. Taking one building entity as an example, users can get attributes such as building name, latitude, longitude and polygon from Wikimapia, and retrieve other attributes such as elevation, state name and county name from OpenStreetMap. Both of them share the same geographic information such as name, latitude, longitude and polygon. In order to provide users with complete information, we need to merge the extracted geospatial data from all the sources. However, the merging process is based on the linking process, which could recognize the same entity from different sources through linking algorithms. The linking algorithm is shown in Figure 6, and the details are discussed in the following subsection.

```
Input:  RDF file: $Source_1$;  RDF file: $Source_2$;
Output: LinkedRecord[] matchedPair;
Step1:
    repository← $Source_1$;
    repository← $Source_2$;
    Attributes[] $comm_1$←extract common attributes from
                     $Source_1$;
    Attributes[] $comm_2$← extract common attributes from
                     $Source_2$;
Step2:
 for all $s_1$ in $Source_1$ and $c_1$ in $comm_1$ do
        if($c_1$.hasPolygon!=null){
              $s_1$.location← $c_1$.getPolygon;
        }else if($c_1$.hasPoint!=null){
              $s_1$.location← $c_1$.getPoint;
        }
    for all $s_2$ in $Source_2$ and $c_2$ in $comm_2$ do
      if($c_2$.hasPolygon!=null){
        $s_2$.location← $c_2$.getPolygon;
      }else if($c_2$.hasPoint!=null){
        $s_2$.location← $c_2$.getPoint;
      }
     distance←executeQuery("Select ST_Distance(ST_GeographyFromText($s_1$.getLocation),ST_GeographyFromText($s_2$.getLocation))")
     isContained←executeQuery("Select ST_Contains(ST_GeographyFromText($s_1$.getLocation),ST_GeographyFromText($s_2$.getLocation))")
     isOverlap←executeQuery("Select ST_Overlaps(ST_GeographyFromText($s_1$.getLocation),
     ST_GeographyFromText($s_2$.getLocation))")

     if ($c_1$.hasPolygon and $c_2$.hasPolygon){
                  if (isContained=true){
              similarity ← 1.0;
                  }
     else {
              similarity ← $\left|1-\frac{distance}{threshold}\right|$}
Step3
    if (similarity>0.97){
      matchedPair.add($s_1$,$s_2$);
    }
```

Figure 6. Linking algorithm

4.1 Linking Algorithms


sameAs.n3
Resource1
Resource2
Generate record linkages with RDF format
Dataset
asdf
Download verified matched results

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
<http://localhost:8080/source/Building_1361864137298r100_0> owl:sameAs
<http://localhost:8080/source/Building_1361748132344r39_0>.
<http://localhost:8080/source/Building_1361864137298r107_0> owl:sameAs
<http://localhost:8080/source/Building_1361748132344r53_0>.
<http://localhost:8080/source/Building_1361864137298r109_0> owl:sameAs
<http://localhost:8080/source/Building_1361748132344r41_0>.
```

Matching Results

1 - 50 of 199 (**91** matched, **108** not matched, **0** unsure)

Sorted by: Similar (high to low)

Types	BUILDING_NAME	POINT	POLYGON	Comment	Operation

Building	Al Malaikah Temple& Shrine
	Auditorium
		Point( )
		Point(-118.2814075,34.023538)
			POLYGON((-118.28132 34.022667...
			POLYGON((-118.282735 34.02322...
				Exact match (1.0); Distance is : 0 ; IsOverlaps is : FALSE ; isContained is : TRUE ; *[2013-03-05 20:56:43]*
				History
					Match Not Match Unsure

Building	Wallis Annenberg Building for Science Learning and Innovation
	Science Center School
		Point( )
		Point(-118.2837975,34.017478)
			POLYGON((-118.284386 34.01676...
			POLYGON((-118.284447 34.01714...
				Exact match (1.0); Distance is : 0 ; IsOverlaps is : TRUE ; isContained is : FALSE ; *[2013-03-05 20:56:43]*
				History
					Match Not Match Unsure

Figure 7. Record linkage results

The linking algorithms include three steps. Firstly, we extract common attributes including building name, latitude, longitude and polygon from the sources. Based on these attributes, step 2 calculates the similarity between building entities from various sources. It adopts PostgreSQL data types to represent geospatial information of builings. Then, it uses three functions supported by PostGIS to get the distance between two buildings, and to determine the relationships such as "contain" and "overlap" between them. If both buildings have polygon types and one is contained by the other, then they are the same building. Otherwise, if they are overlapped and the spheroid distance between them is smaller than a specified value, then they are the same. In another situation which suppose that one of buildings has only Point type instead of Polygon type, we find that the possibility of similarity is high if the spherioid distance is less than the specified value. In this experiment, we use the extracted geospatial data of USC campus. We can get 145 building entities from OpenStreetMap, and 99 building entities from Wikimapia. The precision is higher than 97% if we set the specified value equal to 25 meters.

4.2 Record Linkage

Figure 7 illustrates the linking results. We can see that each matched pair consists of two rows. One is from Wikimapia, the other is from OpenStreetMap. They are linked together since the probability that they are the same building is high. The comment column gives the reasons for high similarity. For instance, two buildings have separate names and coordinates; however, the "contained" relationship and the spheroid distance between them is zero, so they get similarity degree equal to 1.0. The operation column gives a recommended result like "Match" on the condition that the similarity degree is higher than the threshold give by the user.

The linking process can also generate the record linkages in RDF format based on an equivalence property like owl:sameAs. As shown on the top layer of Figure 7, two related resources are connected to each other by property owl:sameAs if these two resources are matched. The record linkage results in RDF format will be stored in the triple store and used for queries for integrating data across sources in the next section.

5. Geospatial Data Integration

Our goal in data integration is to provide users an easy way to get complete and integrated geospatial data. Based on the first two steps, users can access various extracted geospatial data from different sources, and all the similar entities can be linked together. The integration process eliminates the data redundance, and combines the complementary properties from the linked data.

5.1 Integration Algorithms

The algorithm shown in Figure 8 is used to do union work on the linked records. The inputs of integration algorithms have already been stored in the repository in the former modeling and linking processes. In the first step, SPARQL queries are used to extract all the values of attributes given by each source from every matched pair. We use a general query sparqlS1(shown in Figure 9) to fetch

```
Input:
   RDF file: $Source_1$;
   RDF file: $Source_2$;
   RDF file: recordLinkageResult;
Output:
   Record[] unionRecord;
Step1:
   URIs←fetch all the resource URI from $Source_1$
   with    $sparqlS_1$
   for all $URI_1$ in URIs do
      $queryS_2$←QueryFactory.create($sparqlS_2$);
      qexe←QueryExecutionFactory.create($queryS_2$
      , dataset);
      $resultsS_1$←qexe.execSelect();
      $queryS_3$←QueryFactory.create($sparqlS_3$);
      qexe←QueryExecutionFactory.create($queryS_3$
      , dataset);
      $resultsS_2$←qexe.execSelect();
Step2:
      if (!$resultsS_1$.equalsIgnore($resultsS_2$)) {
         $resultsS_1$.addAll($resultsS_2$);
      }
      unionRecord.add($resultsS_1$);
```

Figure 8. Integration algorithm

```
(1) $sparqlS_1$ : general query
Select discinct ?uri
Where{
   ?uri owl:sameAs ?u.
   ?uri a BuildingOntology:Building
}
(2) $sparqlS_2$ : retrieve values from $Source_1$
Select ?longitude
Where{
   <subject> BuildingOntology:hasPoint ?p.
   ?p  BuildingOntology:hasGeocoordinates ?g.
   ?g
BuildingOntology:xInDecimalLongitude ?longitude
}
(3) $sparqlS_3$ : retrieve values from $Source_2$
Select ?longitude
Where{
   <subject>  owl:sameAs  ?u
   ?u  BuildingOntology:hasPoint ?p.
   ?p  BuildingOntology:hasGeocoordinates ?g.
   ?g
BuildingOntology:xInDecimalLongitude ?longitude
}
```

Figure 9. SPARQL queries

all the URIs from Source1. All the URIs of Source1 can be connected to the corresponding URIs of Source2 with property owl:sameAs. Then based on the retrieved URI, which is named as subject, we can get all the values of properties such as building name, latitude, longitude, polygon and county name. Taking longitude as an example, the second query sparqlS2 described in Figure 9 extracts the longitude value of one building from Source1, while the third query sparqlS3 retrieves the longitude value of the matched building from Source2. In the next step, the union work eliminates the duplicate values of properties and combines those complementary properties from different sources.

5.2 Display Integration Results

Figure 10 illustrates the integration results. A pair of rectangles denotes a matched pair of entities. The green rectangle means that the property values come from Source1, while the blue rectangle indicates that the property values come from Source2. The red rectangles specify that both sources share the same values of that property. For instance, Wikimapia and OpenStreetMap have a common value "Los Angeles" for "county name" property.

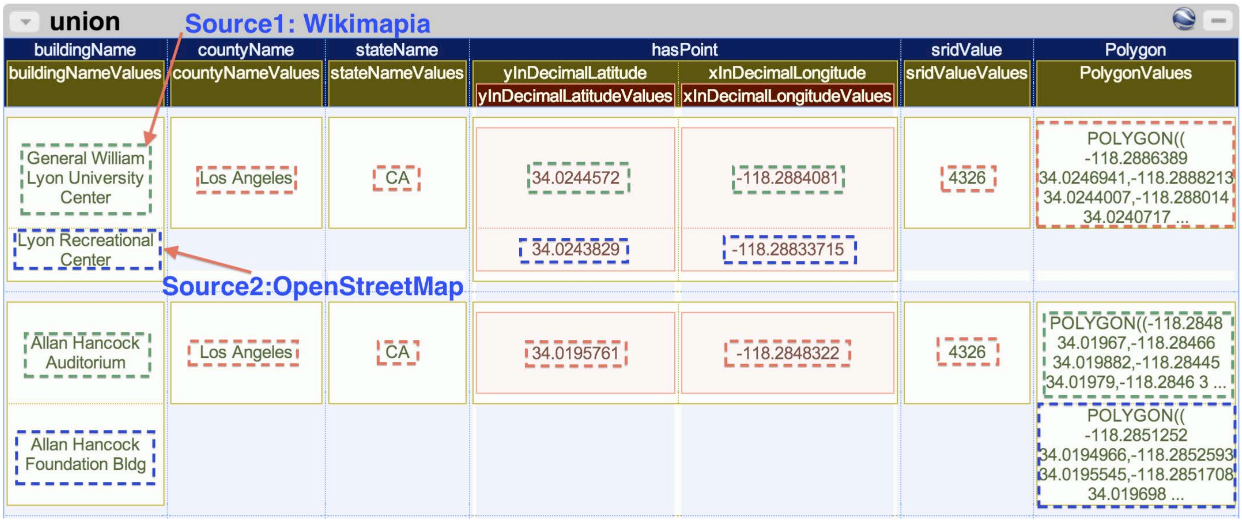
6. Related Work

There is significant work on geospatial data retrieving, linking and integrating. Much of the work exploits semantic Web technologies to facilitate retrieval and integration of geospatial data. Shyu et al [11] put forward a GeoIRIS system, which includes automatic feature extraction and visual content mining from large-scale image databases. Wiegand et al [12] present a task-based ontology approach to automate geospatial data retrieval. With the ontology, reasoning can be done to infer various types of information that can meet specific criteria for use in particular tasks. However, there are many comparisons in the reasoning process. Because the retrieval of geospatial data is typically hindered by domain-related terminology mismatches, Fugazza [13] set up a knowledge base that served as groundwork for harmonising domain knowledge from distint areas. In this paper, we work to extract, model, link and integrate geospatial data from different sources. We also build a generic geospatial ontology to get specific geospatial data. However, we do not need to do many comparisons to get the particular geospatial data, since we take advantage of Karma to map and align the retrieved geospatial data according to the provided generic geospatial ontology. In additon, the retrieval of geospatial data is not hindered by the domain-related terminology mismatches in this paper, because the retrieving process happened before the semantic mapping process.

On the modeling work, Google Refine [14] provides users with an interface to import data from various sources like CSV, XML, etc, and allows users to model sources by aligning the columns to Freebase schema types. It also provides capabilities to integrate data from sources but does not take advantage of semantic Web techniques. Yue et al [15-16] present an approach to add semantics into current geospatial catalogue services for geospatial data discovery and processing. They created three types of ontologies to define data and service semantics that enable dynamic and automatic composition of geospatial Web service chains to achieve a complex geospatial goal. However, service chains required the exact data types of input and output of services. This kind of "DataType" driven service composition cannot work automatically. In addition, their work was based on a semantic match of geospatial scientific theme ontologies. It didn't consider the useful spatial characteristics of geospatial data. In contrast, our approach does the linking and integrating work on the basis of both semantic techniques and the spatial relationships such as overlaps, contains and distance, which are significant in the respect of geospatial data.

7. Discussion and Future Work

This paper presents a new approach that allows users to easily



buildingName	countyName	stateName	hasPoint		sridValue	Polygon
buildingNameValues	countyNameValues	stateNameValues	yInDecimalLatitude	xInDecimalLongitude	sridValueValues	PolygonValues
			yInDecimalLatitudeValues	xInDecimalLongitudeValues		
General William Lyon University Center	Los Angeles	CA	34.0244572	-118.2884081	4326	POLYGON((-118.2886389 34.0246941,-118.2888213 34.0244007,-118.288014 34.0240717 ...
Lyon Recreational Center			34.0243829	-118.28833715		
Allan Hancock Auditorium	Los Angeles	CA	34.0195761	-118.2848322	4326	POLYGON((-118.2848 34.01967,-118.28466 34.019882,-118.28445 34.01979,-118.2846 3 ...
Allan Hancock Foundation Bldg						POLYGON((-118.2851252 34.0194966,-118.2852593 34.0195545,-118.2851708 34.019698 ...

Figure 10. The output results of integration in Karma

extract, link and integrate geospatial data from various sources by means of both semantic techniques and spatial characteristics. The presented approach empowers end users to rapidly extract geospatial data and semantically link and integrate them from various sources. In this paper, we take three steps to implement the presented approach. First, we extracted geospatial data from various sources. The retrieval algorithms are encapsulated as Web Services in Karma. Then we created a generic geospatial ontology to align the extracted geospatial data by mapping them to RDF data with Karma, which provides a visual interface where users can see and adjust the mappings. Second, we linked similar entities from different sources using an equivalence property like owl:sameAs based on the matched similarity. Third, we used SPARQL queries to eliminate data redundance and combine complementary properties for integration. In addition, we also took advantage of Karma's visulization function to illustrate running examples for each step.

For the problem of linking entities across sources, in addition to using spatial characteristics such as latitude, longitude, and spatial polygon, we are working to use other common attributes, such as the entity name. Using additional attributes we can optimize the geospatial data linking process and improve the integration results.

8. Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61305056, No.61001197), the Fundamental Research Funds for the Central Universities, the Department of the Navy, Office of Naval Research under Grant No. N00014-12-1-0121 and the National Science Foundation under Grant No. 1117913.

9. References

[1] Bonan Li and Frederico Fonseca. 2006. TDD-A comprehensive model for qualitative spatial similarity assessment. *Spatial Cognition* and Computation, 6, 1(2006), 31–62. DOI= 10.1207/s15427633scc0601_2.

[2] Krzysztof Janowicz, Simon Scheider. 2012. Todd Pehle and Glen Hart. Geospatial semantics and linked spatiotemporal data – Past, present, and future. *Semantic Web*, 3 (2012), 321-332.

[3] Krzysztof Janowicz, Martin Raubal and Werner Kuhn. 2011. The semantics of similarity in geographic information retrieval, *Journal of Spatial Information Science*, 2 (2011), : 29–57.

[4] Krzysztof Janowicz, Sven Schade, Arne Broring, Carsten Keler, Patrick Maue, Christoph Stasch. Semantic Enablement for Spatial Data Infrastructures. 2010. *Transactions in GIS.* 14,2 (2010), 111–129.

[5] Andrea Rodríguez and Max Egenhofer. Comparing geospatial entity classes: An asymmetric and context-dependent similarity measure. *International Journal of Geographical Information Science,* 18,3 (2004), 229–256. Doi= 10.1080/13658810310001629592

[6] Pedro Szekely, Craig A. Knoblock, Shubham Gupta, Mohsen Taheriyan and Bo Wu. Exploiting semantics of web services for geospatial data fusion, in: Proc. of the 1st ACM SIGSPATIAL International Workshop on Spatial Semantics and Ontologies,SSO'11. (Chicago, IL, USA — November 01 - 04, 2011), ACM New York, NY, USA, 32–39. Doi=10.1145/2068976.2068981

[7] Craig A. Knoblock, Pedro Szekely, Jose Luis Ambite, Aman Goel, Shubham Gupta, Kristina Lerman, Maria Muslea, Mohsen Taheriyan and Parag Mallick. Semi-automatically mapping structured sources into the semantic web. In: Proceedings of the Ninth Extended Semantic Web Conference (Crete, Greece, May 27-31, 2012). Springer-Verlag Berlin, Heidelberg, 375-390. Doi= 10.1007/ 978-3-642-30284-8_32

[8] Mohsen Taheriyan, Craig A. Knoblock, Pedro Szekely and Jose Luis Ambite. Rapidly Integrating Services into the Linked Data Cloud. In Proceedings of the 11th International Semantic Web Conference. (Boston, America, 2012). Spring-

er-Verlag Berlin, Heidelberg, 559- 574. Doi= 10.1007/978-3-642-35176-1_35.

[9] Shubham Gupta, Pedro Szekely, Craig A. Knoblock, Aman Goel, Mohsen Taheriyan, and Maria Muslea. Karma: A System for Mapping Structured Sources into the Semantic Web. In: Proceedings of the Ninth Extended Semantic Web Conference. (Crete, Greece, May 27-31, 2012). Springer-Verlag Berlin, Heidelberg.

[10] Craig A. Knoblock, Pedro Szekely, Jose Luis Ambite, Shubham Gupta, Aman Goel, Maria Muslea, Kristina Lerman, and Parag Mallick. Interactively mapping data sources into the semantic web. In Proceedings of the 1st Int. Workshop on Linked Science 2011 in Conjunction with the 10th Int. Semantic Web Conference, 2011.(Bonn, Germany. 24th, 2011)

[11] Chi-Ren Shyu, Matt Klaric, Grant J. Scott, Adrian S. Barb, Curt H. Davis, and Kannappan Palaniappan. 2007. GeoIRIS: Geospatial Information Retrieval and Indexing System-Content Mining, Semantics Modeling, and Complex Queries. *IEEE Transactions on geoscience and remote sensing*. 45,4 (2007),839-852. Doi= 10.1109/TGRS.2006.890579

[12] Nancy Wiegand, Cassandra García. 2007. A task-based ontology approach to automate geospatial data retrieval. *Transactions in GIS*. 11 (2007), 355–76.

[13] Cristiano Fugazza. 2011. Toward semantics- aware annotation and retrieval of spatial data. *Earth Sci Inorm*, 4 (2011), 225-239. Doi= 10.1007/s12145-011-0088-1

[14] David Huynh, Stefano Mazzocchi. Google refine. http://code.google.com/p/google-refine/

[15] Peng Yue, Liping Di, Wenli Yang, Genong Yu, Peisheng Zhao.2007. Semantics-based automatic composition of geospatial web service chains. *Computers and Geosciences*, 33,5(2007), 649-665. Doi= 10.1016/j.cageo.2006.09.003

[16] Peng Yue, Jianya Gong, Liping Di, Lianlian He and Yaxing Wei.2011. Integrating semantic web technologies and geospatial catalog services for geospatial information discovery and processing in cyberinfrastructure. Integrating semantic web technologies and geospatial catalog services for geospatial information discovery and processing in cyberinfrastructure. Geoinformatica, 15(2011),273-303. Doi=10.1007/s10707-009-0096-1