Exploiting Semantics of Web Services for Geospatial Data Fusion[*]

Pedro Szekely Craig A. Knoblock Shubham Gupta Mohsen Taheriyan Bo Wu
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{*pszekely,knoblock,shubhamg,mohsen,bowu*}*@isi.edu*

ABSTRACT

Using today's GIS tools, users without programming expertise are unable to fully exploit the growing amount of geospatial data becoming available because today's tools limit them to displaying data as layers for a region on a map. Fusing the data in more complex ways requires the ability to invoke processing algorithms and to combine the data these algorithms produce in sophisticated ways. Our approach, implemented in a tool called Karma, encapsulates these algorithms as Web services described using semantic models that not only specify the data types for the inputs and outputs, but also specify the relationships between them. Karma semi-automatically builds these models from sample data and then uses these models to provide an easy to use interface that lets users seamlessly implement workflows that combine and process the data in sophisticated ways.

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 Interfaces and Presentation**]: User Interfaces—*Theory and methods*.

General Terms

Information integration, geospatial fusion, data cleaning, semantic modeling.

Keywords

Geospatial fusion, semantic model, source modeling, service modeling, ontology, data cleaning, information integration, RDF generation.

[*]This work is supported in part by Chevron Corp. under the joint project Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

1. INTRODUCTION

There is a tremendous amount of geospatial data available and there are numerous methods for processing, integrating and fusing geospatial sources. However, the ability for users to combine this type of data today is limited to using GIS systems to display the various layers for a region on a single map. If a user wants to combine or fuse the data in novel ways, they have to do this by building specialized applications that are then applied to specific data sources. This approach does not support the need for rapid integration or fusion of geospatial sources and does not support the reuse of the methods for reasoning or fusion of the available sources.

The object of this work is to provide end-users with an interactive tool to enable them to easily perform a wide variety of geospatial fusion tasks. Our focus is on usability, and our target is to enable users who would be comfortable using spreadsheets to perform geospatial data fusion tasks that today would require programming expertise in scripting languages and Web technologies. To succeed, we must provide users support for the complete workflow, starting with importing data, normalizing it, integrating it with other data, invoking data fusion algorithms, visualizing the results and finally publishing new datasets with the fused data.

Our approach is based on our prior work on Karma [18, 19], a general information integration tool that already supports many of the steps required for geospatial data fusion. Karma already supports importing data from a variety of sources including relational databases, spreadsheet and delimited text files, KML and semi-structured Web pages. It provides support for cleaning and normalizing the data and integrating it using data integration operators such as join and union, and for publishing it in a variety of formats including RDF.

The focus of this paper is on how to integrate reasoning and fusion algorithms into the larger information integration workflow required to solve complex problems. Our approach, consistent with the overall Karma approach, is to model the fusion algorithms as services that users can invoke using data from the Karma workspace. Semantics already plays a central role in our approach to information integration. When users import data, Karma semi-automatically builds models of the data according to a user selected ontology. The models that Karma builds are based on models that users built before, when working with data with similar characteristics. Karma proposes models for the data, and users can adjust them to satisfy the particular characteristics

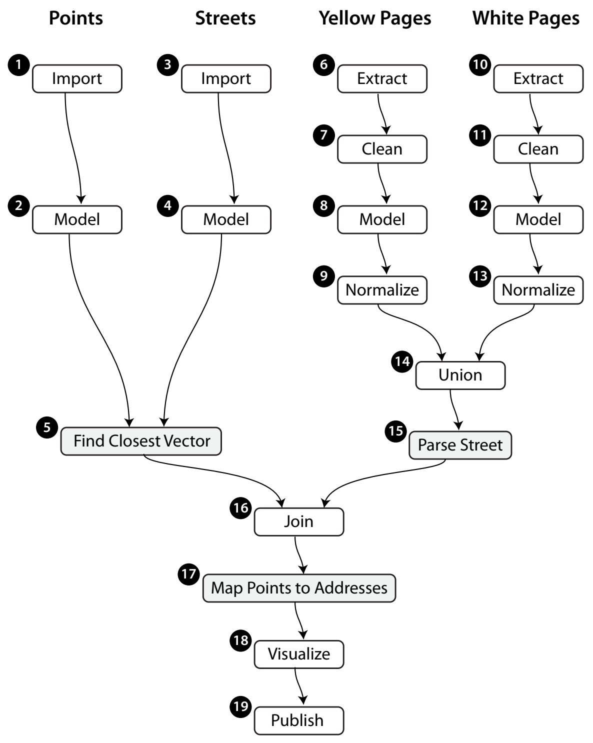


Figure 1: The workflow for associating telephone book information with a building or structure shown in a satellite image.

of the data they are working with. The new work presented here focuses on extending our semi-automatic modeling approach to model services and on the mechanisms needed to invoke the services with the correct data and to integrate the output of the services with the other data.

In the next section we will present a motivating example on linking data from online sources to structures identified in satellite imagery. Then we describe the overall approach and present our methods for modeling sources, modeling services, invoking the fusion algorithms on this data, and then visualizing and/or publishing the results. Next we describe the related work and finally conclude with a discussion of the contributions and future directions.

2. MOTIVATING EXAMPLE

In this section we describe an example that we will use to motivate the remainder of the paper. The general problem that we want to address is how to dynamically combine a set of data sources and fusion algorithms to produce needed results. The specific instance of this problem that we will focus on is how to associate information from the telephone books with buildings or structures visible in satellite imagery [13, 6].

The workflow for solving this task is shown in Figure 1. There are four inputs for the overall task. The first input is `Points`, which are a set of points that correspond to the locations of all of the buildings in the image. These points could be identified manually or extracted from LIDAR data, which can be used to build 3D models of all of the buildings [22]. The second input is `Streets`, which is a labeled road network for the same region. The road network could come from a source such as OpenStreetMap (www.openstreetmap.org) or could be extracted from a raster maps using techniques that we previously developed [2]. The third input is `Yellow Pages`, which is a online source providing yellow page data that gives the details of the businesses in a region. And the fourth input is `White Pages`, which is an online source providing while page data that provides the data on people that live in a given region.

In addition to the data sources, the workflow also requires a set of services for performing various steps in the fusion process. In this particular example, there are three services required. First, there is the service called `FindClosestVector`, which is given a point and set of vectors and it finds the closest vector to a point. In this case, this service is used to find the streets that each building could be located on. If the building is located in the middle of the block, there would only one street, but for a building on a corner there could be several streets. Second, there is a service called `ParseStreet`, which takes a street address and extracts out the street name and building number. Third, there is a service called `MapPointsToAddresses`, which takes as input the building points, the corresponding streets that they are located on, and the possible address numbers for each street and performs a constraint reasoning process to determine the possible addresses of each of the building points [13].

The workflow in Figure 1 shows the overall process for combining the data and the services to fuse the data. This entire workflow can be defined and executed in the Karma framework and the overall process is described in the rest of the paper. The result of this workflow is shown in Figure 11.

3. APPROACH

In this section we describe the details of our approach and illustrate it using the example described in Section 2. Figure 2 shows the ontology that we use in our motivating example. The Business and Person classes will be used to model the information extracted from the Yellow and White pages. The Address class will be used to model the address information extracted from the Yellow and White pages, as well as the street vector data from the Streets source. The Point class is used to model the building locations provided in the Points input source as well as the polylines defining the street vectors in the Streets source. The ontology uses semantically meaningful data types even for the primitive types (e.g., BusinessCategory, BuildingNumber).

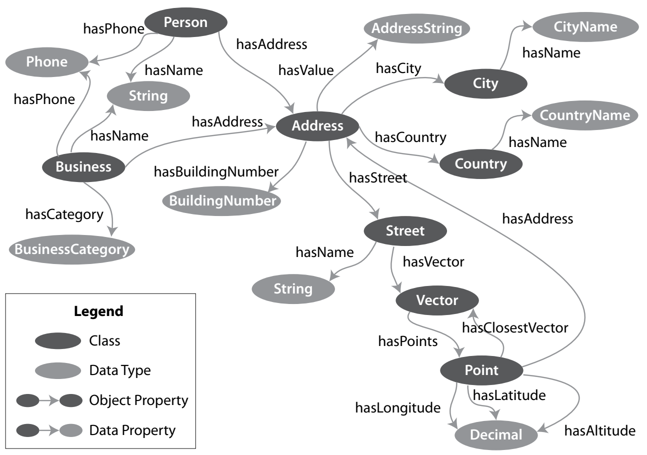
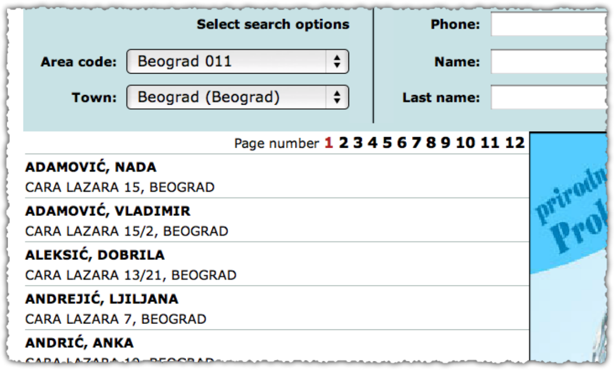


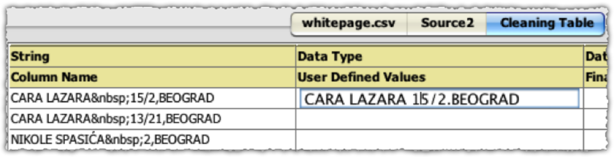
Figure 2: Ontology for our example scenario.

3.1 Importing and Cleaning Sources



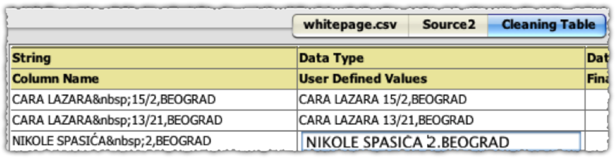
ADAMOVIĆ, NADA
CARA LAZARA 15, BEOGRAD
ADAMOVIĆ, VLADIMIR
CARA LAZARA 15/2, BEOGRAD
ALEKSIĆ, DOBRILA
CARA LAZARA 13/21, BEOGRAD
ANDREJIĆ, LJILJANA
CARA LAZARA 7, BEOGRAD
ANDRIĆ, ANKA

(a) Belgrade online white pages



String	Data Type	Dat
Column Name	User Defined Values	Fin
CARA LAZARA 15/2,BEOGRAD	CARA LAZARA 15/2.BEOGRAD	
CARA LAZARA 13/21,BEOGRAD		
NIKOLE SPASIĆA 2,BEOGRAD		

(b) User provides example of address without "&nbsp".



String	Data Type	Dat
Column Name	User Defined Values	Fin
CARA LAZARA 15/2,BEOGRAD	CARA LAZARA 15/2,BEOGRAD	
CARA LAZARA 13/21,BEOGRAD	CARA LAZARA 13/21,BEOGRAD	
NIKOLE SPASIĆA 2,BEOGRAD	NIKOLE SPASIĆA 2.BEOGRAD	

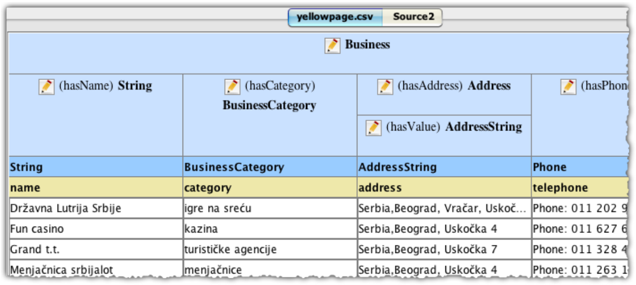
(c) Learned transformation rules are applied to the remaining data, removing all instances of "&nbsp".

Figure 3: Extracting and cleaning data from the Belgrade online white pages

The first step in the geospatial data fusion workflow involves importing into Karma data from the original sources. Figure 3 (a) shows a screen shot of the Belgrade online white pages from where we extract information about possible street addresses. Several research and commercial tools to extract information from Web pages are available [9][1][2] (we used Fetch Agent platform[3]). Users can invoke these extraction tools directly from Karma, which then loads the extracted data and shows it to users as a table.

Raw data, especially data extracted from Web pages is often noisy and needs to be cleaned before it can be processed further. For example, the data extracted from the Belgrade white pages contains "&nbsp;" characters that need to be removed. Consistent with our goal to not require programming expertise from our users, Karma uses a programming-by-example [12] technique for data cleaning and data normalization. To clean data in Karma, users first choose the column of data they want to clean, and then they provide an example of the clean data. For example, in Figure 3(b) the user provided an example of the cleaned address. Karma uses the examples provided to learn a general transformation rule that it then applies to all the values in the selected column. If the results are incorrect, users can provide additional examples to guide Karma to infer an appropriate rule, or they can manually edit the incorrectly cleaned values. Our approach to cleaning is described in more detail elsewhere [19]. Figure 3(c) shows the cleaned data.

[1]http://open.dapper.net/
[2]http://www.needlebase.com/
[3]http://www.fetch.com/product/



Business			
(hasName) String	(hasCategory) BusinessCategory	(hasAddress) Address	(hasPhon
		(hasValue) AddressString	
String	BusinessCategory	AddressString	Phone
name	category	address	telephone
Državna Lutrija Srbije	igre na sreću	Serbia,Beograd, Vračar, Uskoč...	Phone: 011 202 9
Fun casino	kazina	Serbia,Beograd, Uskočka 4	Phone: 011 627 6
Grand t.t.	turističke agencije	Serbia,Beograd, Uskočka 7	Phone: 011 328 4
Menjačnica srbijalot	menjačnice	Serbia,Beograd, Uskočka 4	Phone: 011 263 1

Figure 4: Karma screen to model data extracted from the Belgrade yellow pages.

3.2 Modeling Sources

Once data is clean, it is ready to be modeled (next step in the Karma approach, as illustrated in Figure 1). A distinguishing aspect of our approach is that Karma automatically builds models of imported data according to the ontology that the users have provided. If the models Karma builds are incorrect, it offers menus to enable users to easily fix the incorrect elements. As we describe later, these models play an important role in helping users assemble the workflow to process their data. This help users integrate the data as needed to satisfy the requirements of the services that fuse the data and produce the desired outputs.

Consider Belgrade's online yellow pages data extracted and cleaned similarly to how white pages data was extracted and cleaned. The data modeling process has two parts. The first part is to identify the semantic types (nodes in the ontology) of each column of data. Here we use our CRF based technique [4] that automatically assign labels to sets of data based on learned labelings of previously processed data. We assume that in prior sessions, users trained the system to recognize phone numbers and addresses. As shown in Figure 4, Karma assigned the semantic type Phone to the column labeled TELEPHONE. When Karma sees data for the first time, or when the format for data differs significantly from the formats that Karma was trained on, Karma may not assign the semantic type correctly. In our example, the addresses do not include the country, so Karma did not assign them the AddressString semantic type. In such cases, users can click on the incorrectly assigned type to invoke a menu from where they can choose the correct type. Karma uses the data in the column to re-train its learning algorithms so that they can correctly identify data in the new format in the future.

The second part of the modeling process is designed to identify the relationships among the data columns. For example, we established that the TELEPHONE column contains instances of the type Phone and that the ADDRESS column contains data of type AddressString. However, the model does not yet specify whether the phone number and the address belong to the same entity. They do, but unless explicitly represented, the system cannot reason with this information. Perhaps the phone number is the contact number for a Person associated with the business.

Karma uses the semantic type information of each column to search the ontology graph for links that connect

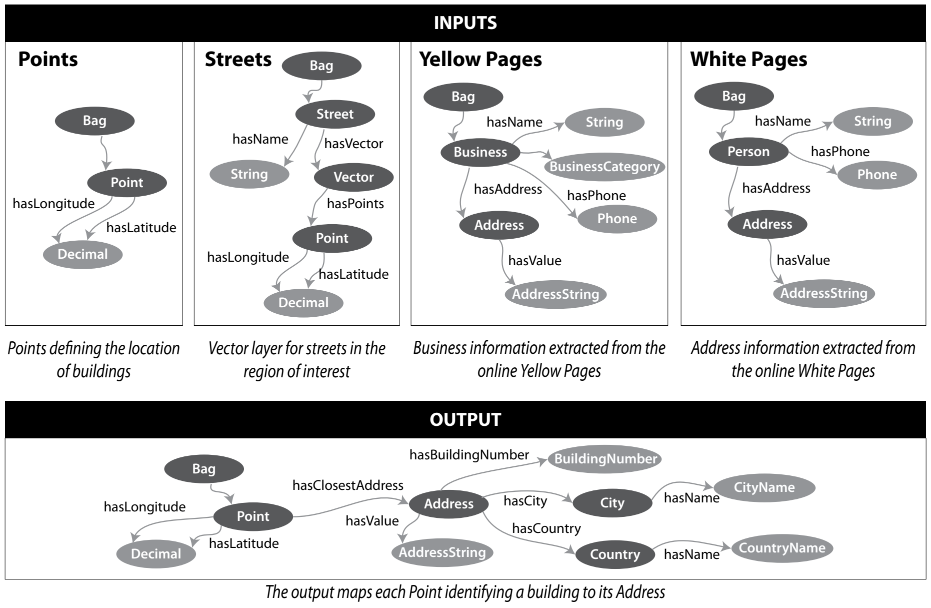


Figure 5: Inputs and output

the corresponding types in the ontology. Karma selects the smallest tree that connects these semantic types and shows it at the top of the data worksheet, as illustrated in Figure 4. In our yellow pages example, the ontology is simple, and Karma correctly identified the relationships among the columns. The smallest tree that connects the semantic types for the yellow pages data has the class Business as its root, as shown in the first row of the figure. The second row shows the properties that connect the Business class to the semantic types. When the ontology or the data are more complex, the tree will have multiple levels. It is also possible that multiple minimal trees exist, or that the correct interpretation of the data is specified by a non-minimal tree. In these cases, users can click on the pencil icons to select from a menu of links originating on the element users click on. For example, if users clicked on the pencil icon above the Phone semantic type, they would get a menu that offers to connect this type to either Business or Person given that in the ontology these are the only two classes that can have phone numbers. We evaluated this modeling approach in a bioinformatics domain using an ontology containing 21 nodes and using 7 data sources containing a total of 39 columns. In that evaluation, fewer than one menu selection per column was needed to obtain the correct model of the data [10].

Figure 5 shows the semantic models of the four data sources used in our scenario (we use the term Bag to represent containers of data). They are all constructed when users load the corresponding data source in Karma. In general, each input source provides only a subset of the data that could be gathered about each entity. For example, even though the ontology specifies that Point can have an hasAltitude property, our Points source only specified latitude and longitude. More importantly, the white and yellow pages sources only provide the AddressString of an Address, but does not break the address into components (building number, street, city, postal code, country). In our approach, we use services to parse AddressString into a structured representation, and to perform the other computation and reasoning steps needed to produce the desired output.

Figure 5 also shows the semantic model of the desired outputs. The task of the next steps in the workflow is to process the inputs to produce a dataset with the output semantic model. The job is challenging because the goal is to connect the Point data in the Points input to the Address data extracted from the yellow and white pages inputs. The Streets input will be used to make the connection.

The next steps in the workflow (Figure 1) are preparation steps to invoke the services. Normalization operations are often required to unify data formats and/or to convert them to the data formats expected by the services. Karma uses the programing-by-example approach illustrated in the data cleaning scenario, so we will not discuss normalization further.

3.3 Models of Reasoning Services

Services are an important component of geospatial data fusion. They can encapsulate relatively simple, but commonly used mathematical computations such as our Find Closest Vector, or can involve sophisticated computations such as our Map Points to Addresses. It has been long recognized that service models are useful to help with discovery, to ensure correct invocation and to automate and hide the technical details of data marshalling and service invocation [16].

Figures 6, 7 and 8 illustrate how we model services in Karma. We think of services as unmaterialized data sources in that specific service invocations with particular inputs materialize a subset of such an unmaterialized data source. Consequently, our approach to service modeling is similar to our approach to data modeling where we model the semantic types of the individual data items, and also model the relationships among them in terms of the ontology graph. The main difference is that we need to distinguish inputs and outputs, and this we do simply by marking elements of the model as either input or output.

Consider Find Closest Vector (Figure 6), an example of a

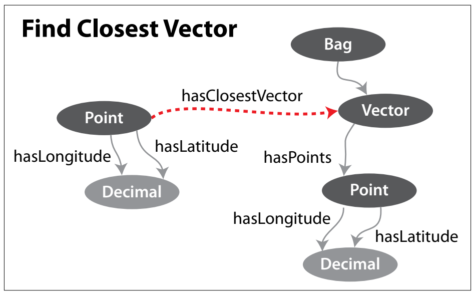


Figure 6: Find the closest Vector from a Point to a set of Vectors.

mathematical computation useful in many scenarios. The input to the service is a Point and a set of polyline Vector. The output is a mapping that maps each point to the possibly empty subset of vectors whose distance to the given point is below a given threshold. The model specifies the data elements needed as input in terms of the ontology. For example, this service needs the latitude and longitude information about the input Point, but does not need the altitude information. In the diagram, the service output is shown in dotted red lines, which in this service is simply the hasClosestVector relationship between Point and Vector. If we think of this service in terms of RDF, the output is a collection of triples of the form (*point* hasClosestVector *vector*).

Parse Address (Figure 7) is an example of a service that returns structured information. This service parses an input string into structured Address objects. The model specifies that this service is able to extract the building number and street name. Because the other properties of Address are not present in the service model, Karma knows that this simple service cannot extract them, and if they were needed, a different service would have to be used.

Map Points to Addresses (Figure 8) is an example of a complex reasoning component. Its input requirements are more complex as it requires points and addresses, and the hasClosestVector relationship between the points representing building locations and the vectors that define the streets where the building could be located. The service formulates the problem of mapping points to addresses as a constraint satisfaction problem: each point must be assigned to an address according to the available building numbers while satisfying assumptions about numbering schemes (e.g., numbers are assigned monotonically, even and odd numbers on opposite sides of a street).

While we expect that the models for many such services would be available in a library, in our work we also want to help users build such models themselves when they want

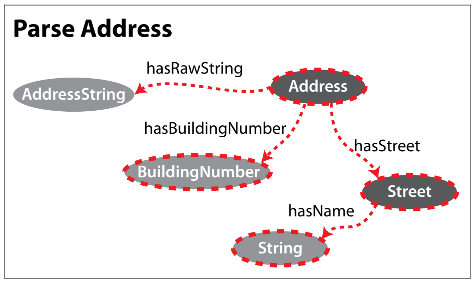


Figure 7: Parses a string into an Address.

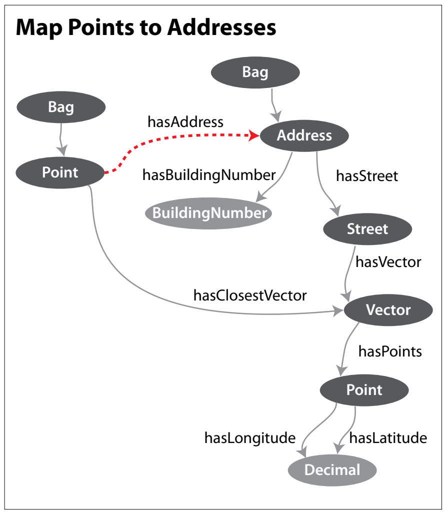
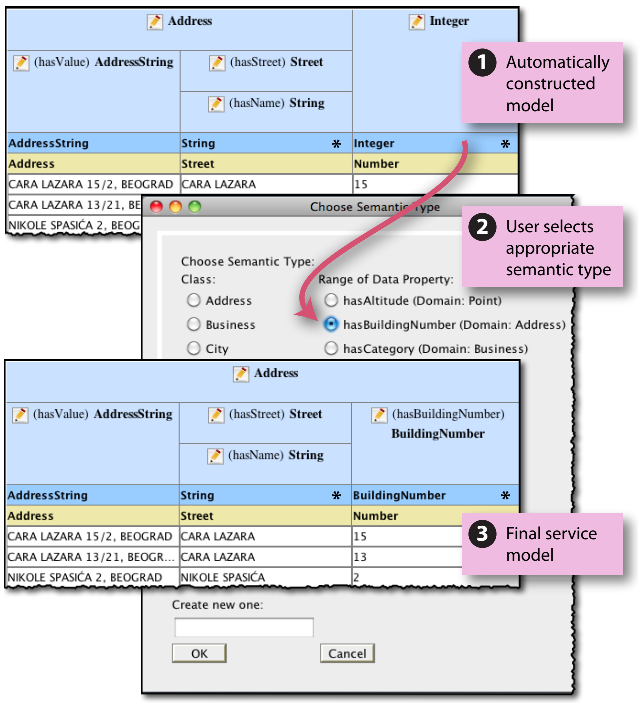


Figure 8: Map each Point to the closest Address.

to use a service that has not yet been modeled. For example, suppose that the Parse Address service exists as a REST service, but Karma has no model of it in its library. As mentioned previously, we think of services as unmaterialized data sources. Consequently, our approach to help users build service models is to materialize a subset of this unmaterialized data source, and use the same approach we use to model data sources.


1 Automatically constructed model
2 User selects appropriate semantic type
3 Final service model
Choose Semantic Type
OK
Cancel

Address		Integer
(hasValue) AddressString	(hasStreet) Street	
	(hasName) String	
AddressString	String *	Integer *
Address	Street	Number
CARA LAZARA 15/2, BEOGRAD	CARA LAZARA	15
CARA LAZARA 13/21, BE		
NIKOLE SPASIĆA 2, BEOG		

Choose Semantic Type:

Class:
- Address
- Business
- City

Range of Data Property:
- hasAltitude (Domain: Point)
- [✓ hasBuildingNumber (Domain: Address)]
- hasCategory (Domain: Business)

Address		
(hasValue) AddressString	(hasStreet) Street	(hasBuildingNumber) BuildingNumber
	(hasName) String	
AddressString	String *	BuildingNumber *
Address	Street	Number
CARA LAZARA 15/2, BEOGRAD	CARA LAZARA	15
CARA LAZARA 13/21, BEOGR...	CARA LAZARA	13
NIKOLE SPASIĆA 2, BEOGRAD	NIKOLE SPASIĆA	2

Create new one:

Figure 9: Modeling the Parse Address service.

For the moment, let us assume that users have access to data sources containing sample inputs for a service and the outputs that the service produces for these inputs. In the future work section we outline an approach for obtaining such data sources automatically. Given these data sources, users would import and model them as described in section 3.1.

Figure 9 illustrates this approach using our Parse Address service. Our service, implemented as a REST service has one input, an address, and two outputs, the building number and the street name. The first column in the table contains the complete address strings, and subsequent columns contain the parsed information, namely the street name and the building number. As shown in Figure 9, step 1, Karma's semantic typing component correctly assigns the semantic type AddressString the ADDRESS. Karma initially assigns the type Integer to the BUILDING NUMBER column as it contains numbers, and assigns the type String to the STREET column as it contains seemingly arbitrary strings. Based on this information, Karma automatically infers that the data source contains Address, infers the relationship to AddressString, and infers that String is the name of the Street. Karma cannot relate the BUILDING NUMBER column to the Address class, so it remains as a top level type. In step 2, the user corrects the mistake by clicking on Integer and selecting BuildingNumber from the menu. At this point Karma infers the correct model in step 3, resulting in precisely the model shown in Figure 7.

This approach works for services such as Parse Address and Find Closest Vector where the inputs and outputs can be represented in a single worksheet. We are investigating extensions to more complex services such as Map Points to Addresses where the inputs and outputs cannot be represented in a single worksheet.

3.4 Data Fusion

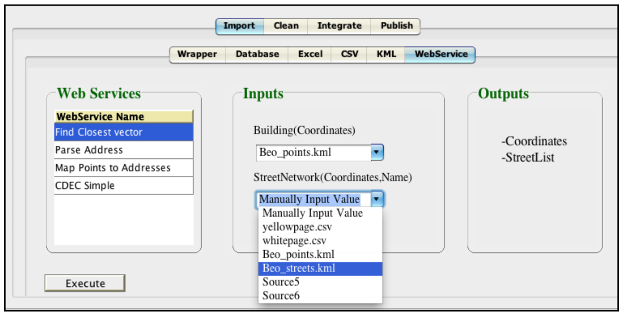


Figure 10: Karma screen showing how users can invoke the Find Closest Vector service.

Karma assists users with data fusion by allowing a user to quickly integrate and invoke services. A significant problem in invoking services is finding the required data to invoke the service and getting that data into the correct format. The capabilities described previously in Karma for extracting data from sources, cleaning and normalizing the data, and integrating the across sources address the data preparation problem. Once the data has been prepared and the service has been defined in Karma, then it is straightforward to invoke the service on the data. The results are then returned as another source that can then be further refined, integrated with other sources, visualized, or published.

Continuing with our sample workflow, in Figure 10 we illustrate how users invoke services in Karma by showing how they can invoke the Find Closest Vector service. The service takes as input a Point and a collection of Vector data. For the Point input, the user provides a source that contains a collection of points, and Karma will iterate the invocation of the service, once for each point.

In general, service invocations produce datasets that need to be joined with other datasets to augment them with needed information before invoking other services. For example, the Find Closest Vector service produces a table that maps points to vectors. Users need to join this table with the Streets table to add the street name column so that they can later be joined with the combined street data from the yellow and white pages to produce the input needed for Map Points to Addresses.

3.5 Visualization and Publication

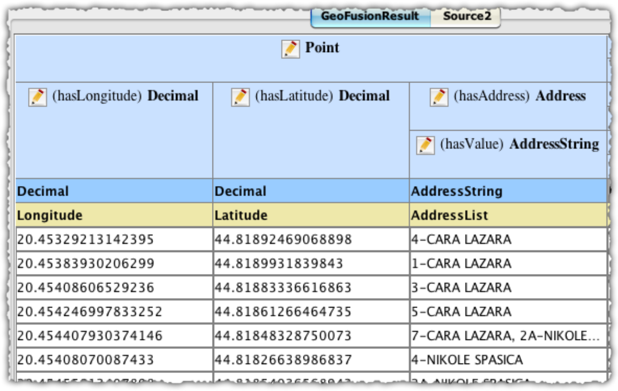
After invoking the Map Points to Addresses service, Karma produces a worksheet containing the fused output, as defined in Figure 5. At this point, users can visualize the information on a map, or publish it in a variety of formats including relational database tables, comma-separated-value files, KML and RDF. When exporting data in RDF, Karma will produce RDF triples using the classes and properties defined in the user's ontology. Karma's RDF export capability is significant as with the click of a button users can produce semantically meaningful RDF triples according to the ontology of their choice.

Figure 11 shows the data for our example scenario visualized on a map: Each building is shown as a pin located at the latitude and longitude given the in the Points input source. The bubbles that appear when users click show the possible addresses as computed by the Map Points to Addresses service. The example in the figure shows two possible addresses, as the building is on a corner, and the service does not have enough information to assign it a unique address.

In general, Karma allows users to visualize any worksheet containing geospatial information on a map. The map shows layers for all worksheets that users have asked to visualize. Furthermore, as the data in the worksheet changes, due to cleaning, normalization, union or join, the map automatically updates to show the latest information.

4. RELATED WORK

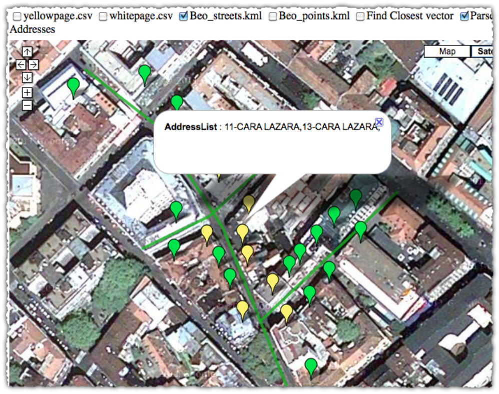
A lot of recent work explores different approaches for the semantic fusion of geospatial data. Much of the work exploits ontologies to attach semantics to geospatial services and data, and then use it to build geospatial reasoning models by chaining the services. Yue et al [20] uses manually generated OWL-S based semantic descriptions for the geospatial web services in a service-oriented architecture (SOA) to enable the automatic discovery, access, and chaining of geospatial Web services. OWL-S semantics addresses the structural interoperability required by the composition of geospatial Web services. In later work [21], they introduce an initial framework for representing, storing and querying geospatial data provenance using Semantic Web technologies. The work done by Di et al [3] requires users to build the service model by manually matching the classes from the ontology to the inputs and outputs of the reasoning services. The eMerges approach [17] employs Semantic Web Services (SWS), allowing it to generate semantically rich geospatial data and perform integration with it. In this approach each data source must be converted into a SWS and the user has to map the sources to integration ontologies manually. All these approaches rely on users to manually model their services and data sources. In our work, Karma offers users a semi-automatic approach to model their geospatial services



GeoFusionResult | Source2

Point		
(hasLongitude) **Decimal**	(hasLatitude) **Decimal**	(hasAddress) **Address**
		(hasValue) **AddressString**
Decimal	**Decimal**	**AddressString**
Longitude	**Latitude**	**AddressList**
20.45329213142395	44.81892469068898	4-CARA LAZARA
20.45383930206299	44.8189931839843	1-CARA LAZARA
20.45408606529236	44.81883336616863	3-CARA LAZARA
20.454246997833252	44.81861266464735	5-CARA LAZARA
20.454407930374146	44.81848328750073	7-CARA LAZARA, 2A-NIKOLE...
20.45408070087433	44.81826638986837	4-NIKOLE SPASICA

Output of the **Map Points to Addresses** *service.*



☐ yellowpage.csv ☐ whitepage.csv ☑ Beo_streets.kml ☐ Beo_points.kml ☐ Find Closest vector ☑ Parse Addresses

AddressList : 11-CARA LAZARA,13-CARA LAZARA,

Visualization of the Points and Streets input data. The Addresses computed by the **Map Points to Addresses** *service shown as bubbles.*

Figure 11: Outputs of the example geospatial workflow: results table and map.

and data sources.

Norton et al. [14] employ Linked Open Services (LOS) [11] to model geospatial services. In LOS, service developers use SPARQL graph patterns to represent inputs and outputs, and RDF is used for content negotiation between the service and its clients. The main idea behind LOS is to define services capable of consuming linked data directly, without marshalling and unmarshalling, and to contribute linked data to linked open data cloud. In contrast to Karma, in which service modeling is semi-automatic, in the LOS approach, service developers must build the service models manually. Additionally, both describing and using LOS services requires expertise in RDF and SPARQL, which our target users do not have. Karma uses a spreadsheet metaphor to show these models to users and to help users build them.

Average Internet users can be assumed to be comfortable working with spreadsheets and tables. Karma exploits this fact and presents a spreadsheet-based user interface to integrate data between different sources. Other approaches such as Google Refine [7] and Google Fusion Tables [5] also use a similar kind of interface and can be compared to Karma in various aspects. Google Refine provides capabilities to import data from various source types such as CSV, XML, etc. and invoking web services. It allows the user to model their sources by aligning the columns to Freebase [4] schema types automatically. It supports geospatial visualization capabilities by letting the user publish his data on a map. Google Fusion Tables also provides similar capabilities, except that it has an advanced geospatial visualization component and allows directly importing KML data. It allows the user to upload large geographic data sets containing street addresses, points, lines, or polygons. It scales large data well by performing the rendering on the server-side and sending the client a collection of small images (tiles) that contain the rendered map. Both the above approaches do provide capabilities to integrate data from different sources but do not exploit any kind of semantics in doing so.

Another interesting approach called GeoDec [15] provides an immersive environment for visualizing the geospatial data making spatiotemporal queries over it to assist decision making. It supports on-the-fly fusion of geospatial data such as vector data, satellite imagery, and raster maps. To address the problem of alignment, it employs a set of techniques to automatically align maps and road vector data on orthorectified imagery [1]. This approach complements the geospatial capabilities of Karma by providing some of the geospatial capabilities that Karma does not support currently. For example, we are currently working on integrating techniques [8] in Karma that can extract the vector data out from raster maps that would then allow the user to use raster map as an input data source to Karma.

[4]http://www.freebase.com/schema

5. DISCUSSION AND FUTURE WORK

This paper described our end-to-end approach to extracting, modeling, and fusing geospatial sources. The key contribution of this work is that it allows end users to quickly and easily fuse a wide variety of geospatial sources. To support the fusion process and to ensure meaningful results, the system builds and maintains a semantic description of the sources and services used. This allows the system to propose meaningful "joins" across the data. In the future will leverage the semantic models to support the automatic composition of sources and services to automatically build fused datasets. Another important advantage of building and maintaining semantic descriptions of sources is that it means that we can produce semantic descriptions of the fused datasets, which in turn can be published and reused.

We are working to apply our modeling approach to the large number of REST services available. We are mining the Web for examples of service invocations in documentation pages, blogs and forums to automatically construct datasets of sample data to invoke services. Using these data sets we can bootstrap the Karma modeling process that has proved successful at modeling data sources, and help our users produce a comprehensive library of service models.

6. REFERENCES

[1] Chen, C.C., Knoblock, C.A., Shahabi, C.: Automatically and accurately conflating raster maps with orthoimagery. Geoinformatica 12, 377–410 (2008)

[2] Chiang, Y.Y., Knoblock, C.A., Shahabi, C., Chen, C.C.: Automatic and accurate extraction of road intersections from raster maps. Geoinformatica 13(2), 121–157 (2008), http://dx.doi.org/10.1007/s10707-008-0046-3

[3] Di, L., Zhao, P., Yang, W., Yue, P.: Ontology-driven automatic geospatial-processing modeling based on web-service chaining. In: Proceedings of the Sixth Annual NASA Earth Science Technology Conference (2006)

[4] Goel, A., Knoblock, C.A., Lerman, K.: Using conditional random fields to exploit token structure and labels for accurate semantic annotation. In: Proceedings of AAAI 2011 (2011)

[5] Gonzalez, H., Halevy, A., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W.: Google fusion tables: data management, integration and collaboration in the cloud. In: Proceedings of the 1st ACM symposium on Cloud computing. pp. 175–180. SoCC '10, ACM, New York, NY, USA (2010)

[6] Gupta, S., Knoblock, C.A.: A framework for integrating and reasoning about geospatial data. In: Extended Abstracts of the Sixth International Conference on Geographic Information Science (GIScience) (2010)

[7] Huynh, D., Mazzocchi, S.: Goole refine. http://code.google.com/p/google-refine/

[8] Knoblock, C.A., Chen, C.C., Chiang, Y.Y., Goel, A., Michelson, M., Shahabi, C.: A general approach to discovering, registering, and extracting features from raster maps. In: Proceedings of the Conference on Document Recognition and Retrieval XVII of SPIE-IS and T Electronic Imaging. vol. 7534 (2010)

[9] Knoblock, C.A., Lerman, K., Minton, S., Muslea, I.: Accurately and reliably extracting data from the web: A machine learning approach. In: Szczepaniak, P.S., Segovia, J., Kacprzyk, J., Zadeh, L.A. (eds.) Intelligent Exploration of the Web, pp. 275–287. Springer-Verlag, Berkeley, CA (2003)

[10] Knoblock, C.A., Szekely, P., Ambite, J.L., Gupta, S., Goel, A., Muslea, M., Lerman, K., , Mallick, P.: Interactively mapping data sources into the semantic web. In: Proceedings of the Workshop on Linked Science (Submitted for review) (2011)

[11] Krummenacher, R., Norton, B., Marte, A.: Towards linked open services and processes. In: Proceedings of the Third future internet conference on Future internet. pp. 68–77. FIS'10, Springer-Verlag, Berlin, Heidelberg (2010)

[12] Lieberman, H.: Your Wish is My Command: Programming by Example. Morgan Kaufmann, San Francisco (2001)

[13] Michalowski, M., Knoblock, C.A.: A constraint satisfaction approach to geospatial reasoning. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05) (2005)

[14] Norton, B., Krummenacher, R.: Geospatial linked open services. In: Proceedings of the Workshop Towards Digital Earth: Search, Discover and Share Geospatial Data 2010 (2010)

[15] Shahabi, C., Banaei-Kashani, F., Khoshgozaran, A., Nocera, L., Xing, S.: Geodec: A framework to visualize and query geospatial data for decision-making. Multimedia, IEEE 17(3), 14 –23 (july-september 2010)

[16] Studer, R., Grimm, S., Abecker, A. (eds.): Semantic Web Services: Concepts, Technologies, and Applications. Springer, Berlin, Heidelberg (2007)

[17] Tanasescu, V., Gugliotta, A., Domingue, J., Villarias, L.G., Davies, R., Rowlatt, M., Richardson, M., Stincic, S.: Geospatial data integration with semantic web services: The emerges approach. In: Scharl, A., Tochtermann, K. (eds.) The Geospatial Web, pp. 247–256. Advanced Information and Knowledge Processing, Springer London (2007)

[18] Tuchinda, R., Szekely, P., Knoblock, C.A.: Building mashups by example. In: Proceedings of the 2008 International Conference on Intelligent User Interface (2008)

[19] Tuchinda, R., Knoblock, C.A., Szekely, P.: Building mashups by demonstration. ACM Transactions on the Web (TWEB) (2011), to appear

[20] Yue, P., Di, L., Yang, W., Yu, G., Zhao, P.: Semantics-based automatic composition of geospatial web service chains. Computers and Geosciences 33(5), 649 – 665 (2007)

[21] Yue, P., Gong, J., Di, L.: Augmenting geospatial data provenance through metadata tracking in geospatial service chaining. Computers and Geosciences 36(3), 270 – 281 (2010)

[22] Zhou, Q.Y., Neumann, U.: Fast and extensible building modeling from airborne lidar data. In: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems. pp. 7:1–7:8. GIS '08, ACM, New York, NY, USA (2008)