Interactively Mapping Data Sources into the Semantic Web

Craig A. Knoblock[1], Pedro Szekely[1], Jose Luis Ambite[1], Shubham Gupta[1], Aman Goel[1], Maria Muslea[1], Kristina Lerman[1], and Parag Mallick[2] ⋆

[1] University of Southern California
Information Sciences Institute and Department of Computer Science
{knoblock,pszekely,ambite,shubhamg,amangoel,mariam,lerman}@isi.edu,

[2] Stanford University
Department of Radiology
paragm@stanford.edu

Abstract. The Linked Open Data continues to grow rapidly, but a limitation of much of the data that is being published is the lack of a semantic description. While there are tools that help users to quickly convert a database into RDF, they do not provide a way to easily map the data into an existing ontology. This paper presents an approach that allows users to interactively map their structured sources into an existing ontology and then use that mapping to generate RDF triples. This approach automatically generates a mapping from the data source into the ontology, but since the precise mapping is sometimes ambiguous, we allow the user to interactively refine the mappings. We implemented this approach in a system called Karma, and demonstrate that the system can map sources into an ontology with minimal user interaction and efficiently generate the corresponding RDF.

1 Introduction

The Linked Open Data cloud contains over 28 billion triples. Many of these triples are published directly from existing databases using tools such as D2R [5], which makes it very easy to convert a large relational database into a corresponding set of triples. This conversion process uses the structure of the data as it is organized in the database, which may not be the most useful structure of the information in RDF. But either way, there is often no explicit semantic

⋆ This research is based upon work supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

description of the database and it requires a significant effort if one wants to do more than simply convert a database into RDF. The result of the ease with which one can publish data into the Linked Open Data is that there is lots of data published as RDF triples and remarkably little in the way of semantic descriptions of much of this data. Over the years a tremendous amount of effort has gone into languages, ontologies, and reasoning engines for the Semantic Web, so the next big challenge for Linked Open Data is how to bring more semantics to Linked Data so that it can be more effectively utilized.

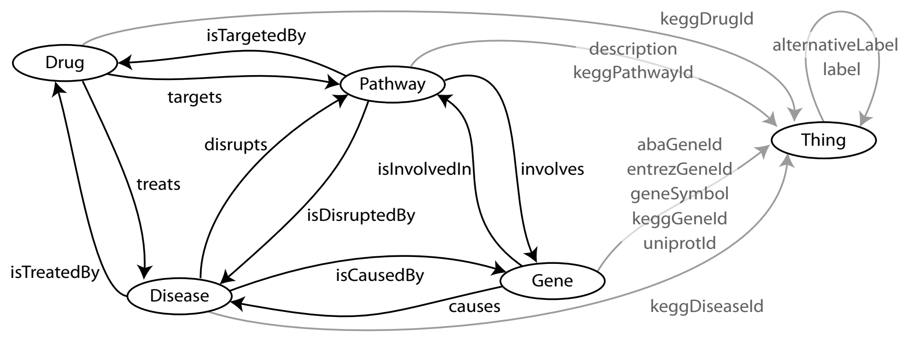
In this paper, we present an approach to interactively mapping data sources into the Linked Open Data cloud with respect to a given ontology. In many cases, the information stored in a database or on a Web page will lack any type of semantic description of the data and will often not even be organized in a way that would make the most sense for representing the data in RDF. The idea behind our approach is to bring the semantics into the conversion process such that the process of converting a data source results in a set of RDF triples that are linked to an ontology. Users can define their own ontology or bring in an existing ontology that may already have been used to describe other related data sources. The advantage of this approach is that it allows the source to be transformed in the process of creating the RDF triples and makes it possible to express the relationship of the resulting RDF triples to other sources.

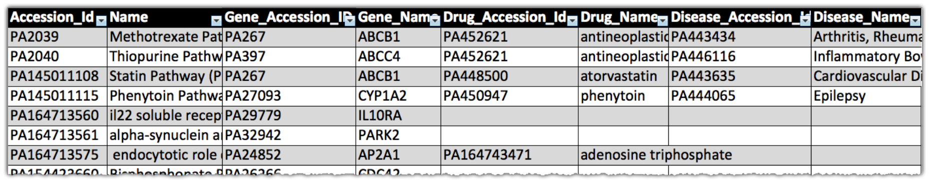
Our approach to mapping data sources into semantically-grounded RDF is based on our past work on performing data integration tasks by example [18]. The contribution of this paper is an interactive approach that allows a user to quickly and easily define the mapping from a source into a domain ontology. There is other work, such as R2R [4] and W3C's R2RML [6], that define languages for specifying mappings between sources, but none of this work addresses the problem of how these mappings are defined. In this work, Karma attempts to automatically build a model of a source, but since there may not be sufficient information to do so, the system allows a user to interactively refine the results. The system first identifies the semantic classes of the data and then builds a model of the relationships between those classes. Thus, as users perform the extraction, cleaning, and integration, they are also building a rich semantic model of the source and the relationship to a domain ontology. In the final step, when users publish the data as RDF, it conforms to the domain ontology.

2 Motivating Example

The bioinformatics community has produced a growing collection of databases with vast amounts of data about diseases, drugs, proteins, genes, etc. Nomenclatures and terminologies proliferate and significant efforts have been undertaken to integrate these sources. One example is the Semantic MediaWiki Linked Data Extension (SMW-LDE) [2], designed to support unified querying, navigation, and visualization through a large collection of neurogenomics-relevant data sources. This effort focused on integrating information from the Allen Brain Atlas

(ABA)[3] with standard neuroscience data sources. Their goal was to “bring ABA, Uniprot[4], KEGG Pathway[5], PharmGKB[6] and Linking Open Drug Data [13] data sets together in order to solve the challenge of finding drugs that target elements within a disease pathway, but are not yet used to treat the disease.”





Accession_Id	Name	Gene_Accession_Id	Gene_Name	Drug_Accession_Id	Drug_Name	Disease_Accession_Id	Disease_Name
PA2039	Methotrexate Pat	PA267	ABCB1	PA452621	antineoplastic	PA443434	Arthritis, Rheuma
PA2040	Thiopurine Pathw	PA397	ABCC4	PA452621	antineoplastic	PA446116	Inflammatory Bo
PA145011108	Statin Pathway (P	PA267	ABCB1	PA448500	atorvastatin	PA443635	Cardiovascular Di
PA145011115	Phenytoin Pathwa	PA27093	CYP1A2	PA450947	phenytoin	PA444065	Epilepsy
PA164713560	il22 soluble recep	PA29779	IL10RA				
PA164713561	alpha-synuclein a	PA32942	PARK2				
PA164713575	endocytotic role	PA24852	AP2A1	PA164743471	adenosine triphosphate		
PA154423660	Bisphosphonate P	PA26266	CDC42				
1, 2	3	4, 7, 8	9	5, 10, 11	12	6, 13, 14	15

```
1.     :Pathway/Accession_Id/PA2039 a :Pathway;
2.        :Accession_Id "PA2039";
3.        :Label "Methotrexate Pathway";
4.        :Involves :Gene/Accession_Id/PA267;
5.        :IsTargetedBy :Drug/Accession_Id/PA452621 ;
6.        :IsDisruptedBy :Disease/Accession_Id/PA443434.
7.      :Gene/Accession_Id/PA267 a :Gene;
8.        :Accession_Id "PA267";
9.        :Label "ABCB1".
10.    :Drug/Accession_Id/PA452621 a :Drug;
11.       :Accession_Id "PA452621";
12.       :Label "antineoplastic agents".
13.    :Disease/Accession_Id/PA443434 a :Disease ;
14.       :Accession_Id "PA443434";
15.       :Label "Arthritis, Rheumatoid" .
```

Fig. 1. Example ontology (black text represents classes and object properties, lighter grey text and grey arrows represent data properties), a structured source that must be aligned to the ontology, and the RDF generated for the first row in the table.

We use the same scenario to illustrate and evaluate our contributions, comparing our results to the published SMW-LDE results. Figure 1 shows the ontology that was used in the SMW-LDE study, shows an example of one of the

[3] http://www.brain-map.org/
[4] http://www.uniprot.org/
[5] http://www.genome.jp/kegg/pathway.html
[6] http://www.pharmgkb.org/

sources that had to be aligned to the ontology and converted to RDF, and shows the RDF generated for the first row in the table. The task is to construct formal, executable mappings between several such tables and the ontology, and then use the mappings to generate RDF for each row.

To define the mapping, we must map each column of data to a class in the ontology. Then, we must select links in the ontology to represent the relationships between the columns of data. For example, the ACCESSION_ID column represents Pathway identifiers, so we map it to the Pathway class. The NAME corresponds to the label property of Pathway, so we map it to the class Thing, and select the label link to represent the relationship between Pathway and Thing (note that in the diagram we show label as a property of Thing, which is inherited by all other classes). The numbers at the bottom of the table indicate the RDF statements that used the data. Several columns give rise to several RDF statements because in addition to recording a fact, they give rise to statements that define relationships among objects. For example, statement 4 records the involves relationship between Pathway and Gene. It is important to note that, in general, these relationships cannot always be computed automatically, and knowledgeable users must resolve ambiguities. For example, the GENE_ACCESSION_ID could be related to Disease via the causes relationship instead or in addition to the involves relationship to Pathway. However, many usage scenarios involve mapping multiple sources to the same ontology, and it becomes possible to learn parts of the mapping and to leverage the structure of the ontology to help users produce the desired mappings quickly.

3 Mapping Data Sources into RDF

Our approach for mapping data sources into RDF is illustrated in Figure 2. The inputs to the process are an OWL ontology of the user's choice, the data sources that the user wants to map to the ontology, and a database of semantic types that the system has learned to recognize based on prior uses of the tool. The main output is a collection of RDF triples that represent the contents of the source, expressed according to the input ontology. The training data for recognizing semantic types is updated during the process to incorporate semantic types learned using the data contained in the sources being mapped.

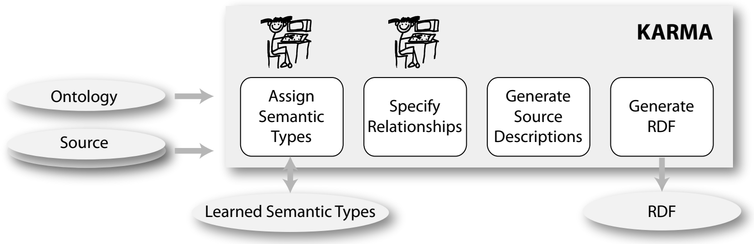


Fig. 2. Karma process to align sources to ontologies and to generate RDF.

Before processing any sources, the system constructs a graph representation of the ontology. Using this graph, the data-source mapping problem consists of the four steps shown in the Figure 2. The first step, *Assign Semantic Types*, involves mapping each column of a source to a node in the graph. This is a user guided process where the system assigns types automatically based on the data values in each column and a set of learned probabilistic models constructed from assignments done in prior sessions. If the semantic type assigned by the system is incorrect, the user can select from a menu the correct node in the graph. The system learns from this assignment and records the learned assignment in its database.

The second step, *Specify Relationships*, involves finding a subgraph that connects these nodes in a meaningful way. In general, there are many subgraphs that connect these nodes, and as mentioned in the previous section it is not possible to automatically select the correct graph. We use a Steiner tree algorithm [19] to compute a minimal subgraph or set of subgraphs that connect the nodes and present it to the user (Figure 4). When the system proposes inappropriate subgraphs, users can select from a menu specific edges to adjust the subgraph. The system computes a new minimal subgraph that incorporates the user-selected edges. After a few iterations a user can guide the system to construct the graph that captures the appropriate relationships.

The third step, *Generate Source Descriptions*, involves generating a logical specification of the mapping. We generate a unary predicate for each internal node in the subgraph, and a binary predicate for each edge.

The fourth and final step, *Generate RDF*, involves using the logical specification to generate RDF for each row in the data source. This is done by plugging-in cell values from each row into their corresponding place in the unary and binary predicates that refer to the cells.

Building the Ontology Graph The central data structure to support the mapping of sources to the ontology is a graph data structure computed from the ontology. The graph (Figure 3) consists of two types of nodes. Class nodes (ovals with black text), for each class defined in the ontology, and data property nodes (ovals with grey text) constructed as follows. For each data property d we construct a collection of nodes $n(d, c_i)$ for each class $c_i \in subclasses(domain(d))$. For example, consider a data property such as Label, whose domain and range is Thing. If we create only a single node corresponding to Thing, then it is not possible to distinguish the labels of drugs, diseases, etc. By creating nodes that encapsulate the more specific domains of such properties, we can reason separately about the labels of drugs, diseases, etc. For example, in Figure 3 we see that the NAME column of our source is mapped to the label node associated with Pathway. For each node $n(d, c_i)$ there is an edge from c_i to $n(d, c_i)$, and for each object property the graph has an edge from the domain to the range of the property. In addition, there is an edge from every subclass to its superclass (not shown in the figure).

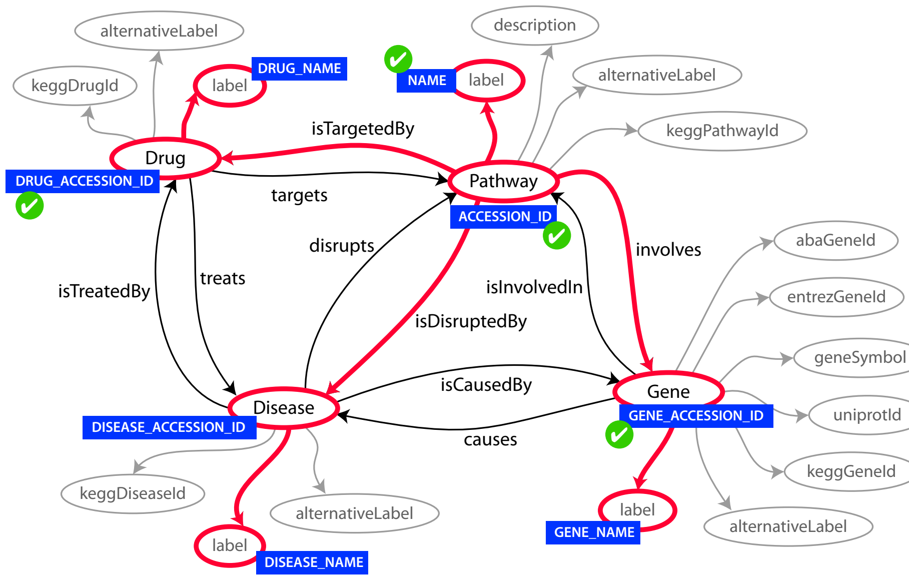
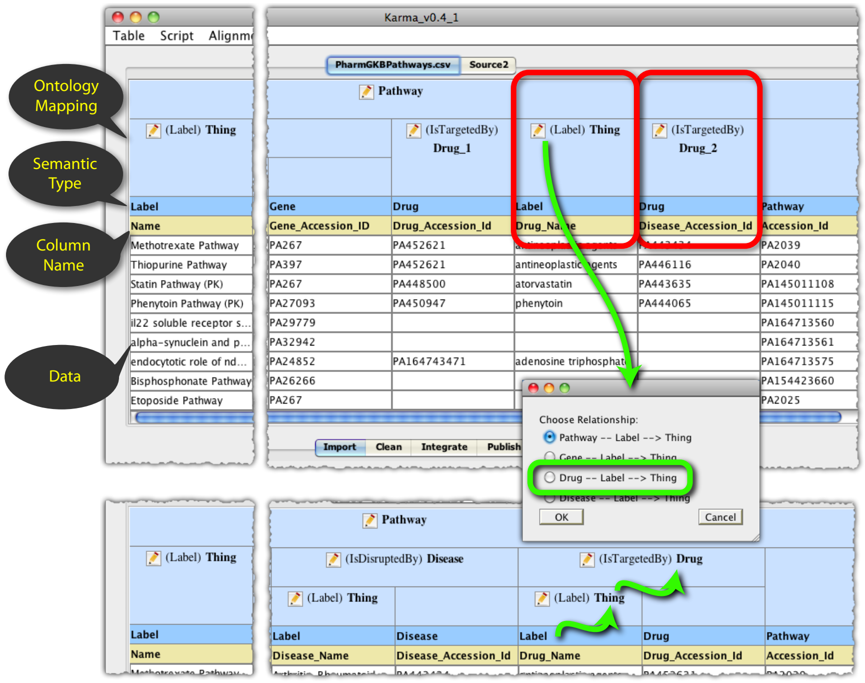


Fig. 3. Internal graph data structure built from an ontology

Inferring the Semantic Types In this step we determine how the columns of a relational table map into the nodes of the ontology graph. Many tables within a database have columns that contain semantically similar data, so the challenge is to learn how to recognize these distinct semantic types, which correspond to the ontological categories (i.e., the classes and data properties) of the ontology. The main identifying characteristics of the semantic types are their token-level syntactic structures. We exploit the hierarchical structure within the values of a column in a database table to accurately identify the semantic type of that column. To do this, we convert each value into a graph of the data value and the corresponding token nodes. Then, we identify the syntactic features from these tokens and train a CRF model [14] to learn the relationships between the various semantic types and the syntactic features of the tokens. We use this trained model to determine the semantic types of new columns of data. The details are described in [10].

Inferring the Relationships The objective of the *Specify Relationships* step is to construct a subgraph that connects all the nodes that correspond to columns in a table. In the example in Figure 3, these are the nodes annotated with a blue rectangle. The red, thicker lines represent the subgraph that correctly encodes the relationships among the columns in the table. For example, the column in the table called ACCESSION_ID represents a Pathway, which is determined by inferring that the semantic type of the ACCESSION_ID is Pathway, and it is targeted by the drug represented in the DRUG_ACCESSION_ID column. An alternative interpretation is that this drug treats the disease that disrupted the pathway. Even though this interpretation is reasonable, it is not the interpretation that correctly describes the data.



(Label) Thing	Pathway				
		(IsTargetedBy) Drug_1	(Label) Thing	(IsTargetedBy) Drug_2	
Label	Gene	Drug	Label	Drug	Pathway
Name	Gene_Accession_ID	Drug_Accession_Id	Drug_Name	Disease_Accession_Id	Accession_Id
Methotrexate Pathway	PA267	PA452621	[illegible]	[illegible]	PA2039
Thiopurine Pathway	PA397	PA452621	antineoplastic agents	PA446116	PA2040
Statin Pathway (PK)	PA267	PA448500	atorvastatin	PA443635	PA145011108
Phenytoin Pathway (PK)	PA27093	PA450947	phenytoin	PA444065	PA145011115
il22 soluble receptor s...	PA29779				PA164713560
alpha-synuclein and p...	PA32942				PA164713561
endocytotic role of nd...	PA24852	PA164743471	adenosine triphosphat		PA164713575
Bisphosphonate Pathway	PA26266				PA154423660
Etoposide Pathway	PA267				PA2025

(Label) Thing	Pathway				
	(IsDisruptedBy) Disease		(IsTargetedBy) Drug		
	(Label) Thing		(Label) Thing		
Label	Label	Disease	Label	Drug	Pathway
Name	Disease_Name	Disease_Accession_Id	Drug_Name	Drug_Accession_Id	Accession_Id

Fig. 4. Karma screen showing the user interaction to change the model of a column from a Pathway label to a Drug label.

We use an iterative, interactive procedure to construct this subgraph. First, the system selects a subgraph that connects the nodes and shows it to the user in the Karma user interface as hierarchical headings over the original data table (Figure 4). We use a Steiner tree algorithm to select the graph, as the minimal subgraph represents the simplest interpretation, and we have found that the simplest explanation is usually the correct one. The top part of Figure 4 shows the Karma GUI with table headings corresponding to the initial graph constructed when the user loads the PharmaGKBPathway data. The two columns highlighted in red illustrate situations where the user will intervene. The first one, for column Drug_Name, illustrates a situation where Karma assigned the incorrect label node to the column. The label node should be associated with Drug instead of Pathway. To select the proper node, the user clicks on the pencil icon to request a menu with the alternative interpretations constructed from the graph. After the user selects the interpretation that specifies that the column represents the label of a drug, Karma adds the selected node and edge to the current subgraph, and recomputes a new subgraph that contains this interpretation. The bottom part of the figure shows that the label now leads to Drug, and furthermore that the drug is the range of the isTargetedBy property of Pathway.

The iterative process continues until the user is satisfied with all the mappings. In the example shown, the system recognized the identifiers under Disease_Accession_Id as representing drugs, not surprising given that the identifiers are very similar to those of Drugs as shown in the Drug_Accession_Id column. The bottom part of the figure represents the final mapping, which corresponds to the subgraph highlighted in Figure 3.

Generation of Source Descriptions We use logical rules to formally define the mapping between the sources and the ontology. Specifically, we use global-local-as-view (GLAV) rules [8] commonly used in data integration [12] and data exchange [1], that is, rules whose antedent and consequent are conjunctive formulas. The (GLAV) rule for the PharmGKBPathways table derived from the chosen subgraph in Figure 3 is:

PharmGKBPathways(NAME,ACCESSION_ID, GENE_ACCESSION_ID, DISEASE_NAME,
GENE_NAME,DISEASE_ACCESSION_ID,DRUG_NAME,DRUG_ACCESSION_ID) →
Pathway(**uri**(ACCESSION_ID)) ^ label(**uri**(ACCESSION_ID), NAME) ^
involves(**uri**(ACCESSION_ID), **uri**(GENE_ACCESSION_ID)) ^
isTargetedBy(**uri**(ACCESSION_ID), **uri**(DRUG_ACCESSION_ID)) ^
isDisruptedBy(**uri**(ACCESSION_ID), **uri**(DISEASE_ACCESSION_ID)) ^
Gene(**uri**(GENE_ACCESSION_ID)) ^ label(**uri**(GENE_ACCESSION_ID), GENE_NAME) ^
Drug(**uri**(DRUG_ACCESSION_ID)) ^ label(**uri**(DRUG_ACCESSION_ID), DRUG_NAME) ^
Disease(**uri**(DISEASE_ACCESSION_ID)) ^
label(**uri**(DISEASE_ACCESSION_ID), DISEASE_NAME)

The rule antecendent is the source relation, or in general a conjunctive query over source relations. The rule consequent specifies how the source data elements map into ontology terms. The rule is generated from the chosen ontology subgraph as follows.

Class nodes generate unary predicates corresponding to classes in the ontology. The **uri** function builds URIs for class instances based on the key/s, or foreign key/s, in the source tables. For example, the Pathway node in Figure 3 generates the predicate Pathway(**uri**(ACCESSION_ID)) because the values in the ACCESSION_ID column represent instances of Pathway. We support class nodes that are not associated with a source column. These correspond to existentially quantified variables in the rule consequent and generate blank nodes[7] in the RDF. For example, assume that the ontology included a Mutation class, where a Gene has a Mutation that causes a Disease, then the corresponding fragment of the rule consequent would be: hasMutation(**uri**(GENE_ACCESSION_ID), **uri**(1)) ^ Mutation(**uri**(1)) ^ causes(**uri**(1), **uri**(DISEASE_ACCESSION_ID)). The index in the **uri** function is used to identify different existentially quantified variables.

Data property nodes generate binary predicates corresponding to data properties in the ontology For example, the label node associated with Pathway in Figure 3 generates the binary predicate label(**uri**(ACCESSION_ID), NAME), specifying that instances of Pathway have the Label data property filled with values from the NAME column.

Edges between class nodes generate binary predicates corresponding to object properties in the ontology. For example, the edge between Pathway and

[7] Actually, we generate unique regular URIs, because we want to support linking (owl:sameAs) into these URIs at a later stage.

Gene in Figure 3 generates the predicate involves(**uri**(ACCESSION_ID), **uri**(GENE_-ACCESSION_ID)).

The resulting GLAV rules can now be used to generate the appropriate RDF for a source in terms of the domain ontology, as in data exchange [1]. Alternatively, the mappings can be interpreted dynamically by a mediator, as in data integration [12]. The mediator would provide an SPARQL endpoint exposing the ontology and perform queries directly over the orginal sources. This latter capability is not yet supported, but we plan to add it in future work.

RDF Generation Our RDF generation algorithm uses the GLAV rules and the sources as inputs, and produces RDF triples as output. Processing a GLAV rule is straightforward. First, the system translates the antecedent of the rule to SQL and evaluates the resulting query over the source/s. Second, it uses the results of the query to materialize the unary and binary predicates in the rule consequent as RDF, ensuring that the correct URIs are generated.

Unary predicates: all unary predicates are of the form $class(uri(X))$ where X is either a column name or an Integer index. If X is a column name, it fetches the corresponding value v from the record being processed and generates a URI concatenating the ontology namespace (an input to the system), the column name, and the value v. For example, the Pathway(**uri**(ACCESSION_ID)) produces the URI :Pathway/Accession_Id/PA2039 when $v = PA2039$. If X is an Integer index, the algorithm produces a blank URI concatenating the ontology prefix with a unique identifier (e.g., `:Gene_1308591004186t1r2_3`). Within the processing of each record, the algorithm stores a map of indices X to generated URIs so that the same URI is used in each occurrence of $uri(X)$. The resulting triple declares the generated URI to be an instance of the corresponding class (e.g., :Pathway/Accession_Id/PA2039 rdf:type :Pathway.)

Binary predicates: all binary predicates are of the form $property(uri(X), O)$ where O is either $uri(Y)$ or a column name. For URIs the system reuses the URIs generated for the corresponding classes, producing for example :Pathway/Accession_Id/PA2039 :involves :Gene/Accession_Id/PA267. For column names, the system fetches the corresponding value from the record being processed and substitutes it for the argument, producing for example :Pathway/Accession_Id/-PA2039 :Label "Methotrexate Pathway".

4 Evaluation

We evaluated our approach by integrating the same set of sources integrated by Becker et al.[2], as described in Section 2. The objective of the evaluation was 1) to assess the ability of our approach to produce equivalent mappings for the sources, i.e., mappings that produce the same RDF for the same sources, and 2) to measure the effort required in our approach to produce the mappings. Becker et al. defined the mappings using R2R, so we used their R2R mapping files as a specification of how data was to be mapped to the ontology. Our objective was to replicate the effect of the 41 R2R mapping rules defined in these files. We were unable to find flat file information for two of the properties found in the

Table 1. Evaluation Results for Mapping the Data Sources using Karma.

Source	Table Name	# Columns	# User Actions		
			Assigning Type	Choosing Path	Total
PharmGKB	Genes	8	8	0	8
	Drugs	3	1	2	3
	Diseases	4	2	3	5
	Pathways	5	3	0	3
ABA	Genes	4	1	1	2
KEGG Pathway	Pathways	6	5	0	5
	Diseases	2	0	1	1
	Genes	1	1	0	1
	Drugs	2	2	1	3
UniProt	Genes	4	1	1	2
		Total: 39	Total: 24	Total: 9	Total: 33
			Avg. # User Actions/Property = 33/39 = 0.85		

R2R mapping files, so in our study we only attempted to replicate the remaining 39 R2R mapping rules (each R2R mapping rule maps a column in our tabular representation). The exact data sets they used are not available so we were unable to compare the resulting RDF triples. We measured effort in Karma by counting the number of user actions (number of menu choices to select correct semantic types or adjust paths in the graph) that the user had to perform. Effort measures for the R2R solution are not available, but appears to be substantial given that the rules are expressed in multiple pages of RDF.

Using Karma we constructed 10 source descriptions that specify mappings equivalent to all but one of the 39 R2R mapping rules. Karma was not able to express a R2R mapping rule for a field that contained a list of alternative names for a gene encoded as a single string of comma-delimited values. The R2R mapping rule used an operator to split the string into multiple values before generating multiple RDF triples for them. We left the list of alternative names as a single value, but in future work we plan to support this type of normalization as well.

Table 1 shows the number of actions required to map all the data sources. The Assigning Type column shows the number of times we had to select the correct semantic type from a menu, and the Choosing Path column shows the number of times we had to select an graph path using a menu (see Figure 4). The total number of user actions was 33, less than one per R2R mapping rule (0.85 per rule), a small effort compared to writing R2R mapping rules in RDF.

5 Related Work

There is significant work on both schema alignment and ontology alignment [16, 17], but much of that work produces simpler alignments, such as relating one single class to another, and often relies only on the structure of the schema/ontology, but not the data. Doan et al [7] use both the structure and data, but their resulting alignments are coarser than ours. In this paper, we produce rich semantic descriptions of sources in terms of a domain ontology, which can be used for data exchange [1] and integration [12]. Specifically, we can transform the data in a source into an semanticaly-enriched RDF that conforms to a given ontology.

Schema matching techniques have also be used to identity the semantic types of columns by comparing them with labeled columns [7]. However, often the format of data in the two columns is different, making it difficult for the schema matching algorithm to recognize a semantically similar column. Our method, on the other hand, learns a general and flexible model of the semantic types from multiple columns of data and aggregates predictions for all field labels to get the label for the whole column. Another approach [15] is to learn regular expression-like rules for data in each column and use these expressions to recognize new examples. Our CRF approach [11] improves over the regular expression approach by better handling variations in formats.

The most closely related work is R2R [4], which provides a language for defining the precise relationships across two ontologies. R2R complements the work on D2R [5] in that once a source has been mapped into RDF triples, R2R provides an approach to describing the relationships between sources. The mappings that we learn here could also be performed in R2R by first using D2R to map a source into RDF and then using the R2R rule language to define the mapping from the D2R triples to another ontology. However, there are two important differences with R2R. First, with R2R a user has to manually write the rules to express the mappings between sources. This paper presents an approach to interactively generate these mappings. Second, the way a source is organized in a database may be quite different than the best way to express the source data as RDF triples and our approach allows a user to rapidly and interactively restructure a source to map it into Linked Open Data.

There has also been a great deal of work in the bioinformatics community on integrating data sources using the Semantic Web, such as TAMBIS [9] and Bio2RDF [3]. TAMBIS followed a mediator approach that provided integrated access to the available bioinformatics sources by linking all of the sources to an ontology. Bio2RDF first converts the data into RDF and then provides integrated access to the data by adding links between the RDF representation of the sources. Karma focuses on ease of use facilitating and speeding up the mapping of the sources into a domain ontology to support the integration across sources.

6 Discussion

A critical challenge of the Linked Data cloud is understanding the semantics of the data that users are publishing to the cloud. Currently, users are linking their information at the entity level, but to provide deeper integration of the available data, we also need semantic descriptions in terms of shared ontologies. In this paper we presented a semi-automated approach to building the mappings from a source to a domain ontology. The system attempts to automate as much of the source modeling as possible and relies on the user to help disambiguate the source descriptions when there is insufficient information to fully automate the modeling task. We focused on building descriptions of structured sources to generate RDF. Often sources require complex cleaning and transformation operations on the data as part of the mapping. We plan to extend Karma's interface to express these operations and to include them in the source descriptions. We also plan to support a SPARQL endpoint that generates RDF data on demand.

References

1. Arenas, M., Barcelo, P., Libkin, L., Murlak, F.: Relational and XML Data Exchange. Morgan & Claypool, San Rafael, CA (2010)
2. Becker, C., Bizer, C., Erdmann, M., Greaves, M.: Extending smw+ with a linked data integration framework. In: 9th International Semantic Web Conference (ISWC2010) (November 2010)
3. Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. Journal of Biomedical Informatics 41(5), 706–716 (Oct 2008)
4. Bizer, C., Schultz, A.: The R2R Framework: Publishing and Discovering Mappings on the Web. In: First International Workshop on Consuming Linked Data (COLD2010) (2010)
5. Bizer, C., Cyganiak, R.: D2R Server–publishing relational databases on the semantic web. In: Poster at the 5th International Semantic Web Conference (2006)
6. Das, S., Sundara, S., Cyganiak, R.: R2rml: Rdb to rdf mapping language, w3c working draft, 24 march 2011. http://www.w3.org/TR/r2rml/ (2011)
7. Doan, A., Domingos, P., Levy, A.Y.: Learning source descriptions for data integration. WebDB 81-86 (2000)
8. Friedman, M., Levy, A.Y., Millstein, T.D.: Navigational plans for data integration. In: Proceedings of the 16th National Conference on Artificial Intelligence (AAAI). pp. 67–73 (1999)
9. Goble, C., Stevens, R., Ng, G., Bechhofer, S., Paton, N., Baker, P., Peim, M., Brass, A.: Transparent Access to Multiple Bioinformatics Information Sources. IBM Systems Journal Special Issue on Deep Computing for the Life Sciences 40(2), 532 – 552 (2001)
10. Goel, A., Knoblock, C.A., Lerman, K.: Using conditional random fields to exploit token structure and labels for accurate semantic annotation. In: Proceedings of the 25th National Conference on Artificial Intelligence (AAAI-11). San Francisco, CA (2011)
11. 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)
12. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10(4), 270–294 (2001)
13. Jentzsch, A., Andersson, B., Hassanzadeh, O., Stephens, S., Bizer, C.: Enabling tailored therapeutics with linked data. In: Proceedings of the WWW2009 workshop on Linked Data on the Web (LDOW2009) (2009)
14. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 282–289 (2001)
15. Lerman, K., Plangrasopchok, A., Knoblock, C.A.: Semantic labeling of online information sources. IJSWIS, special issue on Ontology Matching (2006)
16. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal 10(4) (Dec 2001)
17. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal on Data Semantics IV 3730, 146–171 (2005)
18. Tuchinda, R., Knoblock, C.A., Szekely, P.: Building mashups by demonstration. ACM Transactions on the Web (TWEB) 5(3) (2011)
19. Winter, P.: Steiner problem in networks - a survey. Networks 17, 129–167 (1987)