Leveraging Linked Data to Infer Semantic Relations within Structured Sources

Mohsen Taheriyan[1], Craig A. Knoblock[1], Pedro Szekely[1], José Luis Ambite[1], and Yinyi Chen[2]

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

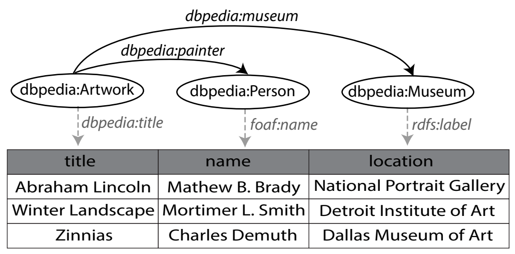
[2] University of Southern California
Department of Computer Science
yinyic@usc.edu

Abstract. Information sources such as spreadsheets and databases contain a vast amount of structured data. Understanding the semantics of this information is essential to automate searching and integrating it. Semantic models capture the intended meaning of data sources by mapping them to the concepts and relationships defined by a domain ontology. Most of the effort to automatically build semantic models is focused on labeling the data fields with ontology classes and/or properties, e.g., annotating the first column of a table with dbpedia:Person and the second one with dbpedia:Film. However, a precise semantic model needs to explicitly represent the relationships too, e.g., stating that dbpedia:director is the relation between the first and second column. In this paper, we present a novel approach that leverages the small graph patterns occurring in the Linked Open Data (LOD) to automatically infer the semantic relations within a given data source assuming that the source attributes are already annotated with semantic labels. We evaluated our approach on a dataset of museum sources using the linked data published by Smithsonian American Art Museum as background knowledge. Mining only patterns of length one and two, our method achieves an average precision of 78% and recall of 70% in inferring the relationships included in the semantic models associated with data sources.

Keywords: semantic model, semantic relation, linked data, semantic label, semantic web

1 Introduction

Information sources such as relational databases, spreadsheets, XML, JSON, and Web tables contain a significant amount of structured data. Given this large amount of data available, we would like to be able to easily search over this information or integrate different pieces of information. Understanding the semantics of data sources enables us to provide a richer search experience and automate the data integration task. In the Semantic Web, we can represent



title	name	location
Abraham Lincoln	Mathew B. Brady	National Portrait Gallery
Winter Landscape	Mortimer L. Smith	Detroit Institute of Art
Zinnias	Charles Demuth	Dallas Museum of Art

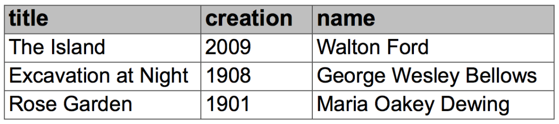
Fig. 1: The semantic model of a sample source containing data of some paintings

the semantics of data by mapping it to the concepts and relationships defined by a domain ontology. In this paper, we use a graphical representation called *semantic model* to show the mapping between a data source and an ontology. In a semantic model, the nodes are ontology classes and the links are ontology properties. Figure 1 depicts a semantic model for a sample data source including information about some paintings. This model explicitly represents the meaning of the data by mapping the source to the DBpedia ontology[3] and the FOAF ontology.[4] Knowing this semantic model enables us to easily transform the data in the table to RDF and publish it on the Web.

The process of creating a semantic model for a source involves two steps: (1) *semantic labeling*, and (2) establishing the *semantic relations*. In semantic labeling, each data field, or *source attribute*, is labeled with a *semantic type*, a class or/and a data property of the domain ontology. In our example in Figure 1, the semantic types of the first, second, and third columns are *title* of *Artwork*, *name* of *Person*, and *label* of *Museum* respectively. However, a semantic model that only includes the semantic labels does not precisely represent the implicit meaning of the data because it is not telling us how the source attributes are related to each other. In our example, a *Person* could be the *owner*, *painter*, or *sculptor* of an *Artwork*, but in the context of the given source, only *painter* correctly interprets the relationship between *Artwork* and *Person*. To build a semantic model that fully recovers the semantics of a data source, we need a second step that determines the semantic relations between the source attributes in terms of the properties in the ontology.

There has been much effort to automatically map data sources to ontologies [4, 5, 7–11, 15, 16], but most focus on semantic labeling or are very limited in automatically inferring the relationships. Our goal is to construct semantic models that not only include the semantic types of the source attributes, but also describe the relationships between them. Inferring the relationships between the attributes is not a trivial task even when the correct semantic labels are given. There might be multiple paths connecting two classes in the ontology and without additional context, we do not know which one characterizes the intended meaning of the data. This work focuses on the second step of building semantic

[3] `http://dbpedia.org/ontology`
[4] `http://xmlns.com/foaf/spec`



title	creation	name
The Island	2009	Walton Ford
Excavation at Night	1908	George Wesley Bellows
Rose Garden	1901	Maria Oakey Dewing

Fig. 2: Sample data from the Crystal Bridges Museum of American Art

models. That is, we assume that source attributes are already annotated with semantic labels and our work focuses on learning the relationships.

We present a novel approach that exploits the knowledge from the domain ontology and the Linked Open Data (LOD) to automatically infer the semantic relations within a given data source. The LOD cloud contains a vast amount of semantic data that can be used to learn how instances of different classes are linked to each other. The main contribution of our work is exploiting the graph patterns occurring in the linked data to disambiguate the relationships between the source attributes. First, we use SPARQL to extract graph patterns with different lengths occurring in the linked data. Next, we combine these patterns into one graph and expand the resulting graph using the paths inferred from the domain ontology. Then, we introduce a search algorithm that explores the graph starting from the semantic labels of the source and heuristically finds the top k semantic models connecting all the labels.

We evaluated our approach on a dataset of well-known museum sources modeled using the CIDOC-CRM[5] ontology along with some other known vocabularies including a total of 147 classes and 409 properties. We used the linked data published by the Smithsonian American Art Museum[6] as the background knowledge. When we used only patterns of length one, our method achieved an average precision of 65% and recall of 55% in inferring the relationships included in the semantic models associated with data sources. Once we added patterns of length two, the precision and recall improved to 78% and 70%. Our evaluation shows that the longer the patterns extracted from the linked data, the more accurate semantic models are generated.

2 Motivating Example

In this section, we provide an example to explain the problem of inferring semantic relations within structured sources. In this example, we want to model a data source using the CIDOC-CRM ontology and then use the created semantic model to publish the source data as RDF. This source is a table containing information about artworks in the Crystal Bridges Museum of American Art[7] (Figure 2). We formally write the signature of this source as *s(title, creation, name)* where *s* is the name of the source and *title*, *creation*, and *name* are the names of the source attributes (columns).

[5] `http://www.cidoc-crm.org`
[6] `http://americanart.si.edu/collections/search/lod/about`
[7] `http://crystalbridges.org`

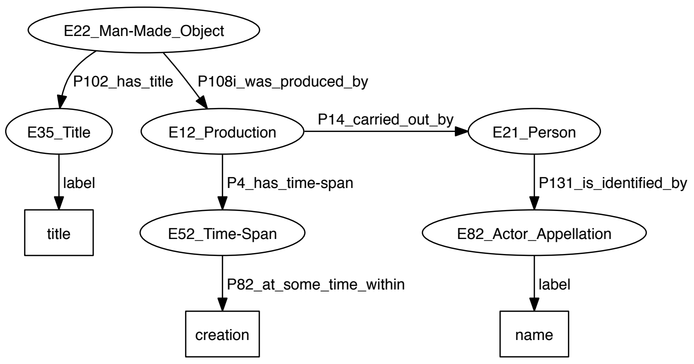


Fig. 3: The semantic model of the source *s*

To model this source, we first need to label the source attributes. This task involves assigning a semantic type to each source attribute. We formally define a semantic type to be a pair consisting of a domain class and one of its data properties ⟨*class_uri,property_uri*⟩ [7]. In this example, the correct semantic types for the columns *title*, *creation*, and *name* are ⟨*E35_Title,label*⟩, ⟨*E52_Time-Span, P82_at_some_time_within*⟩, and ⟨*E82_Actor_Appellation,label*⟩. Various techniques can be employed to automate the labeling task, nonetheless, in this work, we assume that the labeling step is already done and we concentrate on inferring the semantic relations.

Figure 3 shows the correct semantic model of the source *s*. As we can see in the figure, none of the semantic types corresponding to the source columns are directly connected to each other, which makes the problem of finding the correct semantic model more complex. There are many paths in the CIDOC-CRM ontology connecting the assigned labels. For instance, we can use the classes *E39_Actor* and *E67_Birth* to relate the semantic types *E82_Actor_Appellation* and *E52_Time-Span*:

(*E39_Actor*, *P1_is_identified_by*, *E21_Actor_Appellation*)
(*E39_Actor*, *P98i_was_born*, *E67_Birth*)
(*E67_Birth*, *P4_has_time-span*, *E52_Time-Span*)

However, this way of modeling does not correctly represent the semantics of this particular data source. In general, the ontology defines a large space of possible semantic models and without additional knowledge, we do not know which one is a correct interpretation of the data.

Now assume that we have access to a repository of linked data containing the RDF triples published by some other museums. We can exploit this linked data to bias the search space to prefer those models that are used for related source. Once we have identified the semantic types of the source attributes, we can search the linked data to find the frequent patterns connecting the corresponding classes. For example, by querying the linked data, we find out that *P131_is_identified_by* is more popular than *P1_is_identified_by* to connect instances of *E82_Actor_Appellation* and instances of *E21_Person*, and this makes

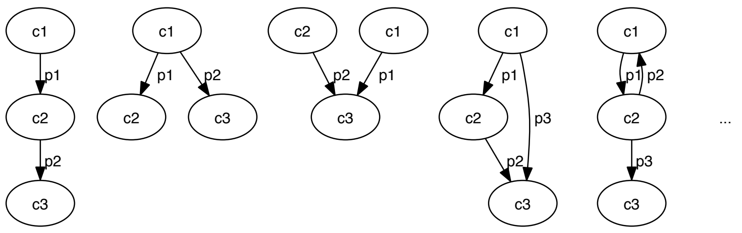


Fig. 4: Sample graph patterns connecting the classes c_1, c_2, and c_3 using the ontology properties p_1, p_2, and p_3

sense when we investigate the definitions of these two properties in the ontology. The property *P1_is_identified_by* describes the naming or identification of any real world item by a name or any other identifier, and *P131_is_identified_by* is a specialization of *P1_is_identified_by* that identifies a name used specifically to identify an instance of *E39_Actor* (superclass of *E21_Person*). We can query the linked data to find longer paths between entities. For instance, by inspecting the paths with length two between the instances of *E22_Man-Made_Object* and *E21_Person*, we observe that the best way to connect these two classes is through the path: *E22_Man-Made_Object* $\xrightarrow{P108i\_was\_produced\_by}$ *E12_Production* $\xrightarrow{P14\_is\_carried\_out\_by}$ *E21_Person*.

3 Inferring Semantic Relations

In this section, we explain our approach to automatically deduce the attribute relationships within a data source. The input to our approach are the domain ontology, a repository of linked data in the same domain, and a data source whose attributes are already labeled with semantic types. The output is a semantic model expressing how the assigned labels are connected.

3.1 Extracting Patterns from Linked Open Data

The Linked Open Data (LOD) includes a vast and growing collection of semantic content published by various data providers in many domains. When modeling a source in a particular domain, we can exploit the linked data published in that domain to hypothesize attribute relationships within the source. We assume that the source attributes are labeled with ⟨*class_uri,property_uri*⟩ pairs.

We use SPARQL to query the linked data in order to extract the graph patterns connecting the instances of the classes corresponding to the semantic types. Each pattern consists of some nodes and links. The nodes correspond to ontology classes and the links correspond to ontology properties. Suppose that we want to find the patterns connecting three classes c_1, c_2, and c_3. Figure 4 exemplifies some of the possible patterns to connect these classes. Depending on the structure of the domain ontology, there might be a large number of possible patterns for any given number of ontology classes. For simplicity, in this paper, we only extract the tree patterns, the patterns in which the number of properties

Algorithm 1 Construct Graph G

```
    Input: LOD Patterns, Semantic Types, Domain Ontology
    Output: Graph $G$
    ▷ Add LOD patterns
 1: sort the patterns descending based on their length
 2: exclude the patterns contained in longer patterns
 3: merge the nodes and links of the remaining patterns into $G$
    ▷ Add Semantic Types
 4: for each semantic type ⟨class_uri,property_uri⟩ do
 5:     add the class to the graph if it does not exist in $G$
 6: end for
    ▷ Add Ontology Paths
 7: for each pair of classes $c_i$ and $c_j$ in $G$ do
 8:     find the directed and inherited properties between $c_i$ and $c_j$ in the ontology
 9:     add the properties that do not exist in $G$
10: end for
    return $G$
```

is exactly one less than the number of classes (the first three patters in Figure 4). That said, we define the length of a pattern as the number of links (ontology properties) in a pattern. For example, a pattern with length one is in the form of $c_1 \xrightarrow{p} c_2$ indicating that at least one instance of the class c_1 is connected to an instance of the class c_2 with the property p. The following SPARQL query extracts the patterns with length one and their frequencies from the linked data:

```
SELECT DISTINCT ?c1 ?p ?c2 (COUNT(*) as ?count)
WHERE {
   ?x ?p ?y.
   ?x rdf:type ?c1.
   ?y rdf:type ?c2.
   FILTER (?x != ?y).}
GROUP BY ?c1 ?p ?c2
ORDER BY DESC(?count);
```

Querying a triple store containing a huge amount of linked data to mine long patterns is not efficient. In our experiments, we only extracted the patterns with length one and two. We will see later that even small graph patterns provide enough evidence to infer rich semantic models.

3.2 Merging LOD Patterns into a Graph

Once we extracted the LOD patterns, we combine them into a graph G that will be used to infer the semantic models. Building the graph has three parts: (1) adding the LOD patterns, (2) adding the semantic labels assigned to the source attributes, and (3) expanding the graph with the paths inferred from the ontology. Algorithm 1 shows the pseudocode of building the graph.

The graph G is a weighted directed graph in which nodes correspond to ontology classes and links correspond to ontology properties. The algorithm to construct the graph is straightforward. However, we adopt a subtle approach to weight the links. We assign a much lower weight to the links added from the LOD patterns comparing to the links added from the ontology. Since we are generating minimum-cost models in the next section, this weighting strategy gives more priority to the links used more frequently in the linked data. The weight of the links coming from the LOD patterns has an inverse relation with the frequency of the patterns.

The other important feature of the links in the graph is their *tags*. We assign an identifier to each pattern added to the graph and annotate the links with the identifiers of the supporting patterns. Suppose that we are adding two patterns $m_1 : c_1 \xrightarrow{p_1} c_2 \xrightarrow{p_2} c_3$ and $m2 : c_1 \xrightarrow{p_1} c_2 \xrightarrow{p_3} c_4$ to G. The link p_1 from c_1 to c_2 will be tagged with $\{m_1, m_2\}$, the link p_2 from c_2 to c_3 will have only $\{m_1\}$ as its tag set, and the link p_3 from c_2 to c_4 will be tagged with $\{m_2\}$. We use the link tags later to prioritize the models containing larger segments from the LOD patterns.

3.3 Generating and Ranking Semantic Models

The final part of our approach is to compute the semantic models from the graph. We map the semantic types to the nodes of the graph and then find the top k minimal trees connecting those nodes. The algorithm that finds the top k trees is a customized version of the BANKS algorithms [1]. It creates one iterator for each of the nodes corresponding to the semantic types, and then the iterators follow the incoming links to reach a common ancestor.

The BANKS algorithm uses the iterator's distance to its starting point to decide which link should be followed next. Because our weights have an inverse relation with their popularity, the algorithm prefers more frequent links. However, selecting more popular links does not always yield the correct semantic model. The coherence of the patterns is another important factor that we need to consider. Thus, we use a heuristic that prefers the links that are parts of the same pattern even if they have higher weights. Suppose that $m_1 : c_1 \xrightarrow{p_1} c_2$ and $m_2 : c_1 \xrightarrow{p_2} c_2 \xrightarrow{p_3} c_3$ are the only patterns used to build the graph G, and the weight of the link p_2 is higher than p_1. Assume that c_1 and c_3 are the semantic labels. The algorithm creates two iterators, one starting from c_1 and one from c_3. The iterator that starts from c_3 reaches c_2 by following the incoming link $c_2 \xrightarrow{p_3} c_3$. At this point, it analyzes the incoming links of c_2 and although p_1 has lower weight, it first chooses p_2 to traverse next. This is because p_2 is part of the pattern m_2 which includes the previously traversed link p_3.

Once we computed top k trees, we rank them first based on their coherence and then their cost (sum of the weights of the links). The coherence metric gives priority to the models that contain longer patterns. For example, a model that includes one pattern with length 3 will be ranked higher than a model including two patterns with length 2, and the latter in turn will be preferred over a model with only one pattern with length 2.

4 Evaluation

To evaluate our approach, we used a dataset of 29 museum data sources in CSV, XML, or JSON format containing data from different art museums in the US. The total number of attributes for this dataset was 418 (on average 14 attributes per source). We applied our approach on this dataset to find the

Table 1: The evaluation dataset

Evaluation Dataset	
number of sources	29
number of attributes	418
number of classes in the ontologies	147
number of properties in the ontologies	409
number of nodes in the gold-standard models	812
number of links in the gold-standard models	785

candidate semantic models for each source and then compared the first ranked models with the gold standard models created manually by an expert in CIDOC Conceptual Reference Model (CIDOC-CRM). Table 1 shows more details of the evaluation dataset. The dataset including the sources, the domain ontologies, and the gold standard models is available on GitHub.[8]. The source code of our approach is integrated into Karma which is available as open source.[9]

The linked data that we used as the background knowledge is the RDF data published by the Smithsonian American Art Museum. The museum has made use of the CIDOC-CRM to map out the concepts and relationships that exist within the artwork collection. This repository includes more than 3 million triples (3,398,350). We injected the data into a Virtuoso triple store and then used SPARQL to extract patterns of length one and two. There were 68 distinct patterns with length one (two nodes and one link) and 634 distinct patterns with length two (three nodes and two links).

We assumed that the correct semantic labels for the source attributes are known. The goal was to see how well our approach learns the attribute relationships having the correct semantic types. We performed three experiments. First, we only used the domain ontology to build a graph on top of the semantic labels and then computed the semantic model connecting those labels. In the second experiment, we took into account the patterns with length one extracted from the linked data, and in the third experiment, we used the patterns of both length one and two. For each source, we computed top 10 candidate semantic models and compared the first one with the correct model in the gold standard set.

We measured the accuracy of the computed semantic models by comparing them with the gold standard models in terms of *precision* and *recall*. Assuming that the correct semantic model of the source s is sm and the semantic model learned by our approach is sm', we define the precision and recall as:

$$precision = \frac{rel(sm) \cap rel(sm')}{rel(sm')}, \quad recall = \frac{rel(sm) \cap rel(sm')}{rel(sm)}$$

where $rel(sm)$ is the set of triples (u, v, e) in which e is a link from the node u to the node v in the semantic model sm. Consider the semantic model in Figure 3, *rel(sm)*=$\{$*(E22_Man-Made_Object, P108i_was_produced_by, E12_Production)*, $\cdots\}$.

[8] `https://github.com/taheriyan/cold-2015`

[9] `https://github.com/usc-isi-i2/Web-Karma`

Table 2: The evaluation results

background knowledge	**precision**	**recall**	**time (s)**
domain ontology	0.07	0.05	0.17
domain ontology + patterns of length one	0.65	0.55	0.75
domain ontology + patterns of length one and two	0.78	0.70	0.46

Since the correct semantic types are given, we excluded their corresponding triples in computing the precision and recall. For example, we do not consider *(E35_Title, label, title)* in our evaluation. From the 785 links in the correct models, 418 links correspond to the semantic types because we have 418 attributes. Thus, our evaluation measures the accuracy of inferring the remaining 367 links.

Table 2 shows the average precision and recall for all 29 sources. An interesting observation is that when we do not use the linked data patterns, the precision and recall are close to zero. This low accuracy comes from the fact that in most of the gold standard models, the attributes are not directly connected and there are multiple paths between each pair of classes in the ontology (and thus in our graph), and without additional information, we cannot resolve the ambiguity. When we exploit the patterns with length one, there is a boost in precision and recall. Since we are using the pattern frequencies in assigning the weights to the links of the graph, using patterns of length one means that we are only taking into account the popularity of the links in computing the semantic models. Once we added the patterns with length two, our approach achieved more than 10% improvement in both precision and recall. This means that considering coherence (even to a small extent) in addition to the link popularity empowers our approach to derive more accurate semantic models.

The column time in Table 2 shows the running time of our algorithm on a single machine with a Mac OS X operating system and a 2.3 GHz Intel Core i7 CPU. This is the time from combining pre-extracted LOD patterns into a graph until generating and ranking candidate semantic models. The algorithm is much faster when we only use the domain ontology as the background knowledge, because adding LOD patterns will be excluded from the graph construction process (lines 1-3 in Algorithm 1). The reason why more time is required when we use patterns of length one comparing to the case where we use patterns of both length one and two is related to details of our algorithm to compute top k minimal trees. Although it takes longer to create the graph when we add patterns of length two, the algorithm to generate the candidate models finds top k trees faster reducing the total running time.

5 Related Work

There has been many studies in the Semantic Web to automatically describe the semantics of data sources as a mapping from the source to an ontology. Since the focus of our work is on inferring the semantic relations, we compare our work with the ones that pay attention to inferring semantic relationship and not only semantic labeling.

Limaye et al. [8] used YAGO[10] to annotate web tables and generate binary relationships using machine learning approaches. Venetis et al. [15] presented a scalable approach to describe the semantics of tables on the Web. To recover the semantics of tables, they leverage a database of class labels and relationships automatically extracted from the Web. They attach a class label to a column if a sufficient number of the values in the column are identified with that label in the database of class labels, and analogously for binary relationships. Although these approaches are very useful in publishing semantic data from tables, they are limited in learning the semantics of sources as a united model. Both of these approaches only infer individual binary relationships between pair of columns. They are not able to find the relation between two columns if no relationship is directly instantiated between the values of those columns. Our approach can connect one column to another one through a path in the ontology.

Carman and Knoblock [3] use known source descriptions to learn a semantic description that precisely describes the relationship between the inputs and outputs of a source, expressed as a Datalog rule. However, their approach is limited in that it can only learn sources whose models are subsumed by the models of known sources. That is, the description of a new source is a conjunctive *combinations* of known source descriptions.

In our earlier Karma work [6], we build a graph from learned semantic types and a domain ontology and use this graph to semi-automatically map a source to the ontology. Since only using the ontology does not necessarily generates accurate models, we had the user in the loop to interactively refine the suggested models. Later, we introduced an automatic approach that exploits the semantic models of similar data sources in addition to the domain ontology to learn a model for a new unknown source [13,14]. Our work in this paper complements our previous work in cases where few, if any, known semantic models are available. In fact, the number of known semantic models is limited in many domains, however, there may be a huge amount of semantic data published in those domain. The presented approach mines the small graph patterns from the available linked data to infer the semantic relationships within data sources.

Our work is closely related to other work leveraging the Linked Open Data (LOD) cloud to capture the semantics of sources. Mulwad et al. [9] used *Wikitology* [12], an ontology which combines some existing manually built knowledge systems such as DBPedia and Freebase [2], to link cells in a table to Wikipedia entities. They query the background LOD to generate initial lists of candidate classes for column headers and cell values and candidate properties for relations between columns. Then, they use a probabilistic graphical model to find the correlation between the columns headers, cell values, and relation assignments. The quality of the semantic data generated by this category of work is highly dependent on how well the data can be linked to the entities in the LOD. While for most popular named entities there are good matches in the LOD, many tables contain domain-specific information or numeric values (e.g., temperature and age) that cannot be linked to the LOD. Moreover, these approaches are only

[10] `http://www.mpi-inf.mpg.de/yago-naga/yago`

able to identify individual binary relationships between the columns of a table. However, an integrated and united semantic model is more than fragments of binary relationships between the columns. In a complete semantic model, the columns may be connected through a path including the nodes that do not correspond to any column in the table.

6 Discussion

We presented a novel approach to infer semantic relations within structured sources. Understanding how the source attributes are related is an essential part of building a precise semantic model for a source. Such models automate the process of publishing semantic data. The core idea of our work is to exploit the small graph patterns occurring in the Linked Open Data to hypothesize attribute relationships within a data source.

Our evaluation results support the theory that more accurate models can be constructed when longer graph patterns from LOD are used. Although these patterns can be pre-computed, using SPARQL queries to extract long patterns from a large number of triples is challenging. For example, the Virtuoso response time to the SPARQL query to extract patterns of length two with a chain shape ($c_1 \xrightarrow{p_2} c_2 \xrightarrow{p_3} c_3$) was approximately 90 seconds, and it was roughly 1 hour for the query to extract patterns of length two having a V-shape ($c_1 \xrightarrow{p_2} c_2 \xleftarrow{p_3} c_3$). We were only able to collect a few patterns with length three and could not extract any pattern with length four from our Virtuoso server in a 5-hour timeout. Efficiently mining more complex patterns from the linked data is one direction of our future work.

One limitation of the presented approach is that it heavily depends on the linked data at hand. It assumes that there is sufficient amount of linked data available in the same domain that we are modeling the target data sources. Another direction of our future work is to extend our approach to cases where no or sparse data is available. We want to investigate how the current method can be combined with our previous work that uses known semantic models to learn semantic models of structured sources [13, 14].

Our work plays a role in helping communities to produce consistent Linked Data so that sources containing the same type of data use the same classes and properties when published in RDF. Often, there are multiple correct ways to model the same type of data. A community is better served when all the data with the same semantics is modeled using the same classes and properties. Our work encourages consistency because our algorithms bias selection of classes and properties towards those used more frequently in existing data.

Acknowledgements. This research was supported in part by the National Science Foundation under Grant No. 1117913 and in part by Defense Advanced Research Projects Agency (DARPA) via AFRL contract number FA8750-14-C-0240. 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 NSF, DARPA, AFRL, or the U.S. Government.

References

1. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword Searching and Browsing in Databases Using BANKS. In: Proceedings of the 18th International Conference on Data Engineering. pp. 431–440 (2002)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. pp. 1247–1250. SIGMOD '08, ACM, New York, NY, USA (2008)
3. Carman, M.J., Knoblock, C.A.: Learning Semantic Definitions of Online Information Sources. Journal of Artificial Intelligence Research 30(1), 1–50 (2007)
4. Ding, L., DiFranzo, D., Graves, A., Michaelis, J., Li, X., McGuinness, D.L., Hendler, J.A.: TWC data-gov corpus: incrementally generating linked government data from data.gov. In: Rappa, M., Jones, P., Freire, J., Chakrabarti, S. (eds.) WWW. pp. 1383–1386. ACM (2010)
5. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: From Spreadsheets to RDF. In: The Semantic Web - ISWC 2008, pp. 451–466 (2008)
6. Knoblock, C., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: Semi-Automatically Mapping Structured Sources into the Semantic Web. In: Proc. 9th Extended Semantic Web Conference (2012)
7. Krishnamurthy, R., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning Semantic Labels to Data Sources. In: Proceedings of the 12th Extended Semantic Web Conference (ESWC) (May 2015)
8. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. PVLDB 3(1), 1338–1347 (2010)
9. Mulwad, V., Finin, T., Joshi, A.: Semantic Message Passing for Generating Linked Data from Tables. In: The Semantic Web–ISWC 2013, pp. 363–378. Springer (2013)
10. Polfliet, S., Ichise, R.: Automated Mapping Generation for Converting Databases into Linked Data. In: Polleres, A., Chen, H. (eds.) ISWC Posters&Demos. CEUR Workshop Proceedings, vol. 658. CEUR-WS.org (2010)
11. Saquicela, V., Blázquez, L.M.V., Corcho, Ó.: Lightweight Semantic Annotation of Geospatial RESTful Services. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC). pp. 330–344 (2011)
12. Syed, Z., Finin, T.: Creating and Exploiting a Hybrid Knowledge Base for Linked Data. In: Agents and Artificial Intelligence, pp. 3–21. Springer (2011)
13. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: A Graph-based Approach to Learn Semantic Descriptions of Data Sources. In: Procs. 12th International Semantic Web Conference (ISWC) (2013)
14. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: A Scalable Approach to Learn Semantic Models of Structured Sources. In: Semantic Computing (ICSC), 2014 IEEE International Conference on. pp. 183–190 (June 2014)
15. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering Semantics of Tables on the Web. Proc. VLDB Endow. 4(9), 528–538 (June 2011)
16. Wang, J., Wang, H., Wang, Z., Zhu, K.Q.: Understanding Tables on the Web. In: Atzeni, P., Cheung, D.W., Ram, S. (eds.) ER. Lecture Notes in Computer Science, vol. 7532, pp. 141–155. Springer (2012)