Linking the Deep Web to the Linked Data Web

Rahul Parundekar, Craig A. Knoblock and **José Luis Ambite**
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
{*parundek,knoblock,ambite*}@*isi.edu*

Abstract

Even though the Linked Data movement is gaining ground, vast amounts of information are only present in the traditional Web of human-readable pages. Data from such sources in the *Surface Web* and the *Deep Web* needs to be published as structured data into the Linked Data Web. The work described in this paper links the schema and individuals in the RDF extracted from surface and deep Web sources with the schema and individuals already present in the linked data cloud. To this end, we extend our prior work on automatically generating Semantic Web Services from Web sources. Once we are able to link individuals of the generated Semantic Web Service with the data present in the linked data cloud, we can populate the Linked Data Web with data from *Deep Web* sources for given domains. Our approach not only integrates known sources from the *Deep Web* into the Linked Data Web, but also *automatically* discovers and links previously unknown sources for the same domain. Our techniques can significantly increase the amount of data available in the Linked Data Web.

Introduction

The Linked Data Web (LDW) is a vast dataset of structured data, which is steadily increasing in volume as a result of independent efforts to publish an organization's knowledge, information and data and linking it with other data already part of the linked data cloud. As of March 2009, according to statistics collected by the linked data community, the estimated size of the linked data cloud is 4.7 billion triples with 142 million RDF links (Bizer, Heath, and Berners-Lee 2009). This has come about due to involvement of big organizations such as BBC, Library of Congress, etc., along with contributions from various organizations in domains like genetics, clinical trials, online communities, etc. Nevertheless, a major part of the WWW remains untapped, as it is based on the traditional Web where data is embedded within HTML pages intended for human consumption.

The amount of data on the linked data cloud would significantly increase if we could automatically convert traditional data sources into structured data like RDF, and at the same time link the RDF to the LDW using the linked data design principles. Much of the interesting data is not available in the surface Web, but only exposed through Web forms that provide a query interface to non-accessible databases, the so-called *Deep Web*. Typical examples of Deep Web sources are sites like amazon.com, weather.com, or finance.yahoo.com, which provide product information, weather forecasts, and stock and mutual funds prices, respectively. Converting such data into RDF would not, by itself, provide its potential benefit unless the knowledge present on the linked data cloud is exploited. It is the added links between local data and what is already out there on the linked data cloud that provides improved knowledge to both the individual as well as the linked data community.

Consider a hypothetical linked data application for tracking a personal portfolio. Assume that data from sources such as banks and trading sites, asset holdings (secure data that cannot be published), and current prices of stocks and mutual funds (data that constantly changes and hence is present only in the Deep Web) needs to be integrated into a single place. Unless these sources are already linked to the cloud, we need a mechanism to pull data from them and link it dynamically to the cloud in order for it to be integrated. Our previous work on automatically generating Semantic Web Services from online sources (described briefly below) provides an approach to generate RDF data from traditional Web sources. In order to exploit the capabilities of integrating this data with the vast knowledge present on the linked data cloud, we propose a mechanism to populate the Linked Data Web dynamically from these sources. Our contribution is thus to solve the problem of information integration between the Linked Data Web and the traditional Web.

In the remainder of the paper, we first describe DEIMOS , our approach to automatically discover and model Semantic Web Services, on which the current work is based. Second, we describe our contribution in automatically connecting novel *Deep Web* sources to the Linked Data Web. Third, we present some initial results on an interesting domain, mutual funds. Fourth, we describe related work in the field of data integration into the Linked Data Web. Finally, we discuss our contributions.

Previous Work

The work in this paper is based on our previous work on automatically constructing Semantic Web Services from on-

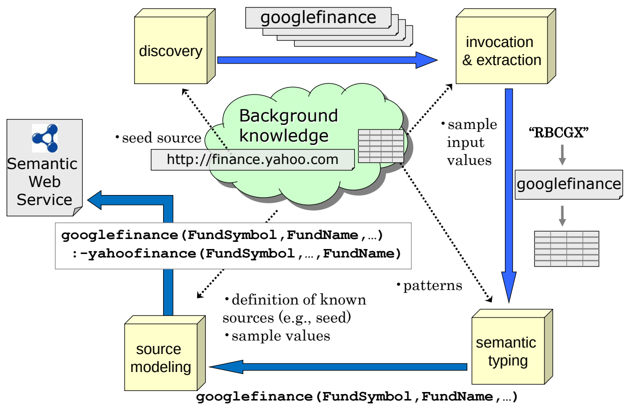

Figure 1: DEIMOS system architecture

line sources (Ambite et al. 2009). DEIMOS is an integration of previous work on tackling the sub-problems of automatic source discovery, extraction and modeling. As a combined system, it works as an end-to-end approach that automatically finds sources, extracts the data from them, determines the semantic types of the outputs, builds the source models, and turns them into Semantic Web Services. Figure 1 shows the overall architecture. DEIMOS starts with a known source (*seed source*) and the description for the domain (*e.g. Mutual funds, Weather, etc.*) it belongs to, and generates Semantic Web Services for similar sources that it discovers. Throughout this paper we explain this system and the linked data generation part using the mutual fund domain.

Background Knowledge

We provide DEIMOS with background knowledge consisting of - (1) Semantic types: e.g., FundSymbol, FundName; (2) Sample values for each type: e.g., "RBCGX" for FundSymbol; (3) Domain input model: a mutual fund source may accept FundSymbol or a FundName as input; (4) Known sources (seeds): e.g., http://finance.yahoo.com; (5) Source descriptions for the seeds: specifications of the functionality of the source in a formal language of the kind used by data integration systems (i.e., Datalog). Using this, DEIMOS executes the following modules to generate a novel Semantic Web Service

Source Discovery

To provide sources that are similar to the seed source, DEIMOS first collects popular tags with which the seed source is annotated on the social bookmarking site del.icio.us. Using Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) DEIMOS learns a compressed description of such sources (Plangprasopchok and Lerman 2009). This is used as input to a similarity determining mechanism to generate the top 100 similar sources, which are then forwarded to the next module. For example, in the mutual funds domain, the system discovers sources such as http://www.google.com/finance and http://moneycentral.msn.com/ among others.

Source Invocation and Extraction

The sources discovered in the previous step are typically Web Pages that use standard HTML forms for input and return a result HTML page. Thus, they can be invoked with inputs and produce output pages that are formatted with the document object model. From the source Web page, DEIMOS extracts all the forms and input fields including text boxes, select items, etc. DEIMOS uses a brute force approach, trying all permutations of input values based on background knowledge of the type of data (e.g. FundSymbol or FundName for the current domain) in the input form's fields, to identify inputs that give meaningful and extractable results.

Next, DEIMOS extracts data from pages returned by the source in response to a query. For this, DEIMOS uses the Autowrap algorithm (Gazen and Minton 2005), which exploits the regularity of dynamically generated pages. It assumes that the organization of dynamically generated pages is specified through a *page template* that is shared by all pages returned by the source. Given two or more sample pages, we can derive the page template and use it to extract data from the pages. The output of the extraction step is a table of data fields.

Semantic Typing of Sources

DEIMOS uses the approach described in (Lerman, Plangprasopchok, and Knoblock 2007) to semantically type data extracted from Web sources. Each semantic type can be described using certain patterns. Using heuristics to evaluate the quality of the match between the values of a particular column in the data extracted in the previous step with a semantic type, DEIMOS assigns the best semantic type to each of the inputs and outputs of the source. For example, the input/output type signature learned for the discovered source www.google.com/finance is:

```
googlefinance($FundSymbol,FundName,FundName,FundSymbol,
  YTDReturn,YTDReturn,NetAssets,YTDReturn,NetValue,Yield,
  ChangePercent,ChangeAmount,NetValue,ChangeAmount).
```

Source Modeling

The typed input/output signature of a new source offers only a partial description of the source's behavior. What we need is a semantic characterization of its functionality—the relationship between its input and output parameters. We use the approach described in Carman & Knoblock (2007) to learn a Local-as-View (LAV) description of the target source (a Datalog rule) (Levy 2000). DEIMOS learns these definitions by combining *known* sources to emulate the input/output values of a new *unknown* source.

DEIMOS uses an approach based on Inductive Logic Programming to enumerate the search space of source descriptions in an efficient, best-first manner and prune candidate hypotheses to find the best rule to explain the observed input and output data. As a result, we have a definition of the discovered source in terms of the seed source/s. In our running example, the definition of the newly discovered source googlefinance in terms of the background source yahoofinance is:

```
googlefinance($FundSymbol1,_,FundName3, _,
    YTDReturn5,_,NetAssets7, _,_,Yield10,
    ChangePercent11,ChangeAmount12,NetValue13,_) :-
  yahoofinance($FundSymbol1, NetValue13, _,
      ChangeAmount12, ChangePercent11, _,
      YTDReturn5, NetAssets7, Yield10, FundName3).
```

Since the background source yahoofinance has an associated LAV source description, DEIMOS can automatically generate a source description for googlefinance as we describe next.

Source descriptions are based on a schema/ontology of the application domain. For example, the definition for yahoofinance is:

```
yahoofinance($FundSymbol, NetValue, ChangeDirection,
     ChangeAmount,ChangePercent, PreviousClose,
     YTDReturn, NetAssets, Yield, FundName) :-
  Company(@C), offersSeries(@C,@S), Series(@S),
  offersContract(@S,@Ct), Contract(@Ct),
  hasSymbol(@Ct,@Sy), Symbol(@Sy),
   hasValue(@Sy, FundSymbol),
  hasName(@Ct,@N), Name(@N), hasValue(@N, FundName),
  hasNetValue(@Ct,@Net), NetValue(@Net),
   hasValue(@Net, NetValue),
  hasNetAssets(@Ct,@NA), NetAssets(@NA),
   hasValue(@NA, NetAssets),
  hasYield(@Ct,@Y), Yield(@Y), hasValue(@Y, Yield),
  hasYTDReturn(@Ct,@Ret), YTDReturn(@Ret),
   hasValue(@Ret, YTDReturn),
  hasChangeAmount(@Ct,@ChA), ChangeAmount(@ChA),
   hasValue(@ChA, ChangeAmount),
  hasChangePercent(@Ct,@ChP), ChangePercent(@ChP),
   hasValue(@ChP, ChangePercent),
  hasChangeDirection(@Ct,@ChD), ChangeDirection(@ChD),
   hasValue(@ChD, ChangeDirection),
  hasPreviousClose(@Ct,@Pre), PreviousClose(@Pre),
   hasValue(@Pre, PreviousClose).
```

Thus, given the relationship that DEIMOS learned between the target source googlefinance and the background source yahoofinance, the automatically generated description for googlefinance is:

```
googlefinance($FundSymbol1,_,FundName3, _,
    YTDReturn5,_,NetAssets7, _,_,Yield10,
    ChangePercent11,ChangeAmount12,NetValue13,_) :-
  Company(@C), offersSeries(@C,@S), Series(@S),
  offersContract(@S,@Ct), Contract(@Ct),
  hasSymbol(@Ct,@Sy), Symbol(@Sy),
   hasValue(@Sy, FundSymbol1),
  hasName(@Ct,@N), Name(@N), hasValue(@N, FundName3),
  hasNetValue(@Ct,@Net), NetValue(@Net),
   hasValue(@Net, NetValue13),
  hasNetAssets(@Ct,@NA), NetAssets(@NA),
   hasValue(@NA, NetAssets7),
  hasYield(@Ct,@Y), Yield(@Y), hasValue(@Y, Yield10),
  hasYTDReturn(@Ct,@Ret), YTDReturn(@Ret),
   hasValue(@Ret, YTDReturn5),
  hasChangeAmount(@Ct,@ChA),  ChangeAmount(@ChA),
   hasValue(@ChA, ChangeAmount12),
  hasChangePercent(@Ct,@ChP), ChangePercent(@ChP),
   hasValue(@ChP, ChangePercent11).
```

Automatically Generating Semantic Web Service

DEIMOS generates a Semantic Web Service(SWS) which encapsulates the Web source discovered in the source modeling phase. The SWS acts as a 'semantic' wrapper that accepts RDF as input and generates RDF output. The semantic wrapper includes specific instantiations of the DEIMOS modules. First, it forwards the appropriate values from the input RDF to the discovered Web form as inputs, and invokes the form. Second, it extracts the data from the resulting HTML page. Finally, it applies the learned source description to generate RDF triples according to the domain ontology. Conversion from the source description into RDF is straightforward. The variables prefixed by '@' denote an object id, which in RDF correspond to an automatically generated URI. For example, a subset of output triples for googlefinance with input fund symbol 'RBCGX' is:

```
company5179861 rdf:type Company .
series382953 rdf:type Series .
company5179861 offersSeries series382953 .
contract1885719 rdf:type Contract .
series382953 offersContract contract1885719 .
symbol2139169 rdf:type Symbol .
contract1885719 hasSymbol symbol2139169 .
symbol2139169 hasValue "RBCGX" .
name1093443 rdf:type Name .
contract1888902 hasName name1093443 .
name1093443 hasValue "Reynolds Blue Chip Growth" .
...
```

This Semantic Web Service can now be used to produce structured (RDF) data from *Deep Web* sources.

Integrating *Deep Web* data sources into the Linked Data Web

The Semantic Web Service generated by DEIMOS produces results as RDF which we now want to integrate into the Linked Data Web. During execution of this Web service, the RDF data which is output contains auto-generated individuals (URIs). We need to link these URIs to their corresponding individuals already present in the linked data Web. To do this, we first model our seed source in terms of the ontology of the linked data source with which we are trying to integrate our domain. We thus begin with integrating our seed source into the LDW. For any newly discovered source that we are able to model, we use its definition in terms of the seed source for its integration into the LDW.

Linking the seed source to the Linked Data Web

If we use the values extracted during the execution of the newly discovered source to retrieve individuals from the linked data source, we could link the individuals generated by the SWS to the LDW. We first devise a mechanism to query the linked data source using the known model of the seed source. As described in the source modeling section, we can understand any newly discovered source in terms of the seed source. This property can be used in combination with the query mechanism to retrieve LDW individuals and subsequently produce linked data.

We begin by describing the seed source with the ontology of the linked data source under consideration. Around

2.5 million triples of U.S. corporate information is already published at http://www.rdfabout.com/demo/sec/ as part of the LDW using an ontology which draws from the EDGAR database of the Securities and Exchange Commission(SEC). As the mutual fund domain cannot be completely described by the existing ontology, we slightly extend it by extrapolating the concepts for 'Contract' or 'Series' from the EDGAR database and assume it to already be a part of the LDW for the purposes of this paper.

We then use the data related to each individual in the SWS, which was generated at runtime, to search for a corresponding individual in the LDW by querying over the values of properties of the individuals. Because the ontology of the seed source is same as the linked data source, the similarity of two individuals is concluded by comparing values of the data properties of the local individual with the values of the corresponding aligned properties of the linked source individual. Ideally, this would be a matching based on the equivalence of strings. However practically, we need to use a string similarity metric to overcome variations in representation of the values of the same thing.

The matching of individuals generated by the SWS to those present on the LDW can be represented by a SPARQL query that constructs 'owl:sameAs' assertions, with the WHERE part selecting the LDW URIs which we want to link the generated URIs, using data values of the properties of the individuals. Because our seed source is defined with the same ontology as that of the linked data source, formulation of the query is straightforward. Following is an explanatory SPARQL query for matching individuals of yahoofinance with the linked data source at http://www.rdfabout.com/demo/sec/.

```
CONSTRUCT{
  ?C1 owl:sameAs ?C2 .
  ?S1 owl:sameAs ?S2 .
  ?Con1 owl:sameAs ?Con2 .
}
WHERE{
  ?C1 rdf:type Company .
  ?S1 rdf:type Series .
  ?C1 offersSeries ?S1 .
  ?Con1 rdf:type Contract .
  ?S1 offersContract ?Con1 .
  ?Con1 hasSymbol ?S1 .
  ?S1 hasValue ?FundSymbol .
  ?Con1 hasName ?N1 .
  ?N1 hasValue ?FundName .

  ?C2 rdf:type Company .
  ?S2 rdf:type Series .
  ?C2 offersSeries ?S2 .
  ?Con2 rdf:type Contract .
  ?S2 offersContract ?Con2 .
  ?Con2 hasSymbol ?S2 .
  ?S2 hasValue ?SV2 .
  ?Con2 hasName ?N2 .
  ?N2 hasValue ?NV2 .

  FILTER { matchContractsUsingValues(?Con1,
           ?FundSymbol, ?FundName, ?Con2, ?SV2, ?NV2) }
  FILTER { presentInLDWSource(?Con2) }
  FILTER { generatedByYahoofinanceSWS(?Con1) }
  }
}
```

The presence of a functional relation from one or more of the values of the arguments (e.g. FundSymbol), to the individual, whose URI is the output, is a requirement for using those values as input to the matcher. At execution time, the input variables of the matcher can be grounded to values of the arguments of the source predicate, which are extracted from the result page.

The SPARQL query for inputs FundSymbol='RBCGX' and FundName='Reynolds Blue Chip Growth' can be explained as follows. The query would first retrieve individuals of type *Contract* based on values for FundSymbol and FundName from the individuals generated in the Semantic Web Service for yahoofinance (Filter on YahooSWS). This query then invokes a function (*matchContractsUsingValues*) that matches *Contract* individuals from the seed and the linked data source, based on a string similarity metric on the data values of their properties. The URI of such an individual is linked to the *Contract* individual from the yahoofinance SWS by using the 'owl:sameAs' relation. We then match individuals of type *Series* using the *offersContract* property, and finally match individuals of type *Company* using the *offersSeries* property.

After this process, the seed source becomes a part of the Linked Data Web.

Linking discovered sources to the Linked Data Web

After a new source is discovered and semantically modeled, we can integrate it into the LDW by using the same URI matching technique described in the previous section. DEIMOS defines the target predicate in terms of the seed source, we get an unfolding of the target as RDF (unary and binary predicates) and thus produce the Semantic Web Service (c.f. the googlefinance definition shown above). At runtime, we use this SWS and augment it with the URI matching approach from the previous section, as our target definition is semantically characterized in terms of the seed source. We can now link auto-generated individuals from the structured data produced by the SWS to the individuals from the LDW using the 'owl:sameAs' relation.

The linked Semantic Web Service that is generated for the target source - googlefinance executes as follows:

```
1. Accept a Mutual Fund symbol as input.
2. Execute live over www.google.com/finance.
3. Extract the relevant values from the output page.
4. Execute the SPARQL query which invokes the matcher
    over the SEC database to get URIs for individuals
    already present on the LDW.
5. Generate the RDF Triples from the target description,
   which is defined in unary and binary predicates.
```

Implementation & Results

We implemented our system to integrate Web sources belonging to the mutual fund domain (as described in this paper). Our Seed Source was finance.yahoo.com. We modeled the seed source based on an ontology extrapolated from http://www.rdfabout.com/demo/sec/. This ontology originally contains only details about the companies. We assumed a similar representation of the concepts of *Series* and *Contract* in the same way that they are defined in the SEC database to create an extended version of the ontology. Two example sources belonging to the mutual fund domain that were automatically discovered and modeled by DEIMOS are www.google.com/finance and moneycentral.msn.com/detail/stock_quote.

The mechanism described in the SPARQL query was implemented as follows. Ideally, the triples belonging to the LDW source, that we intend to link our seed source to, could be downloaded into a local triple store. Using suitable string similarity metrics, we can query this store to retrieve the URIs for the mutual fund (Contract, and thus Series & Company). The URIs generated by the SWS can now be linked to these URIs using the 'owl:sameAs' property. The SWS of our seed source can now generate linked data. For a newly discovered source, using its definition in terms of the seed source, we can perform a similar retrieval query for the URIs and thus integrate it into the Linked Data Web. Practically, we had to abstract this URI retrieval to a wrapper on account of the complete SEC database not being downloadable. A wrapper that accepted FundSymbol & FundName called the search page on the SEC website[1] and produced the URIs after relevant data extraction from the result page (matching the contracts). We were able to do this because the inputs to the search form have a correspondence with the value of the 'hasValue' properties of the Symbol and Name for the Contract individual. The 'owl:sameAs' triples that this generated are as explained in the Construct part of the SPARQL query.

For the discovered source googlefinance described in the paper, a sample part of the result of the implemented Semantic Web Service, producing linked data for the input fund symbol 'RBCGX' is shown below. The URIs generated for the Series and Contract are not currently present at www.rdfabout.com. So we infer the pattern of URIs based on those for Company and assume their existence on the linked data Web. The resulting owl:sameAs statements are:

```
company5179861 rdf:type Company .
company5179861 owl:sameAs
  http://www.rdfabout.com/rdf/usgov/sec/id/cik0000832574 .
series382953 rdf:type Series .
series382953 owl:sameAs
  http://www.rdfabout.com/rdf/usgov/sec/id/S000000865 .
company5179861 offersSeries series382953 .
contract1885719 rdf:type Contract .
contract1885719 owl:sameAs
  http://www.rdfabout.com/rdf/usgov/sec/id/C000002481 .
series382953 offersContract contract1885719 .
symbol2139169 rdf:type Symbol .
contract1885719 hasSymbol symbol2139169 .
symbol2139169 hasValue "RBCGX" .
name1093443 rdf:type Name .
contract1888902 hasName name1093443 .
name1093443 hasValue "Reynolds Blue Chip Growth" .
...
```

Our results show that it is possible to convert data provided by a previously unknown data source from the Deep Web, into structured linked data and thus populate the LDW.

Related Work

The integration of sources on the Semantic Web is not a new concept. Noy (2004) presents a survey of Ontology based approaches for data integration. This involves, the alignment of the source and target ontologies based on concepts and their hierarchy, properties, rules and individuals of the concepts. For example, GLUE (Doan et al. 2003) uses machine learning techniques to match instances and, thus, the Ontology itself. In this case however the source to be matched is assumed to have a background ontology and is not a traditional Web source.

There is also some existing work done on generating linked data from present data sources on the Web. With the rise in popularity of the Linked Data Web, there have been numerous initiatives to convert existing formatted data e.g. data formatted with XML, into linked data. Garce and Gil (2009) describe a mechanism to publish documents based on the XML Business Reporting Language as linked data. The mechanism involves aligning the Schema to an OWL Ontology and transforming document instances into linked data by using an XML to RDF mapping.

The Virtuoso Sponger (OpenLinkSW 2009) is an *'RD-Fizer'* that transfroms non-RDF data into RDF. At the heart of the Sponger are Cartridges that extract metadata from a page and then match entities to ontologies and produce RDF output. These cartridges are mostly programmed manually[2], using one of the various supported languages, for a group of Web pages with matching URLs or content type and are thus limited to known sources. In order to generate a domain specific cartridge, one would need to extract trained semantic types from pages, unrestricted to a URL pattern and content type, and then perform modeling and data linkage in a similar way described in the paper so that a previously unseen, unknown source can effectively be *RDFized*.

Most of the integration of data to the Linked Data Web, that is currently part of the cloud, is based on known sources with a predetermined set of integration rules and is thus human centric. Automated and semi-automated means of generating also have been studied. For example, the SILK - Link Discovery Framework (Bizer et al. 2009) tries to discover links, specified by users with the *Link Specification Language*, between sources that already have structured (RDF) data. Our system, however, is not only able to dynamically generate and integrate data from known *Deep Web* sources into the linked data cloud, but also has the capability of integrating previously unknown data sources in the same domain.

Conclusion

We were able to automatically integrate sources from the *Deep Web* into the Linked Data Web by extracting structured

[1]http://www.sec.gov/edgar/searchedgar/mutualsearch.htm

[2]http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtProgrammerGuideRDFCartridge

data from these sources and linking them with the data in the cloud. Our results suggest that the large data present in the *Deep Web* can now be accessible as linked structured data. The proposed mechanism thus proves to be a substantial step in solving the problem of information integration between the *Linked Data Web* and the *Deep Web*.

References

Ambite, J.-L.; Darbha, S.; Goel, A.; Knoblock, C. A.; Lerman, K.; Parundekar, R.; and Russ, T. 2009. Automatically constructing semantic web services from online sources. *International Semantic Web Conference*.

Bizer, C.; Volz, J.; Kobilarov, G.; and Gaedke, M. 2009. Silk - a link discovery framework for the web of data. In *18th International World Wide Web Conference*.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data - the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Carman, M. J., and Knoblock, C. A. 2007. Learning semantic definitions of online information sources. *Journal of Artificial Intelligence Research (JAIR)* 30:1–50.

Doan, A.; Madhavan, J.; Dhamankar, R.; Domingos, P.; and Halevy, A. 2003. Learning to match ontologies on the semantic web. *The VLDB Journal* 12(4):303–319.

Garca, R., and Gil, R. 2009. Publishing xbrl as linked open data. In *WWW2009 Workshop: Linked Data on the Web (LDOW2009)*.

Gazen, B., and Minton, S. 2005. Autofeed: an unsupervised learning system for generating webfeeds. In *K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture*, 3–10. New York, NY, USA: ACM.

Lerman, K.; Plangprasopchok, A.; and Knoblock, C. A. 2007. Semantic labeling of online information sources. *International Journal on Semantic Web and Information Systems, Special Issue on Ontology Matching* 3(3):36–56.

Levy, A. Y. 2000. Logic-based techniques in data integration. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Kluwer Publishers.

Noy, N. F. 2004. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.* 33(4):65–70.

OpenLinkSW. 2009. Virtuoso sponger. *http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSpongerWhitePaper*.

Plangprasopchok, A., and Lerman, K. 2009. Modeling social annotation: a bayesian approach. Technical report, Computer Science Department, University of Southern California.