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 constantly gaining ground, a huge chunk of information is still 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 individuals in the RDF extracted from such sources with individuals already present in the linked data cloud. The extraction of structured data from these sources is based on 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 data from *Deep Web* sources belonging to different domains into the Linked Data Web. The contribution of the system is that it not only integrates known sources from the *Deep Web* into the linked data web but also links previously unknown, similar sources, and helps generate potentially huge amount of structured data.

Introduction

The Linked Data Web is a vast dataset of structured data which is steadily increasing in volume as a result of independent efforts to publish an organisation'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. A major part of the WWW however remains untapped as it is based on the traditional web where data is embedded within HTML pages in unstructured text as well as structured tables, etc.

The amount of data on the linked data cloud would significantly improve if we had a way to convert traditional data sources into structured data like RDF and at the same time, link them to the cloud using the linked data design principles. Some of these traditional sources also fall into the category of the *Deep Web*, where data is not directly exposed on the surface and thus cannot be indexed. Moreover, we may also come accross temporal data which though changing, might provide useful knowledge e.g. the current weather conditions for a zip code, prices and statistical data of stocks and mutual funds, etc. Converting such data into RDF would not, by itself, be able to provide its potential benefit unless the knowledge present on the linked data cloud is exploited. It is the added links between local data and whats already out there on the cloud that provides improved knowledge to both the individual as well as the linked data community.

Consider a hypothetical linked data application for tracking personal portfolio. Data from various sources such as various banks and trading sources (*secure data that cannot be statically published*), asset holdings, and current prices of stocks, mutual funds etc. (*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 in the previous work section) provides a means to generate RDF data from the 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 paper that follows, we first describe the implementation of the DEIMOS system on which the current work is based. This is followed by the core section which describes our contribution in automatically connecting novel *Deep Web* sources to the Linked Data Web. We also describe results of our implementation for the mutual fund domain. The related work section describes other work in the field of data integration into the linked data web. Finally, we provide the conclusion of our work.

Previous Work

The work in this paper is based on previous work on automatically constructing Semantic Web Services from online sources (Ambite et al. 2009). DEIMOS is an integration

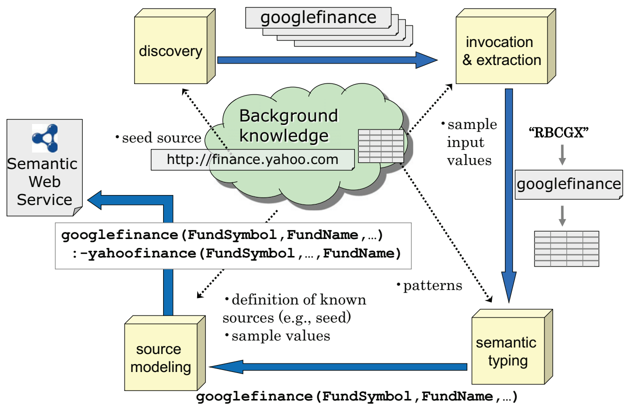

Figure 1: DEIMOS system architecture

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.

The example used in this paper is the mutual fund domain, where the background knowledge consists 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.

DEIMOS then executes the following modules to generate a 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. E.g. in the Mutualfund domain, the system discovers sources like http://www.google.com/finance & 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 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. The valid set of mappings from inputs that give meaningful and extractable results are then used. The Autowrap algorithm (Gazen and Minton 2005) is then used to generate a *page template* from the result pages of multiple input examples for one candidate mapping. The set of mappings along with corresponding *page templates* are forwarded to the next phase.

Semantic Typing of Sources

Using the approach described in (Lerman, Plangprasopchok, and Knoblock 2007), we can now 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 from the *page template* with a semantic type, DEIMOS assigns the best semantic type to each of the inputs and outputs of the source. For example, a subset of the type signature defined for seed source finance.yahoo.com is:

```
yahoofinance($FundSymbol, NetValue, ChangeDirection,
   ChangeAmount, ChangePercent, PreviousClose,
   YTDReturn, NetAssets, Yield, FundName).
```

Source Modeling

In order to learn the relationship between the input and output parameters of a discovered source, we use the approach described in (Carman and Knoblock 2007) to learn a Local-as-View (LAV) description of the source. Even though we know the semantic types of the parameters of the discovered source (target), we need to define its predicate in terms of the seed source and match it to the semantic characterization of the seed source. With a discovered source, there may be multiple occurences of arguments with the same semantic types, from which we need to understand the different meanings behind these types. E.g. In a similar stock market domain today’s close, previous day’s close, intraday high, intraday low, etc. all have the semantic type of NetValue. For our Mutual Fund domain, the unmodeled target predicate of the discovered source www.google.com/finance is shown below.

```
googlefinance($FundSymbol1,FundName2,FundName3,
 FundSymbol4,YTDReturn5,YTDReturn6,NetAssets7,
 YTDReturn8,NetValue9,Yield10,ChangePercent11,
 ChangeAmount12,NetValue13,ChangeAmount14).
```

DEIMOS uses an approach similar to Inductive Logic Programming is used to enumerate the search space of inputs and output candidates 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, for example:

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

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

Automatically Generating the Semantic Web Service

DEIMOS can now generate 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, and wraps the web form learnt by DEIMOS . The appropriate values from the input RDF are forwarded to the discovered web form as inputs. Upon execution of the form, the page template described previously is applied to the result page to extract data values corresponding to the semantic types that the domain understands. The source description is now used to convert these values into RDF triples.

The source description consists of unary and binary predicates that reflect the domain ontology - i.e. the individual declarations and property value assertions for such individuals. Conversion from such a definition into RDF is straightforward, once the local identifiers for the individuals in the definition have been replaced with suitable auto-generated URIs. For example, the definition for yahoofinance is:

```
 yahoofinance($PR_FundSymbol,PR_NetValue,
  PR_ChangeDirection, PR_ChangeAmount,PR_ChangePercent,
  PR_PreviousClose,PR_YTDReturn, PR_NetAssets,PR_Yield,
  PR_FundName) :-
   Company(@C), Series(@S), offersSeries(@C,@S),
   Contract(@Con), offersContract(@S,@Con),
   Symbol(@Sy), hasSymbol(@Con,@Sy),
   hasValue(@Sy, PR_FundSymbol), Name(@N),
   hasName(@Con,@N), hasValue(@N, PR_FundName),
   NetValue(@Net), hasNetValue(@Con,@Net),
   hasValue(@Net, PR_NetValue), NetAssets(@NA),
   hasNetAssets(@Con,@NA), hasValue(@NA, PR_NetAssets),
   Yield(@Y), hasYield(@Con,@Y), hasValue(@Y, PR_Yield),
   YTDReturn(@Ret), hasYTDReturn(@Con,@Ret),
   hasValue(@Ret, PR_YTDReturn), ChangeAmount(@ChA),
   hasChangeAmount(@Con,@ChA),
   hasValue(@ChA, PR_ChangeAmount), ChangePercent(@ChP),
   hasChangePercent(@Con,@ChP),
   hasValue(@ChP, PR_ChangePercent),
   ChangeDirection(@ChD), hasChangeDirection(@Con,@ChD),
   hasValue(@ChD, PR_ChangeDirection),
   PreviousClose(@Pre), hasPreviousClose(@Con,@Pre),
   hasValue(@Pre, PR_PreviousClose).
```

Once a new target like googlefinance is discovered, we can unfold it over the source definition. At runtime the RDF triples are generated using this unfolding. A URI generator automatically replaces local references (e.g. *@C*, *@S*, etc.) with URIs. The data values for the semantic types extracted from the result page are filled into the triples generated from the unfolding in their corresponding places. For example, a subset of output triples for googlefinance with input fund symbol 'RBCGX' look like:

```
  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 consumed for extracting structured 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. We need to link these 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 that we learn to integrate it into the LDW.

Linking the seed source to the Linked Data Web

A local individual can be linked to an individual already present in the LDW by using the 'owl:sameAs' relation. We begin by defining the domain ontology of the seed source with the ontology of the linked data source under consideration. We then use the data related to the local individual, which was generated at runtime, to search for an individual in the LDW by querying over the values of its properties. Because the ontology of the seed source is same as the linked data source, the URI lookup problem gets simplified to a URI matching problem. 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 URI matching can be represented by a SPARQL query that constructs 'owl:sameAs' assertions, with the WHERE part selecting the variable representing the URI, which we want to link the local individual to, using data values of the properties of the local individual. 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 the 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 {
      contractMatcher(?C1, ?FundSymbol,
        ?FundName, ?C2, ?SV2, ?NV2) .
   }
   FILTER {
      LDW(?C2).
   }
   FILTER {
      YahooSWS(?C1) .
   }
}
```

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='RBCIX' 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 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. We then filter and keep those individuals that are from the linked data source. 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.

As part of realizing the linking of individuals generated from the Semantic Web Service to the linked data web, we aligned our domain ontology to a slightly extended version of the ontology present at http://www.rdfabout.com/demo/sec/. This ontology draws from the EDGAR database of the securities and exchange commission(SEC) but does not include the concepts for 'Contract' or 'Series'. We extrapolate this part of the ontology from the EDGAR database and assume it to be already a part of the LDW in order to support our case. We now have our seed source integrated into the LDW.

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. As we can define the target predicate in terms of the seed source, we get an unfolding of the target as unary and binary predicates and thus produce the Semantic Web Service. At runtime, we use this SWS and augment it with the URI matching part 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 indivduals from the LDW using the 'owl:sameAs' relation.

```
googlefinance($PR_FundSymbol,PR_NetValue,
  PR_ChangeDirection, PR_ChangeAmount,PR_ChangePercent,
  PR_PreviousClose,PR_YTDReturn, PR_NetAssets,PR_Yield,
  PR_FundName) :-
 Company(@C), sameAs(@Con, URI1), Series(@S),
 sameAs(@S, URI2), offersSeries(@C,@S), Contract(@Con),
 sameAs(@C, URI3), offersContract(@S,@Con),
 Symbol(@Sy), hasSymbol(@Con,@Sy),
 hasValue(@Sy, PR_FundSymbol), Name(@N),
 hasName(@Con,@N), hasValue(@N, PR_FundName),
 NetValue(@Net), hasNetValue(@Con,@Net),
 hasValue(@Net, PR_NetValue), NetAssets(@NA),
 hasNetAssets(@Con,@NA), hasValue(@NA, PR_NetAssets),
 Yield(@Y), hasYield(@Con,@Y), hasValue(@Y, PR_Yield),
 YTDReturn(@Ret), hasYTDReturn(@Con,@Ret),
 hasValue(@Ret, PR_YTDReturn), ChangeAmount(@ChA),
 hasChangeAmount(@Con,@ChA),
 hasValue(@ChA, PR_ChangeAmount), ChangePercent(@ChP),
 hasChangePercent(@Con,@ChP),
 hasValue(@ChP, PR_ChangePercent),
 ChangeDirection(@ChD), hasChangeDirection(@Con,@ChD),
 hasValue(@ChD, PR_ChangeDirection),
 PreviousClose(@Pre), hasPreviousClose(@Con,@Pre),
 hasValue(@Pre, PR_PreviousClose).
```

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.
```

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. A couple of examples of the sources that were automatically discovered and modeled by DEIMOS are *www.google.com/finance*, *moneycentral.msn.com/detail/stock_quote*.

For the discovered source googlefinance described in the paper, a sample part of the Linked Data produced by the Semantic Web Service for the input fund symbol 'RBCGX' is shown below. The URIs generated for the Series and Contract are though not currently present at www.rdfabout.com, we infer the pattern of URIs based on those for Company and assume their existance on the linked data web.

```
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" .
...
```

The URI matching query was implemented by using wrappers (on account of the complete SEC database not being downloadable) for the Mutual Fund search form on the SEC site[1]. We can 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.

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

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. (Garca and Gil 2009) describes 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.

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.

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

Conclusion

We were able to automatically integrate sources from the *Deep Web* into the Linked Data Web by extracting structured 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 accesible as linked structured data and thus solve 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.

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

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