Learning Source Descriptions for Web Services

Mark James Carman
Universitá degli Studi di Trento
currently at
Information Sciences Institute, University of Southern California
4676 Admiralty Way Marina del Rey, CA 90292

Introduction

New Web Services are being made available on the internet all the time, and while some of them provide completely new functionality, most are slight variations on already existing services. I am interested in the problem of enabling systems to take advantage of new services without the need for reprogramming them. An existing system can only make use of the new service if it has some notion as to what functionality the service provides. There are three approaches to gaining such knowledge, which I term standardization, semantic markup and induction:

- define standard schemas and sets of operations (service descriptions) for the information domain and require that all service providers use those schemas.
- rely on service providers to annotate their services with semantic labels, corresponding to concepts from an ontology, and then employ semantic web techniques to reason about mappings between ontologies "understood" by the client, and those used by the provider.
- place no requirements on the service provider, but employ schema matching (Rahm & Bernstein 2001) and service classification techniques (Heß & Kushmerick 2003) to hypothesize what functionality the service might provide.

I follow the third approach, taking the idea one step further by actively querying sources to see if the output agrees with that defined by the model. Moreover, I search *not only* for identical services, but see whether a new service has a different scope, or indeed combines the functionality of other known services.

I restrict the problem to that of dealing with services which only produce information, without having any affect on the "state of the world". For example, I am interested in services which provide access to "flight information" say, but not those which allow the user to "buy a ticket". I follow the Local-as-View (LAV) (Levy *et al.* 1995) approach to modeling information producing services in which the information domain is modeled by a set of so-called "domain relations" or "global predicates", and each source is described as a view over the domain relations.

The problem of discovering the functionality of a new service can thus be seen as the problem of discovering the LAV source description associated with it. There is a lot of information available that can be used to discover this source description, including the metadata describing the service, similar services with LAV source descriptions, and examples of the input and output of these services.

Motivating Example

The proposal becomes clearer when we analyse an example. In the example we have five semantic data types:

$\texttt{airport}$, with examples $\{\texttt{LAX}, \texttt{JFK}, \texttt{SFO}, \texttt{ORD}\}$
$\texttt{temperature}$, with examples $\{\texttt{15F}, \texttt{12.4F}, \texttt{70F}, \texttt{27.8F}\}$
$\texttt{latitude}$, $\texttt{longitude}$, and $\texttt{zip\_code}$

We also have three domain predicates, defined as follows:

$\texttt{weather}(\texttt{latitude}, \texttt{longitude}, \texttt{temperature})$
$\texttt{zip}(\texttt{latitude}, \texttt{longitude}, \texttt{zip\_code})$
$\texttt{airport}(\texttt{airport}, \texttt{latitude}, \texttt{longitude})$

And three known sources:

$\texttt{Zip2Temp}(\texttt{zip}^b, \texttt{temp}^f)\text{:-}\texttt{zip}(\texttt{X}, \texttt{Y}, \texttt{zip}), \texttt{weather}(\texttt{X}, \texttt{Y}, \texttt{temp})$
$\texttt{ZipFind}(\texttt{lat}^b, \texttt{lon}^b, \texttt{zip}^f)\text{:-}\ \texttt{zip}(\texttt{lat}, \texttt{lon}, \texttt{zip})$
$\texttt{AirportInfo}(\texttt{iata}^b, \texttt{lat}^f, \texttt{lon}^f)\text{:-}\ \texttt{airport}(\texttt{iata}, \texttt{lat}, \texttt{lon})$

The new service for which we want to discover the source description is: $\texttt{AirportWeather}(\texttt{code}^b, \texttt{temp}^f)$

Based on the metadata similarity, the envisaged system would hypothesize that the input parameter should have the semantic type airport. To test this hypothesis it invokes the source using examples of this type, producing the tuples:

$\langle \texttt{LAX}, \texttt{63.0F}\rangle, \langle \texttt{JFK}, \texttt{28.9F}\rangle, \langle \texttt{SFO}, \texttt{60.1F}\rangle, \langle \texttt{ORD}, \texttt{28.0F}\rangle$

The fact that tuples were actually returned by the source lends credibility to the classification of the input type, but to be more confident, negative examples would be needed. The system could generate negative examples automatically, by generalizing the positive examples of type airport to the regular expression $[\text{A-Z}]^3$, and then selecting examples of other semantic types which don't fit the expression.

Now, by comparing both the metadata and data of the output attribute, the system classifies it to be of type temperature. The system can be far more confident about this classification, because the data used to classify the attribute was produced by the service. Having established the semantic types of the source input and output, the system can use this information to generate a set of plausible definitions for the new service, such as:

$\texttt{AirportWeather}(\texttt{code}^b, \texttt{temp}^f)\text{:-}$
$\quad \texttt{airport}(\texttt{code}, \texttt{Y}, \texttt{Z}), \texttt{weather}(\texttt{Y}, \texttt{Z}, \texttt{temp})$

The description involves two domain relations, because no single predicate takes both arguments. According to the principle of Occam's Razor the simplest model which explains the data is assumed the most plausible. In this case, the simplest model would have been the Cartesian product of the two relations. It is unlikely, however, that somebody would create a service which returns the Cartesian product, when they could provide access to the relations separately. Thus the system starts with the more complicated service description, in which attributes of the same type are joined across relations. This simple heuristic provides a kind of inductive search bias for the learning system (Nédellec *et al.* 1996).

To test this new source description, the system needs to generate data which adheres to it, i.e. the system must query the known sources using the new source description. A reformulation algorithm is used to generate a rewriting of the query in terms of the known services:

$\mathtt{q(A,B)}$:- $\mathtt{airport(code,Y,Z), weather(Y,Z,temp)}$

:- $\mathtt{AirportInfo(A,X,Y), ZipFind(X,Y,Z), Zip2Temp(Z,B)}$[1]

Now, using the same `airport` codes, the reformulated query produces the following tuples:

$\langle\mathtt{LAX,62F}\rangle, \langle\mathtt{JFK,28F}\rangle, \langle\mathtt{SFO,59F}\rangle, \langle\mathtt{ORD,28F}\rangle$

Based on the fact that the values returned by the query are "similar" to those returned by the source, we can state that the new source description is probably correct. Notice that the coverage of the new source is different from the query over the other sources, as it can provide weather information about airports outside of the US. Thus by inducing the definition of a new source using our knowledge of existing sources, we are not only discovering new ways of accessing the same information, but are also expanding the amount of information available for querying!

Problem Definition

The problem I tackle can be expressed as: $\langle T, P, S, n\rangle$

T is set of "semantic" data-types, which are arranged in a taxonomy. Each type is associated with a set of synonyms and example values. P is a set of predicates. Predicates have typed arguments and may also have functional dependencies. S is a set of labeled services, each consisting of a set of labeled operations. Each operation has a view definition which is a conjunctive query over the predicates $P \cup \{>, \geq, =, \leq, <\}$, and is adorned with binding patterns. n is a new service, whose operations have labels associated with their inputs and outputs.

Research Plan

I intend to define a search procedure through the space of recursion and disjunction free datalog source descriptions. This work will build on Inductive Logic Programming (ILP) techniques for learning first order descriptions of functions (Quinlan 1996). The problem is complicated by the fact that in contrast to ILP, here the extensions of the domain relations are not directly accessible. Instead, the domain relations must be accessed via the sources available, making query reformulation an integral aspect of the problem. Furthermore, as is generally the case for LAV models, we cannot assume sources to be complete with respect to their source descriptions. This means that the standard "closed world" assumption used in ILP, (that any tuple not present in the examples of a relation is not present in its extension and can be used as a negative example), is not valid. Thus the system should work with positive examples alone or a more sophisticated methodology for generating negative examples must be developed.

Heuristics for guiding the search through the space of source descriptions are important as the search space is very large. The heuristics should take into account all of the information regarding the service, including both data and metadata. Finally, I will rely on techniques from record linkage (Michalowski, Thakkar, & Knoblock 2004) for testing whether two sources are producing the same tuples, and grammar induction algorithms (Lerman, Minton, & Knoblock 2003) to learn data-type descriptions from examples, for use in classifying inputs and outputs.

[1]A functional dependency was added keeping `temperature` constant over a `zip_code`.

Related Work

This work is closely related to the category translation problem defined in (Perkowitz & Etzioni 1995). My approach differs in that I focus on a relational modeling of the sources, and on inducing joins between domain relations, rather than nested function applications. Nonetheless, strategies they apply for choosing the most relevant values to give as input may apply directly to this work. This work also relates to that of (Johnston & Kushmerick 2004), who actively query sources to determine whether schema matching performed over the input/output datatypes was correct. The difference between their work and mine, is that instead of just learning how the input/output types map to some global schema, I am trying to learn the the source description itself, i.e. *how* the input and output *relate to each other* in terms of the domain model.

References

Heß, A., and Kushmerick, N. 2003. Automatically attaching semantic metadata to web services. In *IJCAI-2003 Workshop on Information Integration on the Web*.

Johnston, E., and Kushmerick, N. 2004. Aggregating web services with active invocation and ensembles of string distance metrics. In *EKAW'04*.

Lerman, K.; Minton, S.; and Knoblock, C. 2003. Wrapper maintenance: A machine learning approach. *JAIR* 18:149–181.

Levy, A. Y.; Mendelzon, A. O.; Sagiv, Y.; and Srivastava, D. 1995. Answering queries using views. In *PODS'95*, 95–104.

Michalowski, M.; Thakkar, S.; and Knoblock, C. A. 2004. Exploiting secondary sources for unsupervised record linkage. In *VLDB Workshop on Information Integration on the Web*.

Nédellec, C.; Rouveirol, C.; Adé, H.; Bergadano, F.; and Tausend, B. 1996. Declarative bias in ILP. In De Raedt, L., ed., *Advances in Inductive Logic Programming*. 82–103.

Perkowitz, M., and Etzioni, O. 1995. Category translation: Learning to understand information on the internet. In *IJCAI-95*.

Quinlan, J. R. 1996. Learning first-order definitions of functions. *JAIR* 5:139–161.

Rahm, E., and Bernstein, P. 2001. A survey of approaches to automatic schema matching. *VLDB Journal* 10(4).