Retrieving substitute services using semantic annotations: a foodshop case study

F. Calore, D. Lombardi, E. Mussi, P. Plebani, and B. Pernici

Dipartimento di Elettronica e Informazione – Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano (Italy)
XXXX, {mussi,plebani,pernici}@elet.polimi.it

Abstract. XXX

1 Introduction

The main goal of this project, was represented by the creation of a food ontology to be used for web services annotation and in order to accomplish this objective searches for a valid model to use as a reference have been performed. Eventually the ontology was built basing on the European classification of food with series of appropriate modifications and optimizations, as shown in the following. Once the ontology was completed, the subsequent task of the project consisted in using it as a semantic model to create WSDL-S starting from web services representing food warehouses, and then use a semantic search tool to test if the semantic annotation are useful to find services that fit clients needs. All of the above described tasks were aimed to achieve a small portion of the multiple goals defined in Ws-Diamond, Web Services - DIAgnosability, Monitoring and Diagnosis project [1]. In the following of this paper aspects of annotations with Radiant, the search tool and results of searches will be presented.

PARTE DA COMPLETARE (ENRICO, POI BP)

2 Food Ontology creation

Before starting to create the food ontology some searches have been carried out in order to find a valid model to be used as a reference. A first result has been a sample ontology used in the W3C [2] OWL Guide, composed of a union of two different parts, a wine and a food ontology : the former being too much detailed compared to the desired descriptive level, the latter being designed based on a different concept and therefore showing some redefinition and extension needs. The food part of this ontology was essentially made up of a single macro class divided into eight subcategories: the builders point of view had been mainly focused on describing food as part of a dish or course as defined by someone willing to create a restaurant menu. However using this ontology as a model for food, with warehouses and suppliers in mind and a property about food decay, would have been a rather complex and non-linear process, because of different concept

behind this ontology and the one needed for the project. Also, the number of individuals was too low and the number of changes and optimizations necessary to fit project needs would have been noteworthy. So it was clear that building a brand new ontology would have been a better option, although for some aspects, such as properties number or detail of the wine branch, the sample ontology was richer than the one later on created from scratch.

Searches for an ontology to be used as a model, however, made possible to find a document which has become the basis for the food ontology construction. This Document, created by the Food Safety Authority of Ireland [3] is based on European Union classification of food and describes twenty-one categories in which goods are classified, with a very large number of examples. These informations were a perfect starting point even if, during the creation of the ontology using Protg [5], changes and integration have obviously been necessary.

The twenty-one classes have been subsequently subdivided in multiples subclasses to increase the ontologys level of detail and, since it was needed to separate perishable from not perishable food, leaves containing both of them have been subdivided according to this principle. The distinction based upon food decay has been necessary because web services to be subsequently annotated in a second part of the project could have been both warehouses with non perishable food or suppliers with perishable goods. A datatype property named "isPerishable", with Boolean range, has been defined and associated with classes using the correct value, to distinguish the ones with perishable individuals from the others. This restriction and the disjoint between siblings classes entails that the ontology uses OWL DL [4] sublanguage. The ontology population has been done using the examples inside Food Safety Authority of Ireland document as main reference, with an appropriate selection after many searches on the web. Also, further searches on food producers and big food resellers websites have been made with the aim to augment the quantity of individuals, reaching this way a number of about a thousand instances or so.

In Figure 1 a global view of the ontology classes hierarchy is portrayed.

3 Semantic annotations

The purpose of semantic annotations for this project was to enhance food selling web services based on identical interfaces and differentiate them using their different kinds of goods. The following WSDL-S [6] file describes a service representing a food warehouse that can be used by actors willing to buy the same category of products offered by the selling actor; it only provides, given the nature and distinction between food based on decay , non perishable goods having "isPerishable" property set to False value and that belong to one of the ontology classes marked as non perishable. Semantic annotations have been added to every WSDL [7] entity somehow related to food, message parts and operations. The following are the most significant excerpts, message and operation parts, taken from a WSDL describing a frozen meat selling service after annotations with Radiant [8] were complete:

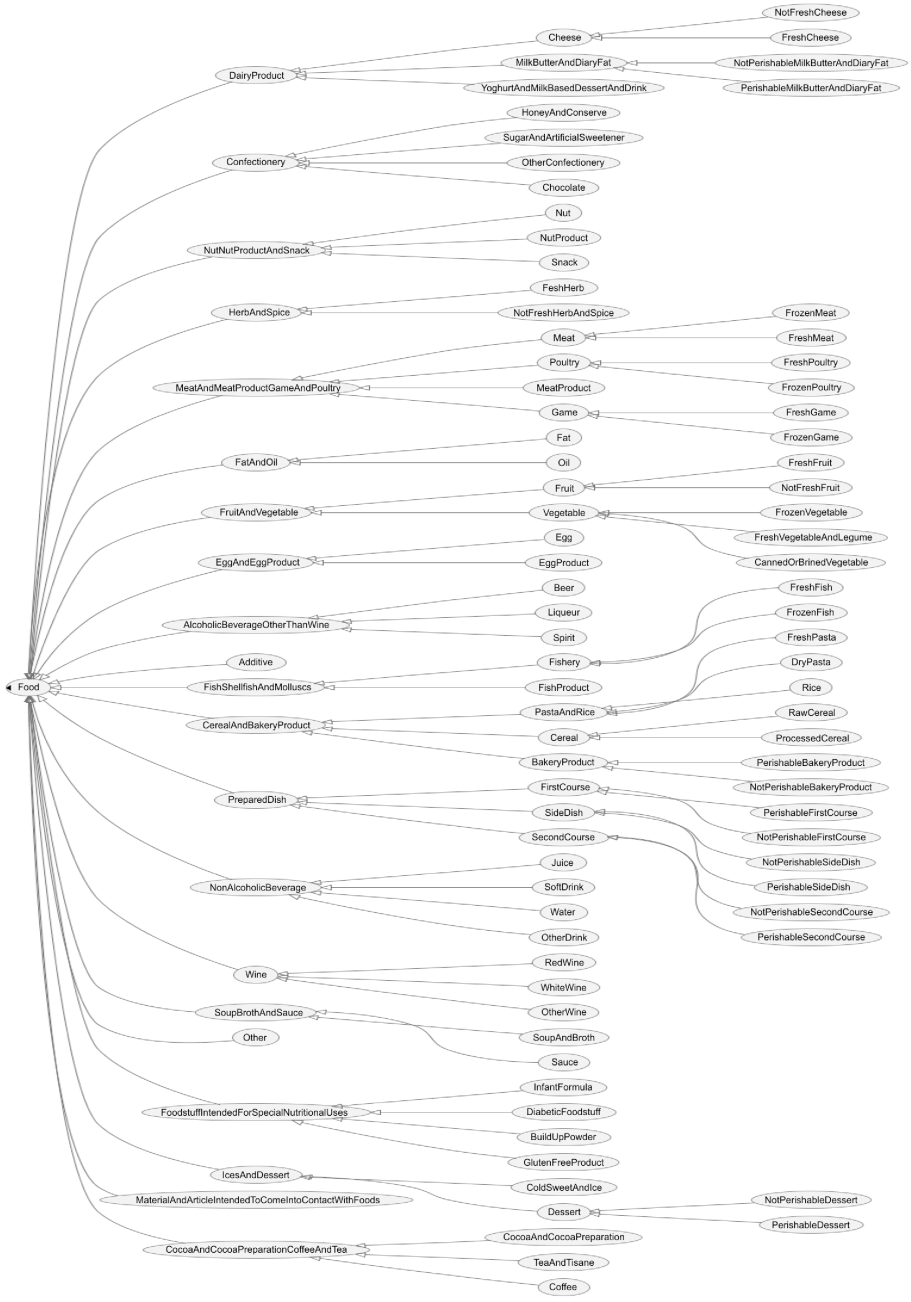

Fig. 1. Food ontology

```
<wsdl:message name="reserveRequest">
    <wsdl:part name="itemList" type="xsd:string" wssem:modelReference="Ontology5#FrozenMeat
    <wsdl:part name="customerInfo" type="xsd:string"/>
    <wsdl:part name="PID" type="xsd:int"/>
</wsdl:message>
```

This annotation of the document has been bound to the "itemList" part of message "reserveRequest". This choice is motivated because of what "itemList" represents, that is a string used by the message to hold food items while the remaining parts, and therefore the message itself from a global point of view, do not closely match with any of Foods semantic concepts. This same approach in annotating only leaves that were definitely related to food has been used for every annotation of other messages. Besides them, every operation has been annotated because, although not strictly related to food, they have at least one input or one output referred to messages containing a part semantically bound to an element of the Food ontology.

```
<wsdl:operation name="checkAvailability" parameterOrder="customerInfo itemList PID unPerisha
    <wsdl:input message="impl:checkAvailabilityRequest" name="checkAvailabilityRequest"/>
    <wsdl:output message="impl:checkAvailabilityResponse" name="checkAvailabilityResponse"/
</wsdl:operation>
```

The above is one of the operations of this WSDL-S file and, as just stated, they were annotated according to a given criterion but also to easily comply with a requirement of the adopted research tool, that explicitly needs every operation of a WSDL-S file to be annotated. This constraint could be an issue in case of different types of WSDL files, whose operations could not be annotated by simply using concepts from the same semantic model adopted for message parts.

4 Service retrieval for substitutability

Semantic annotations have been exploited during the Web service retrieval. We assume that a user request is defined using an annotated WSDL where the desired operations are listed and annotated as well as their input and output parameters. In the same way, all the available Web services are annotated in the same way as described above.

The Web service retrieval algorithm is based on a similarity distance computation. The higher a published Web service is similar to the requested one, the better the published Web service fulfill the user request.

Entering in the detail, similarity among Web services is computed comparing names and annotations at all levels in the WSDL-S description: service level, operation level and parameter level. On the one hand, name comparison relies on the assumption that all the names are included in a reference ontology. Such an ontology can be, for instance, Wordnet. On the other hand, annotation comparison relies on the same ontology adopted for annotating the Web service description.

Name similarity Given two names their similarity is returned by the function $simName(name_a, name_b)$. In detail:

- $simName = 1$ if $name_a = name_b$ or $name_a$ and $name_b$ are connected in the ontology by a "is-a" relationship.
- $simName = \frac{1}{(lengthpath(name_a, name_b)+1)}$ if $name_a$ and $name_b$ are connected by a subsumption path and $lenghtpath$ return how many hops constitutes such a path
- simName = 0 if $name_a$ and $name_b$ are not connected or there are connected by a "opposite-to" relationship.

Annotation similarity The similarity evaluation between two annotations depends on the nature of the annotations that could be terms or properties. More precisely, the $simAnn(ann_a, ann_b)$ is defined as follows:

- $simAnn = simName$ if ann_a and ann_b are both terms.
- $simAnn = \frac{max(simAnn(ann_a, term_{b,i}))}{2} \forall i \in cod(ann_b)$ if ann_a is a term and ann_b is a restriction on a property.
- $simAnn = \frac{max(simAnn(term_{a,i}, ann_b))}{2} \forall i \in cod(ann_a)$ if ann_a is a restriction on property and ann_b is a term
- in case both ann_a and ann_b are restriction on property the similarity takes into account the relationship among the restrictions:
  - if the restriction are equivalent $simAnn = 1$
  - if the properties have not any relation then $simAnn = 0$
  - if the properties have some relations:

$$SimAnn = \frac{SimProp(ann_a, ann_b)}{2} + \frac{SimName(dom(ann_a), dom(ann_b))}{2} \quad (1)$$
$$SimProp(ann_a, ann_b) = \frac{1}{level(ann_a, ann_b) + 1}$$

SimWS Given these two functions, i.e., $simName$ and $simAnn$, the similarity among Web services is obtained as the average of the similarity among operations:

$$SimWS(s_a, s_b) = \Sigma_{i=1,N} \frac{max(SimOp(op_{a,i}, op_{b_j}))}{n} \quad (2)$$

$SimOp$ returns the operation similarity which takes into account both the similarity among the operation names ($simOpName$) and the similarity among the input and output parameters ($simOpPar$)

$$SimOp(op_a, op_b) = \frac{SimOpName(op_a.name, op_b.name)}{2} + \frac{SimOpPar(op_a.par, op_b.par)}{2} \quad (3)$$

if the both requested and published operations are annotated then $simOpName = simAnn$. Otherwise the $simName$ are used to compare the names adopted to identify the operations.

About the parameters similarity the same importance has given to both the inputs and output parameters.

$$SimPar(par_a, par_b) = \frac{SimParIn(par_a.in, par_b.in)}{2} + \frac{SimParOut(par_a.out, par_b.out)}{2} \quad (4)$$

Regardless of the kind of parameter the similarity is obtained comparing the names or the annotation associated to the parameter name using the $nameSim$ and $annSim$ introduced above. Even in this case, the parameter similarity is given by the average of the maximum similarity among the parameters.

5 Analysis of Results

To test the effectiveness of semantic annotations, similar services have been annotated with links to different semantic concepts inside the food ontology. Then, with a semantic search tool created by Eng. Prazzoli [10], some searches have been done, to see if the results provided by the tool were coherent with semantically annotated web services. In particular the semantic search behaviour has been tested when services were annotated with classes related with father/child bound and with completely disjoint classes. The search tool has an input zone where a similarity threshold value can be entered. The tool uses this value to filter out results, and for these tests the similarity threshold has been set to 0.1, a low value, in order to better observe how the tool works with disjoint classes. The results obtained were good, in fact in the first case, where two services were annotated with classes related with a father/child bound, the tool retrieve both services, assigning them two different scores, the higher to the one searched and the lower to the child (or the father in a second inverted test). In the second case instead only one service has been retrieved by the tool, in accordance to the disjoint between classes and demonstrating that the tool works well.

Results obtained during tests, have been analyzed to find out how good they were using some of the measures of Information Retrieval: Precision, Recall and Fallout. Precision indicates the proportion of retrieved and relevant web services to all the web services retrieved. Once a similarity threshold has been chosen by a user, the tool filters all results that dont meet users needs and therefore it retrieves only relevant services, resulting in a Precision value of 1. Recall indicates the proportion of relevant documents that are retrieved, out of all relevant documents available and, considering what has been just said above about similarity threshold chosen by user, the tool will retrieve all relevant documents available. Fallout at last, indicates the proportion of non-relevant documents that are retrieved, out of all non-relevant documents available, which is none, as stated when talking about Precision. All these values are “too perfect” and this is because of the consideration about the similarity score chosen by

the user. Differently, considering as relevant only web services which exactly match the search, Precision value will decrease along with the decreasing of the similarity threshold and, by doing this, the value of Fallout will increase. Recall instead will maintain its value of 1 as long as the similarity threshold wont be greater than the similarity score of the relevant service, when this happens Recall will go to 0.

For the time being one of the major constraint about semantic annotations for WSDL is related to multiple annotations: apart from those services providing only one kind of food items there should be obviously some services more similar to a real scenario, that is a multiple categories seller which raises a question about how annotations could reflect this concept. There is actually a way to represent a single entity within a WSDL file using different semantic concepts but neither WSDL-S nor SAWSDL [9] declare any relation between different URIs composing a multiple annotation, even if they all have to be considered admissible. At this time the main purpose of this feature is to suggest how a single concept can be expressed, and hence referenced, by different semantic models or using different languages and, however, the adopted research tool does not currently support more than a single URI in every annotation of a WSDL document. Its hence impossible, for this project, to semantically define a web service as a multi-item store such as, for example, a meat and eggs selling service, so a choice must be done to annotate the service as a only meat or only egg seller. Doing this, the research tool will obviously not "know" when services may offer something they arent annotated with and this changes all the results obtained in the analysis above. To evaluate this aspect of a real world scenario, a sample group of services has been used, containing 2 egg selling services and 8 services selling items disjoint from eggs, among which 2 services have been considered also egg selling (but not annotated as egg selling). Research for egg selling services has been done, obtaining this way a modification of the Recall value compared with the results obtained above; Recall value this time will be indeed $2/(2+2) = 0.5$ and this result will obviously change when a different sample group of service will be used.

6 Conclusions and future work

This whole work is roughly composed of three phases: the creation of an ontology describing a specific domain, namely food; the use of such semantic model to annotate WSDL documents in compliance with WSDL-S rules and, finally, a series of test to verify actual benefits deriving from this project. Some of the different kinds of problems encountered during each phase have been quite various, such as the search of a valid model to describe foods in a hierarchical way or the choice between two software tools to annotate WSDL file; the option offered by SA-WSDL and WSDL-S, though strictly imposed by the search tool used or the analysis of this very tool created by Eng. Prazzoli to understand first problems and failures during tests. One of the major constraint, basically bound to search tool and to consequent use of WSDL-S, has been imposed by

the modelReference attributes because of no multiple annotations allowed, thus preventing a semantic description of web services selling multiple categories of food, such as bread and milk at the same time. Although most of the mentioned choices and solutions adopted during the development of this project have been quite immediate some others required deeper considerations. The main purpose of this project, which aimed at services annotation in order to improve searches and make them easier, was based on the creation of an appropriate ontology; the goal of offering web services, as adequate as possible with the actors' requirements, has been pretty much fulfilled given the results of semantic searches. This work can lead to further integrations or improvements, such as a simple switch from WSDL-S to SAWSDL for annotations; this scenario would obviously abstract away from the specific search tool used here and would not imply any modifications to the ontology, but would lead to a more recent context in semantic annotations. A second possible way of using this ontology can consist in its expansion, integration or improvement depending on requirements and specific detail level needed, for instance adding classes and/or subclasses or merging it with other akin taxonomies/ontology.