Conceptual Discovery of Web Services using WordNet

Reza Karimpour* and Fattaneh Taghiyareh+
ECE Department
University of Tehran
Tehran, Iran
*r.karimpour@ece.ut.ac.ir, +ftaghiyar@ut.ac.ir

Abstract—With the emergence of Web service mashups, selecting the appropriate Web services from the vast amount of available services has emphasized the role of service discovery. In this research, we present a new way of enhancing Web services semantically using WordNet concepts (synsets). The important advantage of our solution is that it allows developers to enhance Web services with semantic information without semantic annotation against an ontology. This is different from traditional, ontology-based researches, which require significant cost and effort for semantic annotation and ontology management. Our proposed solution allows associating semantic tags on the message parts of Web services. We solve the semantic service discovery problem with a domain neutral service annotation technique. Additionally we test the precision of system by using different similarity measures on a tagged service data set. In addition, we provide a service description binding for WADL documents.

Keywords—ReST; Semantic Annotation; Service Discovery

I. INTRODUCTION

With the emergence of Web service mashups [1] as the new breed of web applications, finding appropriate Web services has emphasized the role of semantic Web service discovery. The increasing number of Web services renders taxonomy-based service selection inappropriate where user has to spend a great deal of time browsing for proper Web services.

Being a specific form of information retrieval, Web service discovery has attracted the attention of research community. The mainstream research in service discovery uses semantic annotations in either signature matching [2] or specification matching [3]. Mostly all semantic service annotations are based on ontologies. Despite being theoretically ideal solution for semantic annotation, ontologies have some drawbacks. First, they are domain dependent. In case of mashups where services from different domains are involved, domain dependent annotations introduce ontology matching problems [4]. Second, the cost of ontologies is relatively high and requires special expertise for creation and maintenance.

WordNet [5] is aggregation of dictionary and thesaurus that is perfect for text analysis and AI use-cases. Industry and research community have been inspecting its potentials. Many has employed it for different cases like word sense clarification and text retrieval [7]. AdSense service from Google [6], for instance, uses a special version of WordNet to provide context sensitive adds to customers.

In this research, we propose a synset based semantic model for services. We specially, focus on data processing ReST Web services that mainly, produce output by processing and transforming the input. Our approach tries to inherit all the benefits of WordNet: It is not domain specific and nearly eliminates the semantic annotation cost of services. We propose a solution that uses concepts defined in WordNet to associate semantic information with service message parts. We will demonstrate how this model can be employed for discovery of services that satisfy special user requirements.

Being simple textual annotations, the concept-based descriptions can be provided for any service description standard such as WSDL [8] and WADL [9]. The presentation of the introduced enhancements in service description documents is also investigated by providing a sample for WADL documents.

In brief, the contributions of this research are:

- Proposition of a semantic Web service model based on a lexical taxonomy.
- Discovering semantic Web services using our proposed model.

In the following sections, first we review the related literature. In section 3, we describe the proposed service model followed by a service discovery system in sections 4 and 5. Thereafter we evaluate the performance of discovery process on a collection of real world Web services dataset. Finally, section 8 concludes this paper.

II. RELATED WORK

Our new model is different from other Web service models such as those based on ontologies [10,11]. The most difference important is the simplicity and minimal cost. Service models based on ontologies (like [12] and [13]), require much more time and cost for design and implementation which is not that appealing in case of ReST Web services which happen to be used mostly in B2C settings.

In [14] Eric Bouillet proposed a folksonomy based service modeling that delegates the task of service annotation to users. Although this approach reduces the service maintenance costs significantly, it introduces the possible annotation mistakes, which is common in many folksonomy-based systems. Furthermore, in contrast to our approach, the discovery
III. WEB SERVICE MODEL

We describe Web service operations by the type of messages they expect as input and the type of messages they emit in output. Our model presents the semantics of these messages in terms of WordNet concepts or synsets. Synset is the building block of WordNet. A synset contains a group of synonymous words or collocations. Different senses of a word are grouped in different synsets. The meaning of a synset is further clarified with short definitions and/or example sentences [5].

Our model is carefree regarding the inner work-flow of service operations; it only describes the inputs and outputs as WS= <I, O>. I and O stand for collection of concepts that describe inputs and outputs of WS semantically. There can be as many as needed concepts assigned to I and O collections.

A. Model Binding

In order to enrich syntactic information of Web service interfaces with synset based semantic metadata, we need to bind synsets to service message parts. While there are many service description suggestions for ReST Web services [15], by considering available developer tools, it seems that WADL is more widely adopted than the others. A WADL document is a way to describe a ReST service. It includes information about the resources, along with appropriate HTTP methods to access them. Here we demonstrate how service interfaces can be bound to WordNet synsets.

For example, consider a service that allows users to search for books by title. The following listing shows a typical service description in WADL. By sending a GET request to search resource and providing the book title, one can retrieve book information encapsulated in BOOK structure (which is defined in grammars section of WADL file but not shown here). Note how input bookquery parameter and output BOOK representation are tagged using synsets. Synsem (synset semantic) is our addition to WADL to allow binding between service parts and WordNet synsets. Each element can have one or more synsem child element describing the semantic identity of the parent element. The binding for WSDL is similar to WADL.

...<method name="GET" id="bookSearch">  
  <request>  
    <param type="xsd:string" name="bookquery" required="true">  
      <synsem>  
        <tag lexeme="title" senseNo="1" wordnetVer="3.0" />  
      </synsem>  
    </param>  
  </request>  
</resource>

IV. SEMANTIC DISCOVERY

Typically, a discovery processes starts with sending a request to service selection system and retrieving a list of best matching services is return. In our case, a request is a service description with the same model as Web service that includes tagged messages parts using concepts. Discovery phase involves calculating the similarity of all registered services to the submitted query followed by returning the most similar services as the result of discovery and selection phase.

V. DISCOVERY AND SELECTION ALGORITHM

Let ws and ws be the user query and candidate service while WS is the collection of all registered services. The function that computes the sameness of a pair of Web services is called matching utility (MU).

The matching utility function combines normalized maximum input and output similarity values using the following equation:

\[
MU(wsq, ws) = \frac{1}{1 + (iMax_{wsq, ws} + oMax_{wsq, ws})}
\]

iMax and oMax functions select the highest matching value from all possible mappings between candidate service and query. iMax function is illustrated in Figure 1.

Similarity measures compute how semantically similar, two WordNet concepts are and return a positive nonzero value. Bigger return value means concepts are semantically more similar. In the following sections, different measures used in similarity function are explained.

By evaluating and normalizing MU on query and all candidate services, the result would be \( \text{EVAL}_{wsq} \):

\[
\text{EVAL}_{wsq} = \left\{ \text{MU}(wsq, ws_1), \text{MU}(wsq, ws_2), \ldots, \text{MU}(wsq, ws_n) \right\}
\]

The services with highest values in \( \text{EVAL}_{wsq} \) are selected as relevant services that would probably match the query.

In equation (1) iMax and oMax are values grater or equal to zero while the range of MU is between 0 and 1. Greater iMax and oMax causes MU to approach 1 and smaller values causes
smaller matching utility. This normalization enables us to combine different similarity results regardless of related measures.

VI. IMPLEMENTATION

We employed our model for describing and then discovering services from a multi-domain dataset. We have compiled a set of 104 Web services from different sources like [16] and [17]. Semantic tags in form of WordNet concepts are added to all services in the dataset. 13 queries have also been prepared from various domains like telecommunication, economy, learning and traveling. Relation of every annotated service to each query was judged by checking the official documentation of Web service. In order to minimize the errors related to judgment phase, at least two votes out of three should be casted stating the relation of a query to a candidate service. If a query-service pair receives less than two votes regarding being relevant or irrelevant, we perform a deeper investigation on the pair to clarify its judgment state by inspecting the query and/or service description and documentation.

People have different point of view hence a single Web service can be tagged differently by disparate developers. For example through our observations, we witnessed that a service part that resembled a Pre-Prepared food was tagged as Frozen-Food (first sense) by one developer and Food-Product (first sense) by another. Due to this fact, we have assigned the tagging task of Web services to different people so that the final discovery evaluation would consider this fact. A team of three developers undertook the development and annotation of services and queries.

We have developed ReST Studio, which facilitates the Web service annotation by providing tools like easy WordNet navigation integrated in service annotation and discovery process (Figure 2). In addition, ReST studio is an integrated tool that can be used by semantic service annotators as well as users trying to select required services. ReST studio contains all the similarity measures as well as discovery algorithm as discussed in this paper. The tagged services dataset and ReST Studio is publicly available for download1.

VII. WORDNET SIMILARITY MEASURES

Using the discovery algorithm in section 5, we implemented a service discovery system based on various synset similarity measures provided for WordNet as a Perl module by Pederson, et al.[18]. The employed algorithms include Hirst[19], Jiang[20], Leacock[21], Lesk[22], Lin[23], Resnik[24], and Wu[25] (We refer to similarity measures with the name of first authors for the rest of the paper). The detail of these measures is out of scope of this paper and is discussed in related references; hence, we will only skim through these measures.

There has been a lot of research on WordNet similarity methods since its introduction. Synset Similarity measures can be grouped into distinct classes: “based on information content” and “based on path length” [26]. Three of previously mentioned similarity measures are based on the information content of the least common subsumer of two synsets. Information content is a particularity measure of a synset, and LCS of two synsets is the most specific synset that both synsets derive from. These measures include Resnik, Lin, and Jiang. Leacock and Wu measures are based on path lengths between a two synsets. Leacock finds the shortest path between two synsets, and scales that value by the maximum path length found in the inheritance hierarchy in which they participate. Wu calculates the depth of the LCS of the synsets, and then scales that by the sum of the depths of the individual synsets. The depth of a synset is simply its distance to the root synset, which happens to be the Entity synset.

The measure presented by Lesk is based on the extent of overlaps two synsets have in their dictionary definitions. The Hirst measure, classifies relations in WordNet as being directional, and then calculates the relatedness between two synsets by discovering a path that is neither too long nor that changes direction too often.

In order to select the most suitable similarity measure for our scenario, we will compile different discovery settings by using different similarity measures. We will focus on the precision of different discovery settings as well as the time it took to perform the calculations and return the relevance service list.

The result of different discovery settings is discussed in evaluation section.

1 http://ece.ut.ac.ir/mas/karimpour/reststudio
VIII. EVALUATION

A. Precision

We investigate the feasibility to do discovery on synset annotated message parts of Web services using service dataset and discovery algorithm. To depict the performance of our solution, we examine the precision of seven synset similarity measures on each query previously mentioned in implementation section.

We employed our service discovery system on the tagged dataset. The results are summarized in Table 1. Although synsets are less expressive than ontology concepts, we observed that the overall precision of the system is relatively high. This is because hierarchy of WordNet is constructed as human beings categorize the real world, which matches to the way developers tend to define entities in object-oriented languages.

Experiments show that different similarity measures do not has any notable impact on the final performance and all perform acceptably. All measures achieved 100% recall that means no relevant service was missed. For queries that where distinct enough, the discovery system returned exactly the relevant services, resulting in 100% precision.

Through our experiments, we came across a case where all approaches with different similarity measures rated a service to be highly relevant to a query that was judge as irrelevant by human assessor. With in depth analysis it turned out that, that specific service was actually relevant to query but due to a misjudgment by human assessor was not categorized as relevant. This can be considered an evident of the effectiveness of our service model and discovery mechanism.

While performing the examinations we also logged the time each similarity method\(^2\) required to complete a set of 100 similarity calculations. As demonstrated in Figure 3, Wu & Palmer method performs faster than others. Wu* is our custom implementation of Wu method which caches the whole collection of WordNet synsets in memory. Comparing the Wu and Wu* one can infer that similarity methods are I/O intensive and can greatly be improved by trading off memory and I/O.

Our experiments were carried out on a Quad core 2.4GHz Intel\(^{TM}\) processor running Windows\(^{®}\) XP SP3. The discovery process took 9 seconds for all 13 queries and all similarity measures. In order to optimize the similarity measurement process and overcome the WordNet::Similarity performance problems, we cached the similarity measurement results and all subsequent requests were served from cache (if available) boosting the overall performance greatly.

B. Comparison to Similar Studies

The high cost related to ontology-based solutions has motivated other researchers as well. Bouillet in [14] proposes an alternative solution based on folksonomies. In [14] service users take control of semantic annotation rather than developers which is done by assigning free form phrases to message parts. This solution has the least cost impact on developers but has some disadvantages. First, there is the risk of wrong annotation since users are less assiduous while tagging compare to the developers of the service. Second, since the proposed method requires an already available collection of

<table>
<thead>
<tr>
<th>Query</th>
<th>Hirst</th>
<th>Jiang</th>
<th>Lek.</th>
<th>Lesk</th>
<th>Lin</th>
<th>Res.</th>
<th>Wu</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
</tr>
<tr>
<td>2</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>4</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>5</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>6</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>8</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>9</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
</tr>
<tr>
<td>10</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>11</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>12</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
</tr>
<tr>
<td>13</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
<td>89%</td>
</tr>
</tbody>
</table>

\(^2\) Except Hirst
services to retrieve the hierarchical relation of tags, barrier to entry of new Web services is relatively high. In contrast, in our solution Web services are tagged by developers, which feel more responsibility toward correct tagging. Moreover, our solution can work with minimal service collection since the entry of new Web services is relatively low. In contrast, in our solution service annotation with the disadvantage of low precision and recall inherent to traditional retrieval systems.

In [27] Wang combines interface matching with traditional information retrieval on human readable description of service descriptions to discover relevant Web services. This solution minimizes the related costs by completely ignoring semantic aspect of service annotation with the disadvantage of low precision and recall inherent to traditional retrieval systems.

IX. CONCLUSION

We provided a minimal data model for processing ReST Web services. We used WordNet as a semi-ontology to semantically enhance these web services. Using different similarity measures along with our Web service model and discovery algorithm, we could achieve high discovery precision while avoiding the cost of similar solutions that are based on ontologies. WordNet facilitates cross-domain service discoveries due to its general-domain taxonomy characteristics. Being I/O intensive, we could implement a relatively quicker version of Wu similarity measure that combined by caching, eliminated the performance problem of similarity measures greatly.

The next step of this research would be the Web service composition scenario in which different services are combined together to achieve the users requirement. Furthermore, extending the service dataset to include greater number of Web services and queries would greatly benefit the evaluation results.

REFERENCES

[16] "www.xmethods.net."