Towards the Automation of the Semantic Annotation Process for Web Services

Leandro Ordóñez-Ante, Luis Antonio Rojas-Potosi, Luis Javier Suarez-Meza and Juan Carlos Corrales
Telematics Department, Telematics Engineering Group (GIT), University of Cauca, Popayan, Cauca, Colombia

Abstract- The large and growing amount of services available in the current Web has placed the need for developing efficient mechanisms for service discovery, in order to meet either a particular user request or the requirements of software agents. In this regard, a lot of work has been addressed on adding semantics over service descriptions to improve the accuracy of search engines. Nevertheless, the adoption of semantic annotation models, nowadays, has been restricted by service designers and providers, since they require certain specialized knowledge – related to formal knowledge representations –, and given that their actual implementation is very resource intensive. In this paper, we present an approach to overcome this latter issue by automating the semantic annotation process. The approach we propose, builds automatically and incrementally formal representations of knowledge from a corpus of service descriptors, by using text mining techniques and an unsupervised learning approach.

Keywords: web services, automatic semantic annotation, machine learning, LDA, FCA

1 Introduction

The Service Oriented Architecture (SOA) emerged as a means for heterogeneous, distributed and component-based applications to work seamlessly, through the definition of well-known standard interfaces, such as WSDL descriptors for REST and SOAP Web Services. This way, whenever a user or software agent requires consuming a service, it just has to know the content of the service interface to bind and implement its capabilities. However, due to the steady growth in the number of current Web resources, the search of suitable services for meeting some particular needs is an increasingly challenging task. In this regard, the scientific community behind this matter has proposed a way out to this problem, aligned with Semantic Web Technologies, conceiving what has been called Semantic Services Oriented Architecture (SSOA) [1].

The foundations of SSOA are laid on three key concepts [2]: SOA –from which its features of separation of concerns, standard interface provision, and capabilities discovery and reuse are taken–, Semantic-Based Computing –that provides sense to content, services and resources, using a formal and machine readable specification–, and Standard Based Design –that enable the integration of currently available applications with novel or future technologies–. Thus, SSOA would allow automatic service detection and selection (and consequently the automatic service discovery and composition).

Deploying SSOA requires, as stated, for resources and services to be formally specified, in such a way that a software agent can interpret and capture its functionalities in a semantic level. Nonetheless, the poor adoption of mechanisms for semantic description of services, by developers and providers (given their high cost in terms of time and resources) has inhibited the development and effective implementation of such architectures.

Researches documented in [3, 4, 5, 6] introduce proposals aimed at the integration of Artificial Intelligence technologies –specially, multiagent systems and planning- and semantic web services technologies, in order to enable automated service discovery and composition. Those works however, demand for each service the existence of two descriptors: the traditional (syntactical, e.g. WSDL) one, and one that defines its semantics (OWL-S/WSML). Given the complexity of such semantic descriptors, a large number of existing services don’t meet the requirement of these works, thus limiting their actual implementation.

In order to overcome this limitation, currently some approaches are considered to tackle the problem of semantic service annotation, by applying knowledge discovery and emergent semantics techniques over huge corpus of service descriptors, which in some cases already contains annotations made by consumers in a collaborative way. Those approaches however, have failed in leave aside human intervention and also lack of precision in search and selection processes. Therefore it’s considered necessary developing mechanisms that enable the automation of semantic service annotation tasks.

In this paper we present a research work in progress, which aims to address the stated problem by applying unsupervised machine learning techniques over a corpus of web service descriptors. This work seeks to answer the question: How to automate the semantic annotation of web services?

The remainder of this paper is organized as follows: we first outline the context into which our work is developed. Then, we describe a review of the current approaches regarding the stated problem. Next, it is depicted and defined
the architecture of a platform for enabling automatic semantic annotation of web services. Finally the conclusion and open issues of our work are addressed.

2 Background and Motivation

The work we are developing is framed around the transition between the dominant paradigm of the Web, the so-called social Web, and the establishment of the semantic Web or Web 3.0, specifically in regards to semantic annotation of services, which in turn is related to the subject of ontologies. Currently there is no generalized notion of the ontology concept; however in [7] it is formulated a conception that is widely accepted, according to which ontology “is a formal, explicit specification of a shared conceptualization”.

The semantic annotation, core concept for the current proposal, is the result of a procedure that aims to make explicit for machines, the meaning (the semantics) of content and resources available in large repositories of information. This latter constitutes one of the requirements to meet to finally materializing the Semantic Web. The semantic annotation procedure is commonly supported in formal representation of knowledge, as the aforementioned ontologies, and for services, consists in associating ontological entities to the terms defining the attributes of the service in its descriptor document [8], allowing for instance, for service search engines to effectively comprehend (on a semantic level) both the services functionality as the service’s clients requests, enabling them to accurately respond to service inquiries.

Traditionally, this semantic annotation procedure must be performed by hand by service designers and developers or in a collaborative way by service users (conceiving a sort of folksonomy of services). In both cases, the large and growing amount of services, along with the lack of knowledge regarding semantic description methods for services and the scarceness of suitable domain ontologies, has overwhelmed the human ability for performing this semantic annotation task. Additionally, the human intervention in marking up the services descriptors with ontological entities involves a very expensive process in terms of time, effort and resources.

In this regard, the focus of the present approach is on leveraging current techniques taken from the fields of machine learning, information retrieval and knowledge discovery, for automating the semantic annotation of web services. The next section will deal the revision of some previous works regarding the problem being tackled herein.

3 Related Work

This section explores some approaches that deal with semantic annotation, not only for web services, but also for content and other kinds of web resources.

In [9, 10, 11] the authors explore alternative approaches for semantic annotation of available services and resources in the Web. Such an approach consists of recognizing the information constructs from collaborative tagging systems (folksonomies) as specifications of shared knowledge, which can be suitable for semantically annotating service interfaces, dispensing with the use of ontologies. The main goal of these proposals, however, is to assist the process of semantic enrichment, still requiring human intervention (developers, users, providers, etcetera) for fulfilling the complete process.

The authors of [12] and [13] address two works regarding to semantic annotation of folksonomies, for various kinds of online available resources. In contrast to aforementioned works, the proposals of Angeletou in [12] and the one described by Siopraes in [13] argue that it is required to formalize the knowledge generated within folksonomies, by using ontologies, in order to overcome their limitations in terms of organizing, searching and retrieving resources based on tags.

The work of Angeletou differs from the current proposal, as long as the former is focused on an image folksonomy. In turn, the project addressed in [13], although it takes into account the services as part of its working resources, its scope is limited to promote collaborative tagging thereof. Furthermore, the development of that project is still in an early stage, so the results from its implementation are not yet conclusive.

The approaches outlined in [14, 15, 16, 17] pose the use of techniques of machine learning such as Formal Concept Analysis (FCA) and most recently Relational Concept Analysis (RCA), for extracting and representing the knowledge covered by documental corpus, as conceptual hierarchies or taxonomies. This way, the approaches described in these works are suitable for composing formal models of knowledge, such as core ontologies, avoiding the intervention of domain experts. However, none of the aforesaid proposals had considered the automation of such a process.

From observations made on related proposals, the present work aims to automate the process of semantic annotation of web services descriptors, through an approach that combines techniques of text mining, unsupervised machine learning (FCA) and others taken from the Information Retrieval field (Latent Dirichlet Allocation—LDA and Nearest/Normalized Similarity Score—NSS) for enabling automatic and incremental generation of formal models of knowledge from service descriptors. Such models are meant to be used in annotating and categorizing services, through a platform that implements the above techniques.

Next section will address the description of our proposal, by outlining the architecture of the platform for automatic semantic annotation of service descriptors.

4 Overview of Our Approach

According to [8] there exist four types of semantics associated with web services: data semantics –formal definition of data in input and output messages--; functional semantics –formal definition of the capabilities of a Web service--; non-functional semantics –formal definition of quantitative or non-quantitative constraints--; and execution semantics –formal definition of the execution flow of services in a Process, or of operations within a service. Our proposal is
focused on the former two types of semantics, this way, our platform enable composing two formal models of knowledge for semantic annotating both capabilities and input/output data from web services, by harvesting the information on their descriptors. Figure 1 illustrates the components that make up the annotation platform, which will be described below.

The platform we propose, receive as input a set of wsdl service descriptors. For each of these descriptors a procedure is performed in order to map its content into an abstract entity model (which is then stored in a service registry), and to extract their relevant attributes (i.e. service, operations and input/output types). Such attributes are then categorized in two arrangements: a functional taxonomy and a data (input/output) taxonomy, which are built in an incremental way as long as new services are entered to the platform or new categories are detected. The annotation consists then in automatically associating service attributes to corresponding taxonomy categories, while these latter are made up. All the outcomes of the above procedure are in turn saved in the platform storage layer.

The main components of the proposed platform are explained next:

4.1 Descriptor Processing

The purpose of this module is to extract the relevant information from incoming service descriptors. This component of the platform provides a procedure that allows abstracting the information regarding the attributes that define the service functionality (operation and input/output types) by applying a set of text mining techniques (i.e. filtering, tokenization, POS tagging) which are described and used in [17] serving a similar purpose. The output of this module is twofold: first it loads into memory the complete service descriptor, outcome that is taken by another component of the platform, in charge of mapping the content of the descriptor into an abstract service model intended for further storage and retrieval. The second outcome has to do with applying the above text mining techniques on each of the service descriptors. This output comprises the specification of tree attributes of the service: service, operation and types, along with its corresponding POS tagged values. As an example, consider a service defined by:

```
service: “CurrencyService”
operation: “GetExchangeRate”
(input)Type: “CurrencyISO”
(output)Type: “GetExchangeRateResult”
```

For the previous example, the outcome of the Descriptor Processing module will be:

```
(service, {“Currency:NN”})
(operation, {“Get:VBI”, “Exchange:NN”, “Rate:NN”})
(input, {“Currency:NN”, “ISO:SYM”})
(output, {“Exchange:NN”, “Rate:NN”})
```

Where, NN (noun), VBI (verb), and SYM (symbol) denote lexical categories (part-of-speech) for each of the words defining the service attributes. Notice that, for the service attribute, as well as for the output type, the descriptor processing module has ruled out three words: Service (for service), Get and Result (for output). This is due to these words does not have any valuable information for these particular attributes.

4.2 Service Model Mapping

This module is responsible for processing the information in service descriptors, to map it into the abstract

---

1 Example from seekda, an online registry of web services. (http://webservices.seekda.com).
service model shown in Figure 2. Such a model allows capturing the services attributes—which are specified as the UML entities WebService, Port, Binding, PortType, Operation, Message and DataElement—along with the relationships between such attributes, in order to ease its storage, search and retrieval. This platform component, takes the descriptor information loaded into memory by the previous module, and instantiates the entities, in correspondence to the referred model.

![Diagram](image)

**Figure 2. Abstract Service Model**

### 4.3 Storage Layer

This component of the architecture consists, first in a service registry, whose relational model matches the service abstract model. Thus, all of the incoming services to the platform are stored and its information is maintained for further retrieval. Secondly, this storage layer implements a mechanism for storing the artifacts obtained as outcomes from the taxonomy building module. Such mechanism allow storing both taxonomies (functional and data) as RDF graphs, as well as querying and manipulating such knowledge structures by employing the SESAME Framework along with SPARQL, the widely used RDF query language.

### 4.4 Taxonomy Building

This component takes as input the information generated by the descriptor processing module, for incrementally composing two taxonomies: one classifying the knowledge related to the terms that define the service operations (functional taxonomy), and one that is intended to arrange the knowledge about its input/output types (data taxonomy). This is a quite complex component as it is shown in the platform architecture (see Figure 1). It comprises a set of subsystems that are involved in the generation of the abovementioned taxonomies. The first of such subsystems, receives the terms supplied by the descriptor processing module and estimates their semantic relatedness relative to the concepts/categories previously classified in both taxonomies, by employing a measure of semantic relatedness (MSR) referred to as Nearest/Normalized Similarity Score (NSS), which is explained in [19]. As long as the measure of semantic relatedness between one of the terms of the service descriptor and one of the taxonomy concepts exceeds a predefined threshold, an association between these two elements (the descriptor term and the taxonomy concept/category) is placed and saved through the storage layer.

Eventually, it may not have significant similarities between the service attributes and taxonomy concepts, suggesting the income of new, uncategorized content. In that case, subsystems based on an online/incremental variant of Latent Dirichlet Allocation (LDA) [20, 21], and Formal Concept Analysis (FCA) [22] are involved. Jointly, these subsystems allow identifying additional concepts/categories, as well as their location within the taxonomy structure.

The subsystems depicted above operate over a taxonomy management component, which enables reading and updating the taxonomies and serves as mediator between both the taxonomy building and storage layer modules.

The use of FCA in approaches regarding categorization and knowledge representation of documental corpuses has spread recently. However, as argued in [23], applying this technique is constrained by the size of the corpus, due to the computational complexity that involves building the conceptual lattices, through which the information is structured in FCA. In this regard, the architecture proposed herein, applies a Latent Dirichlet Allocation (LDA) model—which is considered as an extension of the widely used PLSA in Information Retrieval applications [24]—as a mechanism for reducing the complexity linked to the generation of FCA formal contexts.

The purpose of using LDA is to understand and uncover the underlying semantic structure of a corpus of service descriptors. Through the probabilistic model proposed by [25] in LDA (which assumes each of the descriptors documents as a bag-of-words), it is possible to find out a set of categories (known as topics) covered by the descriptor documents, while associating for each document a probability distribution over all the categories/topics, i.e. a descriptor is conceived as a mixture of various topics, so that it can belong to more than one of them.

As stated before our proposal seeks to incrementally build representations of the information in the service descriptions. Thus, the traditional LDA model isn’t suitable since it requires a training step, which is performed over a whole batch of documents. That is why the abovementioned online/incremental variant of LDA is applied in this component of the architecture. Such variant of LDA enables the incremental identification of categories/topics, as new services descriptors enter to the platform.

This incremental model is applied on both the functional (service, operations) and data (input/output types) service attributes. Such a procedure generates as outcome, several ranked sets of words (corresponding to the service attributes), each of which defining a different category. The word order in each of the ranked sets is determined by the probability of occurrence of every single word in documents regarding the
category each set defines. Table 1 presents an example of the output obtained by applying the incremental LDA model, for the functional attributes of the services.

From the example it is possible distinguishing three ranked sets of attribute values, each related to one particular topic: Currency Conversion, Weather Forecast, and Geo Positioning. For the sake of space, in the example it is only considered three categories, each one with four associated service attribute values, however, there may be actually much more categories than that, then, a service could belong to multiple categories, as long as its descriptor contains terms from various categories.

One of the key benefits of applying LDA over the service descriptors entering the platform is the grouping of service attributes in the aforementioned categories, which leads to a dimensionality reduction of the space over which the next subsystem in the taxonomy building component operates: the lattice generator. Such subsystem applies FCA, a well-known lattice-based technique for knowledge representation and unsupervised machine learning, which allows identifying groups of objects sharing common attributes. The formalism posed by this technique is founded on the relation between (formal) objects and its (formal) attributes or properties, from which the triplet \( K = (G, M, I) \) is composed, referred to as formal context. In this notation \( G \) represents the set of formal objects; \( M \) denotes the set of formal attributes and \( I \) states an incidence relation between an object \( g \in G \) and an attribute \( m \in M \) ("g has m").

The lattice generator subsystem applies this FCA technique by configuring a formal context where the formal objects are the set of services entered to the platform, and the formal attributes are the categories extracted by LDA. This way, consider \( S = \{s_1, s_2, s_3, s_4\} \) a set of services and \( C = \{c_1, c_2, c_3, c_4, c_5\} \) the set of categories to which the services of \( S \) belong. The formal context \( K = (S, C, I) \) is built from making explicit the membership relation \( I \) between services and categories, which can be represented by a cross table, as shown in Table 2.

In the above formal context, the relationship between services and categories are specified by the crosses, so for example service \( s_1 \) belongs to categories \( c_1 \) and \( c_5 \), service \( s_2 \) belongs to categories \( c_1 \) and \( c_4 \), and so on.

FCA defines a derivation operation (‘) that links services and categories:

\[
A' = \{ c \in C | \forall s \in A : (s, c) \in I \}
\]

is the set of common categories shared between the services in \( A \). Similarly, having the set \( B \subseteq C \),

\[
B' = \{ s \in S | \forall c \in B : (s, c) \in I \}
\]

denotes the set of common services for the categories in \( B \). As an example, consider the formal context depicted above, being \( A = \{s_3, s_4\} \) and \( B = \{c_1, c_4\} \); then, \( A' = \{c_2, c_5\} \) and \( B' = \{s_2\} \).

From this derivation operation stems the definition of the FCA formal concept: a pair of sets \((A, B)\) is called a formal concept of the context \( K \) if \( A' = B \) and \( B' = A \), (being \( A \) and \( B \) the extent and the intent of the formal concept respectively). So for instance, in our example, the pair \( (\{s_3, s_4\}, \{c_2, c_5\}) \) is a formal concept of the Services × Category formal context, while the pair \( (\{s_1, s_2\}, \{c_5\}) \) is not.

The whole set of formal concepts of \( K \) denoted by \( \mathcal{B}_K \) are partially ordered by a specialization relation between concepts. Thus, it is said that a concept \((A_1, B_1)\) is subconcept of \((A_2, B_2)\) if \( A_1 \subseteq A_2 \). Having identified the set of formal concepts, it is possible to arrange them into a so-called concept lattice, which allows uncovering a hierarchical structure (based in the above specialization relation) of categories and the services they comprise. The concept lattice for the given formal context is depicted in Figure 3.

Table 1. Words by topic distribution obtained by applying LDA.

<table>
<thead>
<tr>
<th>Category/Topic 1</th>
<th>Category/Topic 2</th>
<th>Category/Topic 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency</td>
<td>Weather</td>
<td>Coordinate</td>
</tr>
<tr>
<td>Convert</td>
<td>Climate</td>
<td>Locate</td>
</tr>
<tr>
<td>Rate</td>
<td>Forecast</td>
<td>Distance</td>
</tr>
<tr>
<td>Exchange</td>
<td>Temperature</td>
<td>Zip</td>
</tr>
</tbody>
</table>

Table 2. Services × Category Formal Context (K = (S, C, I)).

<table>
<thead>
<tr>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( s_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( x )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_3 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_4 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c_5 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Concept Lattice for the Services × Category formal context.

A process similar as the explained above is performed for both, operations and input/output service attributes. This way, the taxonomy building module enables incrementally building of both functional and data taxonomies, while stating corresponding associations between the incoming web service attributes (operations and input/output types) and the categorized elements in such knowledge models.
5 Conclusions

Nowadays there is a huge amount of both resources and services available online, so much so that it has overwhelmed the search engines capability for meeting efficiently the queries from users and software agents.

Up to now there exists several research efforts aimed to tackle this problem, but it is largely an open question. Part of such efforts are focused on developing mechanisms for indexing and categorizing service through collaborative tagging applications, by using technologies of Web 2.0; nevertheless, the lack of ordering and limited reliability of the annotations on these resources, have hampered the benefits of such approaches. Other studies seek to leverage formal models of knowledge, such as ontologies, in order to attach semantic components on service descriptors, and thus enable automatic service discovery and composition. However, the costs involved in building and maintaining these formal models, as well as in the semantic annotation process had constrained, not only the actual implementation of these approaches, but also the deployment of the Semantic Web.

In this paper, we have introduced a novel and suitable approach to overcome the problem of semantic annotating web services. Our work focuses on the extraction of relevant information about services attributes, by applying some text mining techniques on their descriptors, in order to compose – in an automatic and incremental way – formal models of knowledge regarding those attributes. Such models are intended to specify two types of semantics of web services: the functional (operations) and the data (input/output types) semantics. The referred formal models are generated as conceptual taxonomies by defining hierarchical relationships between concepts, applying an online LDA model along with a FCA technique. To the best of our knowledge, the joint use of these two latter on tackling the problem we have stated, has not been addressed in previous works, thus it is one of the main contributions of our research. LDA allows uncovering the underlying semantic structure of a corpus of service descriptors as a distribution of relevant categories/topics, while FCA enables identifying a hierarchical structure of categories and the services.

This way the proposed approach merges both formal knowledge models generation, and a fully automatic semantic annotation method for web services.

The platform that implements this proposal also includes a registry where processed services are stored (once they have been mapped into an abstract model), as well as the above formal knowledge models as RDF documents.

The proposed approach is under development and as complementary work it is considered taking into account already annotated services, (i.e. services holding a semantic descriptor, such as OWL-S or WSML), in conjunction with web ontologies in order to enhance both the concepts and relationships of formal knowledge models, as the semantic annotations on services, which involves the use of semi-supervised machine learning methods.

6 Acknowledgments

The authors would like to thank University of Cauca, Colciencias and TelComp2.0 project for supporting this paper.

7 References


ICSOC/ServiceWave 2009 Workshops, Lecture Notes in Computer Science.


