A topic modeling approach for web service annotation

Leandro Ordóñez-Ante[*]
Ruben Verborgh[**]
Juan Carlos Corrales[***]

Recibido: 27/09/2013 • Aceptado: 17/01/2014

Abstract

The actual implementation of semantic-based mechanisms for service retrieval has been restricted, given the resource-intensive procedure involved in the formal specification of services, which generally comprises associating semantic annotations to their documentation sources. Typically, developer performs such a procedure by hand, requiring specialized knowledge on models for semantic description of services (e.g. OWL-S, WSMO, SAWSDL), as well as formal specifications of knowledge. Thus, this semantic-based service description procedure turns out to be a cumbersome and error-prone task. This paper introduces a proposal for service annotation, based on processing web service documentation for extracting information regarding its offered capabilities. By uncovering the hidden semantic structure of such information through statistical analysis techniques, we are able to associate meaningful annotations to the services operations/resources, while grouping those operations into non-exclusive semantic related categories. This research paper belongs to the TelComp 2.0 project, which Colciencas and University of Cauca founded in cooperation.

Key words: semantic web, web services, topic modeling, knowledge representation.

[*] Magister(C) en Ingeniería Telemática, Investigador del Grupo de Ingeniería Telemática de la Universidad del Cauca, Colombia, Popayán – Cauca, Carrera 17 # 19N-166B, Tel: (+57-2) 8364921, email: leandro@unicauca.edu.co

[**] Ph.D en Ciencias de la Computación, Investigador del Grupo Multimedia Lab de la Universidad de Gante, Bélgica, Ghent, Gaston Crommenlaan 8/102, 9050, Tel: (+32) 0473514847, email: ruben.verborgh@ugent.be

[***] Ph.D en Informática, Profesor Titular del Departamento de Telemática de la Universidad del Cauca, Colombia, Popayán – Cauca, Edificio de Ingenierías sector Tulcán, Oficina 403, Tel: (+57-2) 8209800, ext. 2129, email: jcorral@unicauca.edu.co

Un enfoque basado en modelos temáticos para la anotación semántica de servicios

Resumen

En términos prácticos, la implementación de mecanismos de recuperación de servicios basados en sem**ántica** ha sido limitada, debido al costoso procedimiento que involucra la especificación formal de servicios. Este procedimiento comprende una tarea dispendiosa de anotaci**ón semá**ntica, la cual se lleva a cabo manualmente por desarrolladores de servicios, quienes, además, deben conocer modelos para la descripción semántica de este tipo de recursos (p. ej. OWL-S, WSMO, SAWSDL). Para superar esta limitación, este artículo introduce una propuesta para la anotación de servicios web, basada en el procesamiento de su documentación disponible, para extraer la información relacionada con las capacidades que estos ofrecen. Al descubrir la estructura semántica oculta de dicha información, a través de técnicas de análisis estadístico, el mecanismo propuesto es capaz de asociar anotaciones relevantes a las operaciones/recursos de los servicios, así como agruparlos en categorías semánticas no exclusivas. Este artículo de investigación está enmarcado en del proyecto TelComp 2.0, financiado por Colciencias y la Universidad del Cauca.

Palabras clave: web semántica, servicios web, modelos temáticos, representación de conocimiento.

INTRODUCTION

The Web is emerging as an instrument for connecting distributed applications, becoming—more than an information system—into a platform that supports the operation of a huge service ecosystem built under different architectures and design philosophies [1]. Leonard Richardson has proposed in [2] a schema for classifying services on the Web, defining three maturity levels (plus a zero level). Each one of these four levels represents one element of what Richardson calls *the technology stack for web services*: URI, HTTP, and Hypermedia. This way, services are classified according to the technologies that supports their operation:

- *Level zero services:* services at this level are characterized by having a unique URI and using only one HTTP method (typically POST). At this level, we find the XML-RPC services [3] and most of the SOAP services.
- *Level one services (URI support):* At this level, services employ various URIs, but only one HTTP method. In contrast to level zero services, services at level one expose multiple logical resources. However, services at this level tunnel their actions by inserting parameters and values into a URI, which is then transmitted to a remote service (via HTTP GET). According to Richardson, most of the services out there that *claim to be RESTful are actually level one services*.
- *Level two services (HTTP support):* this level deals with services that host many resources, each of which is addressable through its own URI and supports various HTTP methods. This level includes CRUD-like services, such as the Amazon cloud storage system (Amazon S3: http://aws.amazon.com/es/s3/).
- *Level three services (Hypermedia support):* At this level, we find real RESTful services: Those having the features of *level two* services, and additionally support the notion of *hypermedia as the engine of application state* (HATEOAS), that is to say, the representations of the resources hosted by the service contain controls that enable consumers to access related resources. Examples of this kind of services include the Web and the REST API of Netflix (http://developer.netflix.com/docs/REST_API_Conventions).

A research conducted by Maleshkova et al. [4] reports that, despite the apparent spreading of RESTful services in the Web, there are actually few services that supports all the tenets and constraints of REST. The results derived from their analysis, evidence that only 32% of services could be considered — at least nearly — REST services (i.e., services from *levels two* and *three* in the Richardson maturity model), while the remaining 68% was RPC and hybrid services (i.e., services from *levels zero* and *one*, according to the same model).

The study of Maleshkova also states that particular customs and norms rather than well-established standards and rules are preferred when it comes to service development. Similarly, service documentation (specifically REST service documentation) is not supported on interface description languages such as WSDL (for SOAP services), but it is provided as HTML pages with no regular or standard structure. Therefore, the use of web services requires a cumbersome manual process, which further hinders the execution of discovery, composition and invocation procedures. In this regard, some initiatives have been devoted to the definition of standard formats for describing REST services. That is the case of WADL (*Web Application Description Language*) [5], a language intended for specifying HTTP-based web services and applications.

Just like WSDL for SOAP services, WADL enables automatic building of services clients, making them easier to consume and accessible to developers. Nonetheless, WADL descriptors merely describe the static view of services and applications, neglecting the user-resources interaction dynamics, which is better specified by *hypermedia* and *media types*. Consequently, as stated in [6], this kind of descriptor is suitable only for CRUD-like REST services (*level two* services) whose functionality often is limited to manipulating records from a dataset.

So far, the adoption of WADL as a language for describing REST services has been scarce. WSDL, on the other hand, has been largely the standard format for describing SOAP Web services; however, since the *IBM*, *Microsoft* and *SAP* public UDDI nodes went down in 2006, efficient mechanisms for discovery and composition of this kind of services have been missing.

Over the last decade, the academic community has been working around the definition of new formats for describing Web services, which include semantic metadata, aiming at enabling automatic discovery and composition of services. In this regard, researchers have come up with proposals like *hRESTS* [7] — along with its extensions *SA-REST* [8] and *MicroWSMO* [9] — and *RESTdesc* [10] for semantically describing REST services/APIs; and languages such as SAWSDL [11], OWL-S [12], and WSMO [13], intended for extending the WSDL descriptor of SOAP services with semantic metadata.

In general, there is a sort of barrier regarding the adoption of new formats for specifying the Web service semantics. Bearing this in mind, the proposal we introduce in this paper intends to use the information currently available in service description documents — i.e., WSDL interfaces for SOAP services and HTML documents for XML-RPC and REST services — in order to derive a knowledge representation based on the content of such documentation. This knowledge representation specifies a set of conceptual categories, used then for classifying and annotating Web service operations and resources. Our proposal lies on three main processes:

1. Extraction of technical information related to service functionality.
2. Analysis of the extracted information for identifying conceptual categories and the services they comprise.
3. Deriving a RDF - encoded taxonomy from the categories obtained in process (2).

The approach we propose contributes towards automating the process of semantic annotation of Web services descriptors, by combining techniques of text mining — specifically tailored to Web services documentation — and unsupervised machine learning methods (i.e. Latent Dirichlet Allocation - LDA) for enabling *automatic* and *incremental* generation of a formal model of knowledge from publicly available service documentation sources. This way we are able to build a dynamic knowledge representation structure that gets updated as new services are analyzed. The remainder of this paper proceeds as follows: Section 1 outlines our proposal for enabling automatic annotation of web services based on their available documentation. Section 2 deals with the experimental evaluation we conducted for verifying the feasibility of our approach. Finally, in section 3 we sum up and give our conclusions and directions for future work.

1. PROPOSAL

This section addresses a detailed description of the foundations of our proposal. First, we outline the analysis of Web services documentation, then the probabilistic topic model used for deriving the hidden semantic structure from such documentation is introduced, and finally we address the formal model for representing such a semantic structure.

1.1 Analysis of Web Service documentation sources

Web service documentation is often limited to the content that API developers provide on their websites [4]. Following subsections deal with the description of mechanisms for extracting technical information regarding service functionality from various documentation sources: WSDL descriptors for SOAP services, and HTML documents for XML-RPC and REST services.

1.1.1 SOAP Services

WSDL is an XML standard format for Web service description. A WSDL document describes the SOAP service interface abstractly and provides concrete technical details about service operation. The later refer to elements that specify service endpoints, and communication and transport protocols used for message exchange. These concrete details are required for service invocation; however, they provide little information about service functionality. That is why descriptor analysis focuses on the components

of the abstract description of the service interface, namely *Types* (*schema*, *element*, *complexElement*, *and sequence*), *Message*, *PortType*, and *Operation*.

Typically, there is some redundancy in the information contained in certain elements of service interface. Thus, for instance, terms defining ports, bindings and portTypes are frequently the same used for describing the service element; likewise terms defining input/output messages, are slight variations of the term specifying their associated operation. In consequence, it was decided that information extraction from service descriptor only takes the content of *service*, *operation* and *types* elements into account, including their natural language descriptions (when available in their *documentation* tag). Frequently, the terms used in defining such elements of the WSDL descriptor follow naming conventions adopted by programmers, e.g. using CamelCase compound words for identifying operations, types and services. Similarly, documentation tags that developers use for describing service attributes in natural language often contain HTML encoded data. Therefore, it is necessary to get the content into a proper format to enable further processing. This involves the use of basic text mining techniques such as *tokenization*, *POS* (*part-of-speech*) *tagging* and *spell checking*.

1.1.2 XML-RPC and REST Services

As mentioned at the beginning of section 1.1, Web service documentation—except for SOAP services—does not meet any standard format. XML-RPC and REST services are commonly described by HTML pages, which provide information regarding service functionality and endpoints[1]. Usually the content of such pages does not follow any formal structure, making it difficult to extract relevant information in an automated way. There are some initiatives, including *ProgrammableWeb*[2] and *APIhub*[3], which promotes the creation of centralized API directories, where service documentation is uniformly stored, by following a regular structure. However, a major issue regarding these initiatives has to do with the manual entering of service documentation into these platforms, so more often than not the information they provide either contains errors (typos, broken links) or is outdated.

Given this limitation, it was decided to deal with documentation that developers provide on API websites. This way, we apply on each HTML page an analysis that involves identifying recurring patterns (which depends on the service type, either XML-RPC or REST) and document segmentation for extracting relevant information regarding service functionality. This analysis is supported on the approach proposed by Ly et al. in [14], which allows identifying and extracting operation description blocks

[1] For example: (XML-RPC) http://www.benchmarkemail.com/API/Library (REST) https://dev.twitter.com/docs/api/1.1

[2] Available at: http://www.programmableweb.com/

[3] Available at: http://www.apihub.com/

from XML - RPC services (comprising textual information about service operations, the data they receive and deliver, and their natural language description), and resource description blocks from REST services (which include URI templates, HTTP methods and a natural language description).

1.2 Discovering the semantic structure of Web service documentation by applying Probabilistic Topic Modeling

Over the last decade a diversity of statistical models known as probabilistic topic models have emerged as powerful tools that allow uncovering the hidden thematic structure that runs through a document collection [15]. The work addressed in this paper aims at applying a process of categorization on the textual information extracted from a corpus of Web service description documents, by using an online variation of the *Latent Dirichlet Allocation* topic model, which enables the incremental categorization of a continuous stream of documents. In this particular case, documents contain information regarding each operation or resource hosted by a collection of services. In the following section, the description of the LDA topic model and how it was adopted within this research is addressed.

1.2.1 The Latent Dirichlet Allocation (LDA) topic model

LDA is one of the most basic and widely adopted probabilistic topic models for identifying the abstract themes that pervade a documentary corpus [15]. In statistical terms, LDA in one of the so-called generative models, which allows sets of observations (i.e. documents) to be explained by hidden or latent variables (topics).

The intuition behind LDA is to assume that documents in the corpus deal with not only one, but several topics. This paper for example is about *Web services*, *Semantics*, *Topic models*, and *REST*. LDA estimates that documents are generated through the random process outlined below (see figure 1):

- LDA assumes:
- Each topic is a distribution over the vocabulary composing the whole document collection. Thus, for instance, in the "*Web service*" topic (the blue one in figure 1), terms like "*SOAP*", "*REST*", and "*XML - RPC*" would have high probability of occurrence.
- The corpus has a finite number of topics (*K*), which are specified before documents have been generated.
- Word order in each of the documents is not relevant: the documents are modeled as *bags-of-words* [17].

- The generative process for each document in the collection is as follows:

  1. Randomly choose a distribution over the topics, as the topic proportions for each document (represented in figure 1 by the three bar histogram).

  2. for each word in the document:

     (a) Randomly choose a topic from the distribution drawn in step #1.

     (b) Randomly choose a word from the topic (distribution over words) drawn in step #2a.

According to the above process, all the documents in the corpus share the same set of topics, but each one of them has different topic proportions, given by the distribution assigned in step #1. This way the generative process satisfies the intuition behind LDA by conceiving the documents as a mixture of topics.

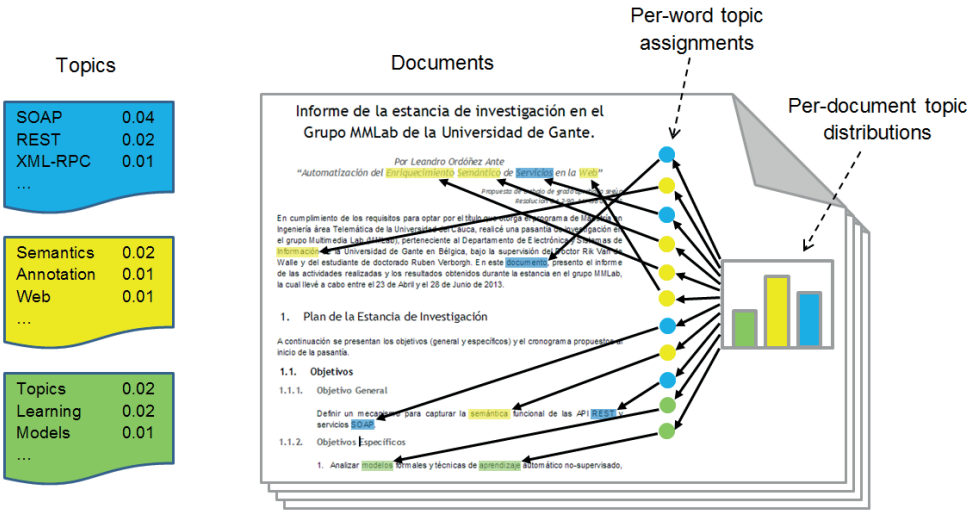

Figure 1. Generative process for LDA.

Source: Adapted from [16]

Algorithm 1 below, formalizes the generative process of the LDA probabilistic model.

Algorithm 1. LDA Generative Process

Data: K : number of topics in the document colection, α : parameterizes the per-document topic distribution, η : parameterizes the per-topic term distribution.

Result: Collection of D documents, each one containing N terms.

/* For each topic, draw a distribution over terms */

For each $k \in \{1,2,\ldots,K\}$ **do**

$\beta_k \sim Dirichlet(\eta)$;

End

For each document $d \in \{1,2,\ldots,D\}$ **do**

/* draw a topic distribution */

$\theta_d \sim Dirichlet(\alpha)$;

For each term n **in** document d, $n \in \{1,2, ..., N\}$ **do**

/* draw a topic from the distribution θ_d */

$z_{d,n} \sim Multinomial(\theta_d)$;

/* draw a word ($w_{d,n}$) from the distribution over terms $z_{d,n}$ */

$w_{d,n} \sim Multinomial(\beta_{z_{d,n}})$;

End

End

The probability distributions assumed by this generative process—i.e., topics (β_k, in algorithm 1), per-document topic proportions (θ_d), and per-word topic assignments ($z_{d,n}$) — are unknown a priori; the only observable variable is the collection of document ($w_{d,n}$). These distributions define hidden variables, which could be inferred by "reversing" the LDA generative process, allowing estimating the hidden thematic structure that generates the observed document collection. Formally, this generative process is represented by the joint distribution of the hidden and observable variables:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d)\, p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right) \quad (1)$$

From the joint distribution in Eq. (1) it is possible to derive the conditional distribution of β, θ, z given w (the *documents*) using the *Bayes theorem*:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} \mid w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \quad (2)$$

Topics in LDA may be defined briefly as categories comprising semantically related elements. This categorization, as opposed to traditional clustering techniques (like *K-means*) is non-exclusive since each document in the collection may belong to multiple categories[4]. This way, LDA enables exposing meaningful relations inside the collection not only at the documents level but also at the topical level too.

The approach documented herein, involves the application of the LDA model on the information extracted from the Web services documentation (according to the process outlined in section 1.1), for automatically categorizing and annotating it. Given that the service documentation focuses on describing operations (for SOAP and XML - RPC services) and resources (for REST services), the information extraction mechanism we propose generates one document for each service operation/resource. This way, the categorization is performed at the operation/resource level.

One remarkable feature of our proposal has to do with the incremental categorization process it performs; so that the derived structure of categories gets updated as new documents (service operations/resources) become available, thus operating in an *online learning* setting. In order to support such online setting, we implement a *variational Bayesian* algorithm proposed by Hoffman et al. in [18], to estimate the posterior distribution in equation (2) from a stream of documents handled in small-sized batches.

In section 2 we describe a prototype based on our proposal, which uses this *online LDA* algorithm for categorizing and annotating the documentation of a corpus comprising 200 real service descriptors from the research dataset collected by Zhang et al. [19] available at http://www.wsdream.net/dataset.html

1.3 KNOWEB-S: an RDF taxonomy for representing the derived service categorization

The third of the processes that shape our proposed solution, involves the construction of an RDF-encoded taxonomy that formally specifies the category structure generated by applying online LDA on Web services documentation. We named this taxonomy *KNOWEB-S* for *KNOwledge representation for WEB Services*. The section below describes the specific data model of KNOWEB-S, designed to represent the entities and relations that shape the taxonomy.

1.3.1 KNOWEB-S Data Model

The KNOWEB-S taxonomy arranges a set of categories that cluster semantically related documents (Web services operations or resources). In turn, each category comprises

[4] Theoretically, all documents belong to all categories but only few categories have a meaningful proportion of each document content.

a set of terms, which defines its meaning. In this way, the entities and relations that make up the KNOWEB-S taxonomy are as follows:

- *Entities*:
  - *Document*: consists of information regarding a Web service operation or resource.
  - *Category*: groups related documents.
  - *Term*: defines a semantic unit associated to a particular category.
- *Relations*:
  - *Is member of:* between *Document* and *Category*.
  - *Has term*: between *Category* and *Term*.

According to the LDA model used for performing the categorization, a document may belong to multiple categories provided its per-category proportions. Therefore the "*is member of*" relation has many-to-many (N:M) cardinality, and is weighted by the proportions of categories of each document, turning this into a non-binary relation. Similarly, the "*has term*" relation between categories and terms is also non-binary, since it has many-to-many cardinality, and provides a probability value that estimates the relevance of individual terms to each category.

Since RDF schema properties are binary relations, i.e. those that link two entities or an entity and a value, dealing with the non-binary relations of KNOWEB-S implies the definition of two new entities and four new relations:

- *New entities*:
  - *Membership relation*: defines a link between a Document and a Category while specifying the probability associated to it.
  - *Term relation*: defines a link between a Category and a Term while specifying the relevance of the term to the category.
- *New relations*:
  - *Category value:* between *Membership relation* and *Category*.
  - *Membership probability*: between *Membership relation* and a numerical value.
  - *Term value*: between *Term relation* and *Term*.
  - *Term probability*: between *Term relation* and a numerical value.

Finally, figure 2 illustrates the final entity-relationship diagram of KNOWEB-S adapted to RDF Schema:

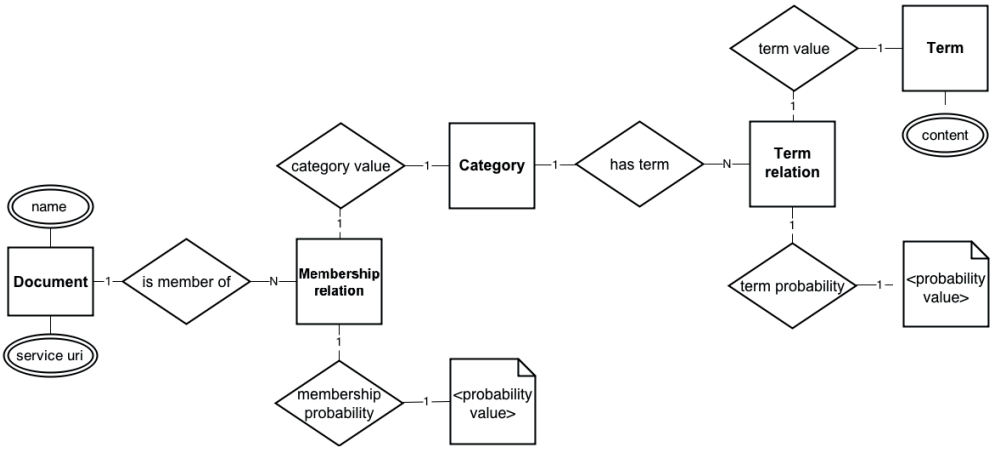

Figure 2. Entity-relationship diagram of KNOWEB-S adapted to RDFS.

Source: Own elaboration

2 EXPERIMENTATION

In order to estimate the feasibility of our proposal we built a prototype called *Topicalizer*,[5] which implements the mechanisms described throughout this document and runs through a public research dataset of SOAP services descriptors.

The prototype receives as input a list of URIs from real WSDL interfaces available online. The system retrieves each WSDL, processes it by following the techniques outlined in section 1.1.1, and stores its relevant information into a service registry. In running this process, the prototype generates a stream of documents, each one of them containing information related to a specific SOAP service operation. Then the stream of documents is categorized based on the document's content by applying the *online LDA* algorithm. Such categorization arranges semantically related documents into clusters defined by a weighted set of terms, which in turn become annotations on the operations each document represents. The information gathered from the categorization process is specified in RDFS—conforming to the KNOWEB-S data model defined in section 1.3.1—and stored into an RDF triplestore.

Figure 3 illustrates the system architecture.

For evaluating our proposal, we have developed a web application[6] intended for human evaluators to visualize and browse through the structure of categories that pervade a corpus of 200 SOAP service descriptors (which add up to 1328 operations)

[5] A description of the source code structure of the prototype is available at https://github.com/LeandroOrdonez/Topicalizer.

[6] *TopicalizerBrowser*, available at http://leandrocloud.cloudapp.net:8080/TopicalizerBrowser

extracted from the research dataset by Zhang et al. [19]. We asked the evaluators to estimate the relevance of the annotations that our prototype assigns to each individual operation by approving or disapproving them, and by adding new terms that they consider the prototype has missed.

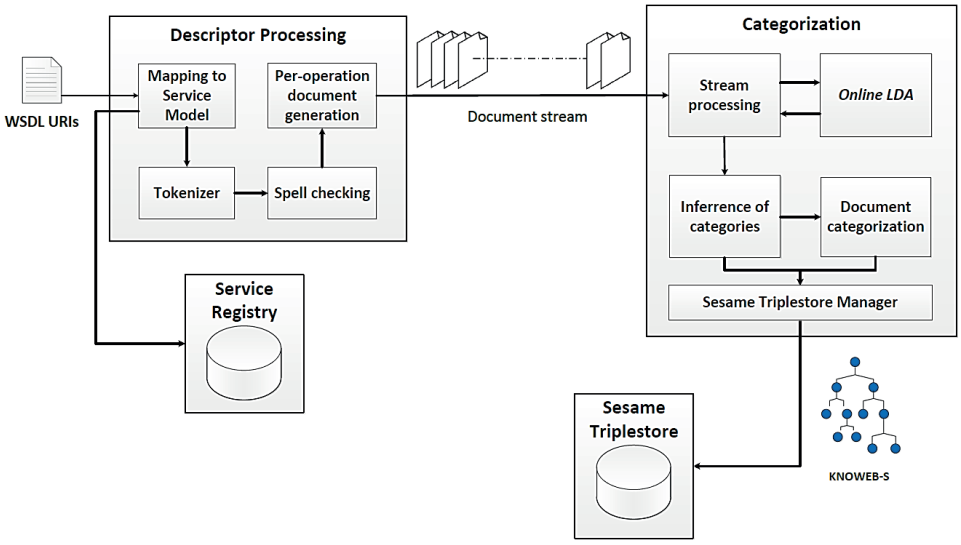

Figure 3. Architecture of Topicalizer.

Source: Own elaboration

We had three evaluators taking part on this process. They went through this procedure on 11 randomly selected categories out of the 40 available, performing the evaluation of 165 operations (that is 15 operations per category).

Based on the results gathered from this evaluation procedure, we estimate the performance of our proposal by taking three measurements on each of the operations reviewed by the human evaluators: p*recision (P)*, *recall (R)* and the b*alanced F-measure* (F_1) [20] according to the following expressions:

$$P_{op} = \frac{1}{|E|}\sum_{e=0}^{|E|}\frac{|AA_e|}{|RA_e|+|AA_e|} \tag{3}$$

$$R_{op} = \frac{1}{|E|}\sum_{e=0}^{|E|}\frac{|AA_e|}{|MA_e|+|AA_e|} \tag{4}$$

$$F_{1op} = \frac{2P_{op}R_{op}}{P_{op}+R_{op}} \tag{5}$$

where AA_e, RA_e and MA_e refer to the sets of accepted, rejected, and manual (missing) annotations for evaluator *e* respectively, while $|E|$ represents the number of evaluators.

When computing the overall precision, recall and F-measure by summing over the entire set of evaluated operations *op*, the average value for each measurement were respectively 81.48%, 90.48% and 85.45%, outperforming similar approaches, like the one proposed by Falleri et al. in [21]. Figure 4 provides a detailed view of the results in terms of the precision and recall of individual operations in the dataset. This graph evidences the capability of our proposal for assigning a *comprehensive* (high recall) set of *relevant* (high precision) annotations to each operation in the dataset since all data points lie in the upper right corner of the precision-recall chart.

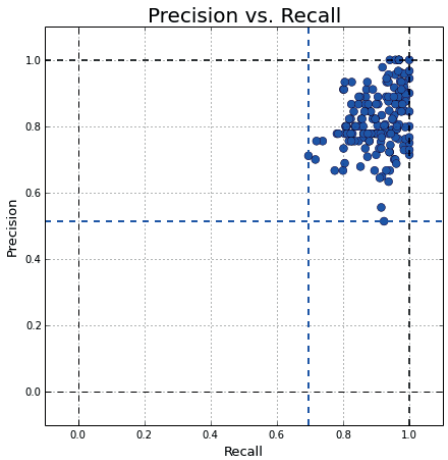

Figure 4. Precision vs. Recall for individual operations in the dataset.

Source: Own elaboration

Figure 5 depicts the average value of precision, recall and F-measure per category. Additionally this chart shows the set of terms defining each one of the evaluated categories. According to this graph, our system is able to identify semantic related tokens out of the content of web services descriptors, while properly classify service operations within the categories each set of terms defines, with an F-measure ranging from 79.24% (category 3) to 95% (category 8).

The results we got from the performance evaluation evidences the feasibility of our approach for classifying and annotating web services supported on probabilistic topic models. It is worth noting that even though the evaluation was performed on SOAP services—given the availability of reliable WSDL datasets—the proposed mechanism

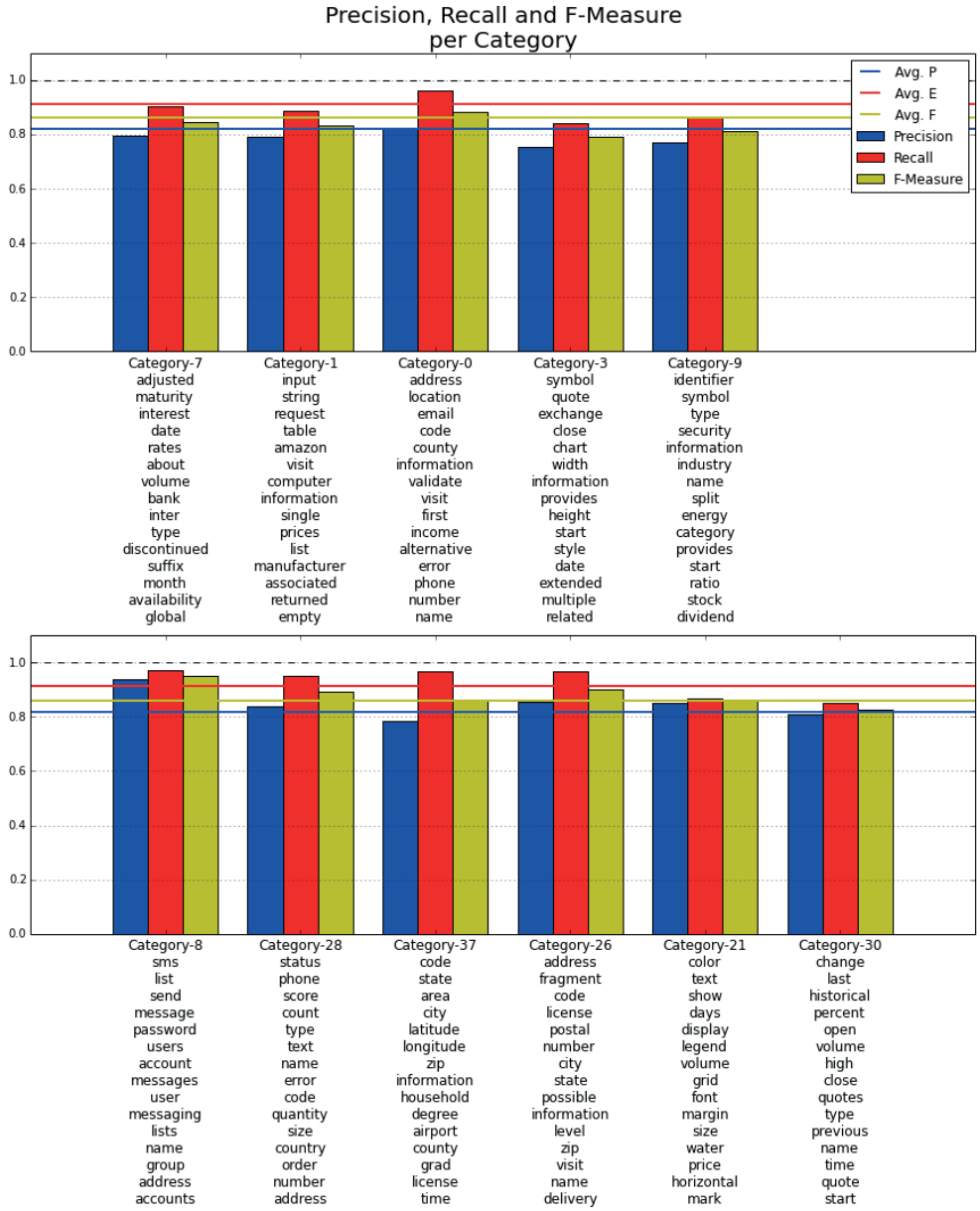

Figure 5. Precision, Recall and F-measure per Category.
Note that the list of terms defining each category is placed under the Category IDs.

Source: Own elaboration

would be also suitable for XML-RPC and REST-like web services, as long as their documentation is available in a semi-structured format (e.g. HTML documents). Upon extracting the raw text documents describing operations and resources hosted by XML-RPC and REST services (by applying the procedure outlined in section 1.1.2) the mechanism we proposed for incremental categorization and annotation operates regardless the kind of service the documents are coming from.

3. DISCUSSION AND CONCLUSIONS

The research work documented in this paper, proposes an approach that leverages on existing Web services and their associated documentation sources for generating a knowledge representation, which captures the semantics that defines them in a machine-readable format. Such a knowledge representation allows arranging the services into a structure that reveals semantic relationships among them.

The knowledge representation is derived by applying an online variant of the LDA topic model on the information extracted from different Web services documentation sources. This model allows deducing a set of categories that cluster semantically related service operations and resources. The derived structure of categories is specified by using a standard format based on the RDF data model, and stored into an RDF triplestore.

Finally, the proposed techniques supported the implementation of a prototype for categorizing a set of operations hosted by SOAP services available online. The performance evaluation made on this prototype—in terms of precision and recall measures—demonstrates the feasibility of our approach for incrementally categorizing and annotating web services based on their publicly available documentation.

The knowledge representation derived from analyzing the service documentation provides a starting point for a wide range of applications supported on the identified semantic related categories. For instance, it is possible to conceive a mechanism for Web services search and retrieval on top of the KNOWEB-S taxonomy, leveraging the semantic relationships between services inferred by applying the LDA topic model on their documentation sources. Both the categorization and the semantic annotation that our proposal performs, would allow reducing the search space, and delivering highly relevant results to user queries.

Further work in our research aims to improve the performance of the proposed mechanism by using more advanced topic models such as *Hierarchical Dirichlet Processes (HDP)* or *Correlated Topic Models*. Future work will also involve the adoption of existing vocabularies such as *SKOS* and *DBpedia* for enriching the *KNOWEB-S* model defined in section 1.3, and the construction of a platform for service discovery based on the annotations generated by the system we introduced in this paper.

4. ACKNOWLEDGEMENTS

The authors would like to thank University of Cauca, Colciencias and TelComp2.0 project (Code: 1103-521-28338 CT458-2011) for supporting the research of the M.Sc. Student Leandro Ordóñez-Ante.

REFERENCES

[1] J. Webber, S. Parastatidis, and I. Robinson, "The web as a platform for building distributed systems," in REST in Practice, O'Reilly Media, 2010.

[2] L. Richardson, "Justice will take us millions of intricate moves," 2008.

[3] S. S. Laurent, E. Dumbill, and J. Johnston, *Programming Web Services with XML-RPC*. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2001.

[4] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating Web APIs on the World Wide Web," in Web Services (ECOWS), 2010 IEEE 8th European Conference on, pp. 107-114, 2010.

[5] M. Hadley, "Web Application Description Language," 2009.

[6] J. Webber, S. Parastatidis, and I. Robinson, "The Web and and WS-*," in *REST in Practice*, O'Reilly Media, 2010.

[7] J. Kopecký, K. Gomadam, and T. Vitvar, "hREST: An html microformat for describing RESTful web services," in Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '08, (Washington, DC, USA), pp. 619-625, IEEE Computer Society, 2008.

[8] A. P. Sheth, K. Gomadam, and J. Lathem, "SA-REST: Semantically interoperable and easier-to-use services and mashups," IEEE Internet Computing, vol. 11, pp. 91-94, Nov. 2007.

[9] J. Kopecký, D. Fensel, and T. Vitvar, "D3.4.3 microwsmo and hRESTs," 2009.

[10] R. Verborgh, T. Steiner, D. Deursen, J. Roo, R. V. D. Walle, and J. Gabarró Vallés, "Capturing the functionality of web services with functional descriptions," Multimedia Tools Appl., vol. 64, pp. 365387, May 2013.

[11] Farrell, J. and Lausen, H. (2007). Semantic annotations for wsdl and xml schema.

[12] Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). OWL-S: Semantic markup for web services. Website.

[13] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. Appl. Ontol., 1(1):77–106.

[14] P. A. Ly, C. Pedrinaci, and J. Domingue, "Automated information extraction from Web APIs documentation," in Proceedings of the 13th International Conference on Web Information Systems Engineering, WISE'12, (Berlin, Heidelberg), pp. 497511, Springer-Verlag, 2012.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," J. Mach. Learn. Res., vol. 3, pp. 993-1022, Mar. 2003.

[16] D. M. Blei, "Probabilistic topic models," Commun. ACM, vol. 55, pp. 77-84, Apr. 2012.

[17] C. D. Manning, P. Raghavan, H. Schütze. *Language models for information retrieval* in "*Introduction to Information Retrieval*" (pp. 237-252). Cambridge University Press, New York, NY, USA.

[18] Hoffman, M. D., Blei, D. M., and Bach, F. R. (2010). Online Learning for Latent Dirichlet Allocation. In NIPS, pages 856–864.

[19] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSExpress: A QoS-aware search engine for web services," in Proc. IEEE Int'l Conf. Web Services (ICWS'10), pp. 8390, 2010.

[20] Cunningham, e. a. (2011). Text Processing with GATE (Version 6). University of Sheffield, Department of Computer Science.

[21] J.-R. Falleri, Z. Azmeh, M. Huchard, and C. Tibermacine, "Automatic tag identification in web service descriptions." in WEBIST (1) (J. Filipe and J. Cordeiro, eds.), pp. 40-47, INSTICC Press, 2010.