Semantic Annotation of SOAP Web Services based on Word Sense Disambiguation Techniques[1]

Anotación semántica de servicios web SOAP basada en técnicas de desambiguación lingüística[2]

Ángela de La Cruz-Caicedo[3]
Yonatan Bolaños-Bastidas[4]
Leandro Ordóñez-Ante[5]
Juan Carlos Corrales[6]

doi:10.11144/Javeriana.IYU18-2.sasw

How to cite this article:
LA CRUZ-CAICEDO, A. de; BOLAÑOS-BASTIDAS, Y.; ORDÓÑEZ-ANTE, L., and CORRALES, J. L. Semantic Annotation of SOAP Web Services based on Word Sense Disambiguation Techniques. *Ingeniería y Universidad*. 2014, vol. 18, no. 2, pp. 369-392. http://dx.doi.org/10.11144/Javeriana.IYU18-2.sasw

[1] Reception date: March 3rd, 2013. Acceptance date: July 31st, 2014. This article is derived from the research project *Automating semantic annotation of converged services based on linguistic disambiguation of their descriptor*, developed by the research group GIT of University of Cauca, Popayan, Colombia.

[2] Fecha de recepción: 3 de marzo de 2013. Fecha de aceptación: 31 de julio 31 de 2014. Este artículo se deriva de un proyecto de investigación denominado *Automatización de la anotación semántica de servicios convergentes con base en la desambiguación lingüística de su descriptor*, desarrollado por el grupo de investigación GIT de la Universidad del Cauca, Popayán, Colombia.

[3] Engineer on Electronics and Telecommunication, University of Cauca, Popayan, Colombia. E-mail: apdelacruz@unicauca.edu.co.

[4] Engineer on Electronics and Telecommunication, University of Cauca, Popayan, Colombia. E-mail: ylbastidas@unicauca.edu.co.

[5] Engineer on Electronics and Telecommunication, University of Cauca, Popayan, Colombia. Msc. in Telematics Engineering, University of Cauca. Popayán, Colombia. E-mail: leandro@unicauca.edu.co.

[6] Doctor of Philosophy (PhD) in Informatics. Tenured professor, Department of Telematics, University of Cauca, Popayan, Colombia. E-mail: jcorral@unicauca.edu.co.

Abstract

A key requirement for materializing the Semantic Web involves the annotation of resources and Web services with semantic metadata. This procedure is traditionally addressed as a manual task, which involves a high consumption of time and resources as well as the expertise on description formats and formal representations of knowledge, such as ontologies. Some research has promoted the development of mechanisms that partially automate the semantic annotation procedure, however, for the services particular case, those works lack of an analysis of the linguistic context of descriptor documents or interfaces, which provide adequate assignment of semantic annotations on the functional attributes of the services. In this context, this paper introduces a mechanism to automate the semantic annotation of SOAP services, supported by Word Sense Disambiguation techniques (WSD), from which it is possible to link the context of descriptor documents to the procedure of identification and association of ontological entities related to service attributes. This document discusses the mechanism described above, by developing an example, as well as the results of the experimental evaluation performed on a prototype that implements the proposal.

Keywords

web services; semantic annotation; ontologies; word sense disambiguation (WSD)

Resumen

Un requerimiento clave para materializar la web semántica involucra la anotación de los recursos y servicios web con metadatos semánticos. Este procedimiento se ha abordado tradicionalmente como una tarea manual, la cual implica un consumo elevado de tiempo y recursos, así como el conocimiento de formatos de descripción y representaciones formales de conocimiento, como las ontologías. Algunos trabajos de investigación han promovido la generación de mecanismos que automaticen de manera parcial el procedimiento de anotación semántica; sin embargo, en el caso particular de los servicios, estos trabajos prescinden de un análisis del contexto lingüístico de las interfaces o documentos descriptores que facilite la asignación adecuada de anotaciones semánticas sobre los atributos funcionales de los servicios. En este marco, el presente trabajo introduce un mecanismo para automatizar la anotación semántica de servicios SOAP, sustentada en técnicas de desambiguación del sentido de las palabras (WSD), a partir de las cuales es posible vincular el contexto de los documentos descriptores al procedimiento de identificación y asociación de entidades ontológicas relacionadas con los atributos del servicio. Este artículo aborda la descripción del mecanismo mencionado, apoyándose en un ejemplo, así como en los resultados de la evaluación experimental realizada sobre un prototipo que implementa la propuesta.

Palabras clave

servicios web; anotación semántica; ontologías; desambiguación del sentido de las palabras (WSD)

Introduction

The semantic annotations of currently available services on the web for easing processes of discovery and composition is an essential activity for the materialization of semantic web. Nowadays there is a huge amount of both resources and services available online, so that they have overwhelmed the search engines capability for effectively meeting the queries from users and software agents. The semantic web development and its introduction in service-oriented architectures, enables applications to be able to discover, invoke, compose and monitor in an automatic or semiautomatic, way web services that respond to particular needs. The information about the functionality of each web service is encoded in documents called service descriptors, which allow their discovery and retrieval (Cardoso, Miller & Emani, 2008).

The semantic annotation of web service descriptors allows automated agents to reason about the capabilities offered by the services and make decision regarding discovery and composition tasks, avoiding the complexity that those processes involve for users or developers. Thus, the amount of time required for service creation and deployment (*time to market*) tends to reduce, encouraging the creation of new value added services (Osman et al., 2006).

There are several research efforts regarding the semantic annotation process of service descriptors. However, while there exist approaches proposing mechanisms and tools for assisting the process of annotation (in a *semi-automatic setting*), this is commonly carried out on each of the attributes of the service in an isolated way, neglecting their meaning in the linguistic context configured by the rest of the elements comprising the service descriptor (Aksoy et al., 2011). Additionally, this semantic annotation process involves several task performed by hand by service designers and developers, turning it into a convoluted and error-prone process (Asswad et al., 2011).

The approach proposed by Chabeb et al., and documented in (Chabeb, 2009), conceives a SAWSDL extension called YASA4WSDL (*Yet another Semantic Annotation for WSDL*). According to the authors of this work, YASA4WSDL allows describing service attributes in terms of the concepts of two kinds of ontologies: *Technical ontologies* —comprising concepts describing functional and non-functional attributes of web services— and *Domain ontologies* —that incorporate concepts defining the business domain semantics, for example, tourism, health, etc. Nonetheless, in YASA4WSDL, both the association of ontological entities to the services attributes, and the selection of the ontologies used in the whole process, are procedures that have to be performed by hand by the users.

In contrast to Chabeb's et al. work in Bouchiha and Malki (2012) propose an approach for semi-automated annotation of WSDL descriptors. The authors of this paper conceive a tool for allowing the user to provide the WSDL descriptor to be annotated along with a set of ontologies. This tool executes two processes for deciding which of the provided ontologies delivers relevant semantics to the WSDL descriptor: (1) *Categorization*, in charge of classifying the WSDL descriptor according to its knowledge domain and defining which ontology is aligned with the specified domain, and (2) *Comparison*, which associates concepts of the chosen ontology to the elements of the WSDL descriptor. In this work and others related approaches like the *METEOR-S* platform (Verma et al., 2005) developed inside the LSDIS group (*Large Scale Distributed Information Systems*) of the University of Georgia, even though part of the annotation process is automated, the intervention of the user is still required for ensuring the appropriate annotation of the service descriptor. Additionally, the user must provide the ontology or sets of ontologies whose entities (concepts/instances) then become the semantic annotations, by performing first a time-consuming process of search.

In view of the above, this paper introduces a novel platform for supporting the semantic annotation process of Web and Telecommunication services (Telco) based on *Word Sense Disambiguation* (WSD) techniques, used for identifying the linguistic context of the service descriptors in order to enable the automatic association of ontological entities to functional attributes of the services. This paper also presents the results of the experimental evaluation performed on the platform we built, evidencing the feasibility of our approach for automating the semantic annotation of the web services and Telco.

The platform proposed in this paper supports the association of ontological entities available on the web to each of the services' functional attributes (*service*, *portType* and *types*), taking into account the meaning of those attributes within the linguistic context set by both the service descriptor and the ontology. In order to identify such contexts, we used WSD techniques enabling the proper assignment of senses to polysemic words according to the context they are in. Once the correct meanings are found, by measuring the semantic relatedness among them, it is possible to associate the ontological entities to the attributes of the service descriptor, thus fostering a high semantic correspondence between the content of the descriptor and the annotations attached to it.

In the next section, a description of the platform we built for automating the semantic annotation of SOAP services (Web and Telco) is presented. The third section deals with the results of the experimental evaluation we performed on the proposed platform. Finally, the last section addresses the main conclusions derived from the research we conducted.

1. Materials and Methods

Web and telecommunications services are important elements for developing component-based distributed systems and convergent services. For both the Web and Telecommunications domains, different architecture models promoting the deployment, publication and consumption of services have been defined: the world of web services has been largely dominated by the SOAP and WSDL standards; however, a resource-oriented paradigm known as REST has been widely adopted in recent years. On the other hand, in the telecommunications domain there is no standard language or framework for describing services. The most representative efforts in this regard are OneAPI and Parlay X, specifications based on Web standards and technologies which allow the formally description of telecommunication services capabilities through WSDL interfaces (Jie et al., 2009; Smith, 2009). Given these reasons and considering that version 2.0 of the WSDL standard supports the description of REST and SOAP services, we have decided to work with WSDL 1.1 and WSDL 2.0 service descriptors, this way comprehending the description of Web (SOAP and REST) and Telecommunications services. Next section introduces a brief description of the WSDL language.

1.1. Web Service Description Language (WSDL)

WSDL is a recommendation from the W3C's *Web Services Description Working Group*, and has become a standard for the definition of web service interfaces. WSDL enables to describe service interfaces regardless of the underlying technology supporting their operation. In June 2007, the W3C released the WSDL 2.0 (W3C, 2007) recommendation, in which it has been incorporated a number of improvements over the language, from its previous version WSDL 1.1. WSDL 1.1 describes a web service through six main components (Khalid et al., 2010):

- *types*: describing the data types used in the interchange of messages when invoking a service operation. There are simple types (one data element) and complex types (a set of simple types).
- *messages*: these are abstract representations of the transmitted information. Typically, a message comprises one or more logical parts (parameters), for instance a message of a purchase order comprises the items being ordered, the price for each item, etc. These logical parts are defined through simple/complex *types*.
- *portType*: this component defines the combination and sequence of messages for each of the abstract *operations* (functions) hosted by the web service. Each operation description comprises an *input* message, an *output* message and optionally an *error/fault* message.
- *binding*: this component specifies the communication protocol and data format of each *operation* and *message* defined within a *portType* element.
- *port*: defines an endpoint by specifying a unique address for a *binding*.
- *service*: this element represents a composite *operation*, aggregating multiple related *ports*.

1.2. Automatic Annotator of Convergent Services (AA-CS) Platform

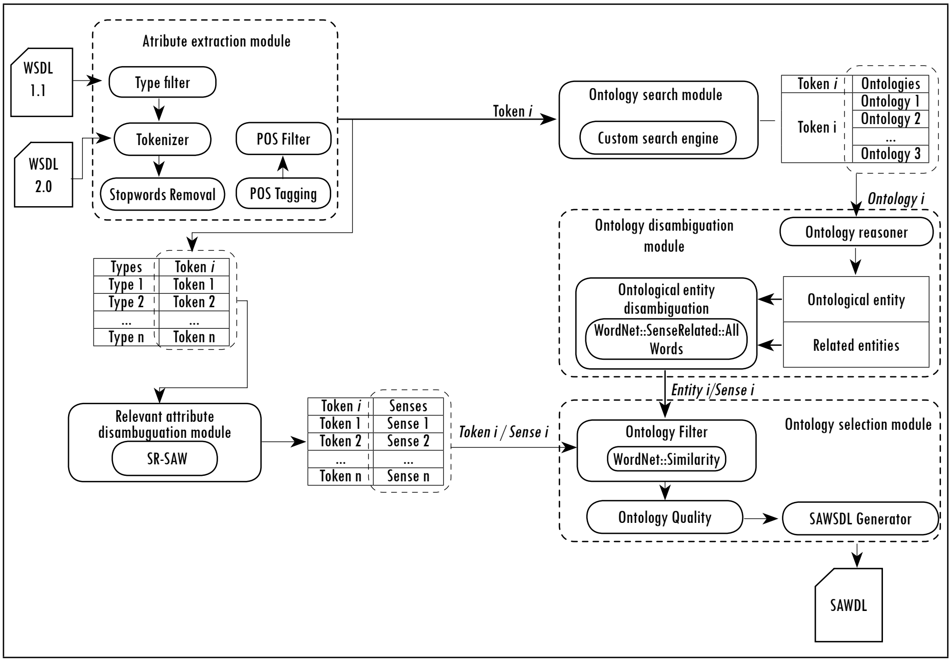
The AA-CS platform receives as input WSDL service descriptors (Christensen et al., 2001). The platform provides an application that allows the user to specify the URI of the service descriptor he/she intends to annotate; the platform retrieves the descriptor by issuing an HTTP request to the supplied URI, starting then the process of semantic annotation. Finally, the platform generates a file holding the annotations following the SAWSDL (*Semantic Annotations for WSDL and XML Schema*) standard format (Farrell et al., 2007). Figure 1 illustrates the components of the platform arranged in six modules. The subsections below deal with the description of each one of these six modules, by specifying their operation on a WSDL document belonging to a sample web service (*Global Weather Service*) presented in Figure 2.

1.2.1. Attribute Extraction Module

This module identifies the relevant terms of the service descriptor, namely those capturing the service functional attributes. The attribute extraction module is based on the process proposed by Falleri et al. (2010), which comprises the following phases:

1.2.1.1. Type Filter: this sub-module extracts the descriptor attributes, and arranges them in pairs {*type*, *identifier*}, where *type* indicates the attribute container element (i.e. *Service*, *Port*, *PortType*, *Message*, *Type*, or *Binding*) and *identifier* is the attribute value extracted from the WSDL document. For instance, from the attribute value *MyGlobalWeatherService* contained in the service label, the pair {*service*, *MyGlobalWeatherService*} is set.

Figure 1. AA-CS Platform

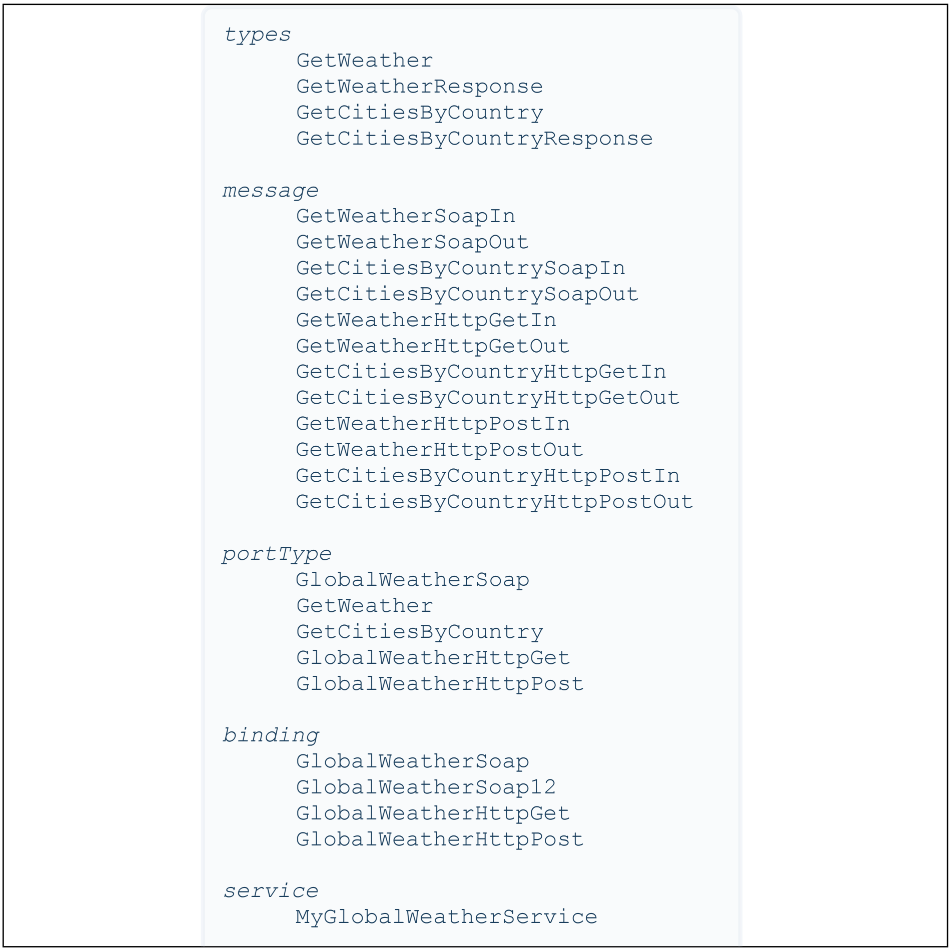


Source: authors' own presentation

This module finally discards those pairs where *type* is *Port*, *Message* or *Binding* since the identifiers (attribute values) of these WSDL elements are usually the same used as identifiers for the remaining tags.

1.2.1.2. Tokenization: Frequently the information provided in WDSL descriptors follows naming conventions adopted by programmers, e.g. using CamelCase compound words for identifying operations, types and services. *Tokenization* refers to the procedure that allows obtaining a set of tokens or terms composing a sequence of characters. This way, the information extracted by the *type filter* module is tokenized for generating a list of words comprising operation identifiers and data elements. So, for example when tokenizing the sequence of characters "*MyGlobalWeatherService*", this module obtains the following set of terms: {"*My*", "*Global*", "*Weather*", "*Service*"}. In the previous example, the procedure for splitting the composed term consists of detecting the alternate use of upper and lower case letters along the sequence of characters. This way, each pair {*type, identifier*} obtained by the previous sub-module is replaced by the pair {*type, tokens*}, where *tokens* represents the set of words *(token*$_1$*, token*$_2$*, ..., token*$_n$*)* comprising the *identifier* element. For the example above, the *tokenization* module obtains: {*service, (my, global, weather, service)*}.

Figure 2. Descriptor of the Global Weather Service



```
types
      GetWeather
      GetWeatherResponse
      GetCitiesByCountry
      GetCitiesByCountryResponse

message
      GetWeatherSoapIn
      GetWeatherSoapOut
      GetCitiesByCountrySoapIn
      GetCitiesByCountrySoapOut
      GetWeatherHttpGetIn
      GetWeatherHttpGetOut
      GetCitiesByCountryHttpGetIn
      GetCitiesByCountryHttpGetOut
      GetWeatherHttpPostIn
      GetWeatherHttpPostOut
      GetCitiesByCountryHttpPostIn
      GetCitiesByCountryHttpPostOut

portType
      GlobalWeatherSoap
      GetWeather
      GetCitiesByCountry
      GlobalWeatherHttpGet
      GlobalWeatherHttpPost

binding
      GlobalWeatherSoap
      GlobalWeatherSoap12
      GlobalWeatherHttpGet
      GlobalWeatherHttpPost

service
      MyGlobalWeatherService
```

Source: authors' own presentation

1.2.1.3. Stop-words removal: **this sub-module gets rid of those tokens whose use in attribute identifiers is** too frequent to be meaningful. A list of stop words has been defined for each one of the service functional attributes specified in a WSDL document, namely:

- *service*: service
- *portType*: port, response, request
- *type*: type

In this way, for the previous example, the pair {*service, (my, global, weather, service)*} drops to {*service, (my, global, weather)*}.

1.2.1.4. Part-Of-Speech (*POS*) *Tagger*: This sub-module takes each pair {*type,* (*token*$_1$, *token*$_2$, …, *token*$_n$)} obtained by the *stop-words removal* module, and replaces each *token*$_i$ with the pair *{token*$_i$*, pos*$_i$*}*, being *pos*$_i$ the part-of-speech of *token*$_i$. The possible values for the *pos*$_i$ element are *NN: noun, NNS: plural noun, NP: proper noun, PP: pronoun, NPS: plural proper noun, JJ: adjective, JJS: adjective plural, VV: verb, VVG: gerund verb VVD: verb in past tense* and *SYM: symbol.*

This way, given the pair {*service, (my, global, weather)*}, the POS tagger module obtains as outcome: {*service, ({my, PP}, {global, NN}, {weather, NN})*}.

1.2.1.5. POS Filter: service attributes values often consist of verbs and nouns (e.g. *getWeather*) and occasionally contains adjectives and symbols (e.g. *getWeather&Presure*, or *getGlobalWeather*) (Ly, y otros, 2012). That is why this *POS filter* sub-module removes the pairs [*token*$_i$*, posicion*$_i$] where *posicion*$_i$ ∉ (*NN*, *JJ*, *VV*, *SYM*).

For the example above this sub-module obtains the pair {*service, ({global, NN}, {weather, NN})*}. So, in conclusion, the terms "*global*" and "*weather*" are the *tokens* specifying the functional semantics of the *service* attribute element.

Finally, the set of tokens the *attribute extraction module* obtains when applied over all the functional attributes of the *global weather service*, comprises the terms {"*get*", "*weather*", "*cities*", "*country*", "*global*"}.

1.2.2. Attributes Disambiguation Module

This module is responsible for finding the right sense for each *token*$_i$ gathered from the *attribute extraction module*, by applying WSD techniques. This procedure sets each *token*$_i$ as *target token*, estimating its sense in function of the remaining tokens, which are regarded as its context. Each *target token* has {S_1, S_2,…, S_{mt}}

senses, and $\{C_1, C_2, \ldots, C_n\}$ *tokens* that make up the context. At the same time, each term of the C_i context has $\{S_{i1}{*}, S_{i2}{*}, \ldots, S_{in}{*}\}$ senses.

Using the *WordNet::SenseRelate::AllWords* (SR-AW) (Pedersen et al., 2009 and 2005; Pedersen & Michelizzi, 2004) tool, this module measures the semantic similarity (R_{ki}) between each pair $\{S_k, S_{in}\}$ being S_k the *k-th* sense of the target *token* and S_{in} the *n-th* sense of a *token* in the context window. Finally, all the R_{ki} of each S_k are aggregated and the sense with the highest value of R_{ki} is assigned to the *target token*.

The process of disambiguation of each *token*$_i$ is conducted upon the descriptor's linguistic context. The subsections below describe in detail the whole disambiguation process.

1.2.2.1. WSD technique for service attributes: **this technique performs** the disambiguation of service attributes, supported on the *WordNet* lexical reference system (Miller, 1995). The SR-AW tool was originally developed for assigning the most appropriate sense to each word within a set of terms, according to the tokens they are surrounded by (*context window*). In the approach we introduce hereby, the set of words this tool receives as input corresponds to the *tokens* gathered by the *attribute extraction module*, which in turn become the linguistic context of the service descriptor upon which the disambiguation process is performed.

The disambiguation algorithm denotes the *tokens* in context window as: $\{t_1, t_2, \ldots, t_n\}$, where t_{target} ($1 \leq target \leq n$), is the *target token* (*token* being disambiguated). The platform sets the context window depending on the number of *tokens* gathered from each service descriptor. For example, a context window of size 5 comprises two words at the right of the *target token*, two words at the left of the *target token*, and the *target token* (Pedersen et al., 2005). Assuming that each *token* t_i has *m* possible senses, denoted by $\{s_{i1}, s_{i2}, \ldots, s_{im}\}$, the set of possible senses of a *target token* would be $\{s_{target1}, s_{target2}, \ldots, s_{targetm}\}$. One of these senses is then selected as the most appropriate sense for the *token* t_{target}.

Algorithm 1 formalizes the process of service attribute disambiguation. First, one of the input terms is defined as *target token*, while the remaining terms configure its context and the score of each sense of the *target token* is set to zero ($score_i = 0$). In order to obtain the most suitable sense for each of the attributes, the algorithm estimates the semantic relatedness between the sense $s_{targeti}$ of the *target token*, and each sense s_{jk}. The algorithm computes the $s_{targeti} \times s_{jk} \rightarrow \mathbb{R}$ relationship for each t_j t*oken*, where $s_{targeti}$ and s_{jk} represent two senses of a pair of *tokens* in the context window. In this way, the measure of semantic relatedness receives two senses ($s_{targeti}$ and s_{jk}) as input and generates as output a real number that quantifies the semantic similarity between them. This relatedness value is assigned to *temp-score*$_k$, and then the highest score is selected (*best-score*). Subsequently, the algorithm aggregates the *best-score* of each *token* in the context window, turning this into the score for the sense $s_{targeti}$ (*score*$_i$) of the *target token*. Finally, the sense $s_{targeti}$ holding the highest score is established as the right sense for the *target token.*

Algorithm 1. ***Token*** **Disambiguation**

```
Input: context[], target: context and target token (to be disambiguated)
Output: $s_{targeti}$, most related sense to the target token
for each token t in context[]
    for each sense $s_{targeti}$ of target token, where $i=0..N$
        let $score_i = 0$
        for each token $w_j$ in context[]
            next if $j = t$
            for each sense $s_{jk}$ of $w_j$
                $temp\text{-}score_k = relatedness\ (s_{targeti}, s_{jk})$
            $best\text{-}score = max(temp\text{-}score)$
            $score_i = score_i + best\text{-}score$
return $s_{targeti}$ so that $score_i > score_j$ for all $S_{targetj}$ in $\{s_{target1}, ..., s_{targetN}\}$
```

Considering the *Global Weather* sample service, let's suppose that it is required the disambiguation of the term "*weather*" (*target token*), given a context window consisting of {"*country*", "*get*", "*cities*" and "*global*"}. Thus, assuming that the *weather* token has five senses, the algorithm 1 computes the semantic relatedness between *"weather"* and each sense of the terms in the context. Thus, for instance, given the first sense of "*weather*" ($S_{target1}$) being:

$S_{target1}$: "*the atmospheric conditions that comprise the state of the atmosphere in terms of temperature and wind and clouds and precipitation*",

And the first sense of the term "*country*" being:

S_{11}: "*a particular geographical region of indefinite boundary (usually serving some special purpose or distinguished by its people or culture or geography)*", the semantic relatedness between them is 0.585, such value is assigned to the *temp-score*$_k$ variable.

Assuming this similarity value being higher than the semantic relatedness scores obtained for the remaining pairs ($S_{target1}$, S_{1k}), then the algorithm sets the best-score value to 0.585. Subsequently, the *best-score* is added for the $S_{target1}$ sense of "*weather*" (*score*$_{1}$).

Lastly, assuming that *score*$_{1}$ is higher than the rest of *score*$_{i}$ values (the score associated to the four remaining senses of "*weather*"), the algorithm assigns $S_{target1}$ as the most suitable sense for the *target token* "*weather*", according to the terms comprising its context window.

The outcome of this *attribute disambiguation module* consists of a set of *tokens* belonging to the service descriptor (e.g. "*country*", "*get*", "*cities*", "*global*" and "*weather*"); each one associated to its most suitable sense, according to the terms that compose its context window.

The *time complexity* of this disambiguation algorithm may be estimated by assuming S_{avg} as the average number of senses for each of the terms registered in the *WordNet* dictionary. Thus, for estimating the similarity between each sense of the *target token* and each sense of the terms composing the context window, (S_{avg} x S_{avg}) = S_{avg^2} comparisons are required. This way, for a context consisting of *n* terms, *n* x S_{avg^2} comparisons should be made. The algorithm performs this procedure for each one of the S_{avg} senses of the target *token* and each of the *tokens* context window comprises. Therefore, the number of comparisons performed by the attribute disambiguation algorithm is about n^2 x S_{avg^3}.

While S_{avg^3} affects the number of comparisons the disambiguation algorithm made, this factor is constant and independent from number of attributes being disambiguated. Consequently, it is possible to state that the execution time of algorithm 1 is proportional to the square of the context window size (n^2).

1.2.3. Ontology Search Module

This module integrates the *Google Custom Search* API (Google, 2012) enabling the AA-CS platform to use *its search functionalities. By using this API, the platform is able to retrieve ontologies available on the Web whose classes, properties or instances match the terms (tokens) gathered by the attributes extraction module*. In order to do this, a query is issued against the Google API (via HTTP GET), by specifying a service attribute value and setting a file extension restriction (using the *filetype* operator), so that the search results contain only OWL documents. The API returns a JSON document holding a collection of URLs pointing to the location of the matched ontologies.

Since this search engine classifies the most relevant results in the top places of the ranking (Lewandowski, 2010), the *ontology search module* takes only the first ten entries of the group of URLs the API retrieves. Figure 3 shows the collection of ontologies (URLs) matched for three *tokens* belonging to the *Global Weather* service descriptor (Figure 2).

Figure 3. Ontology Search

weather	http://www.scs.ryerson.ca/~bgajdero/msc_thesis/code/ontologies/weather-ont-t2.owl
	http://research.ict.csiro.au/conferences/ssn/EventOntology_no_imports.owl
	http://zaltys.net/ontology/AKTiveSAOntology.owl
	http://www.anusuriya.com/sego/SEGOv3.owl
	http://semanticscience.org/ontology/sio.owl
	http://www.ontologyportal.org/SUMO.owl
	http://ontologydesignpatterns.org/ont/fao/asfa/asfad.owl
	https://www.seegrid.csiro.au/subversion/xmml/metadata/ISO19115/iso-19115.owl
	http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1.owl
	http://www.ontologyportal.org/WordNet.owl
cities	http://www.semanticbible.com/2004/04/NTNames.owl
	http://mets.egovservices.net/onts/2010/12/GeoPolitical.owl
	http://www.semanticbible.com/2004/09/NTNames.owl
	http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_4.owl
	http://www.mindswap.org/2003/owl/geo/geoFeatures20040307.owl
	http://users.ecs.soton.ac.uk/cd8e10/airtravelbookingontology.owl
	http://mkrmke.org/knowledge/sumo/sumo.owl
	http://zaltys.net/ontology/AKTiveSAOntology.owl
	http://e-response.org/ontology/2006/20060815/e-response_buildings.owl
	http://swat.cse.lehigh.edu/resources/data/citeseer/cspost_27.owl
country	http://www.w3.org/Consortium/Offices/Presentations/RDFTutorial/rdfs/Countries.owl
	http://www.daml.org/services/owl-s/1.1/Country.owl
	http://www.loc.gov/standards/mads/rdf/mads-ontology-20101119.owl
	http://www.co-ode.org/ontologies/pizza/pizza.owl
	http://aims.fao.org/aos/geopolitical.owl
	http://dev.iptc.org/files/rNews/rnews_1.0_draft3_rdfxml.owl
	http://www.mindswap.org/2003/owl/geo/geoFeatures20040307.owl
	http://users.ecs.soton.ac.uk/cd8e10/airtravelbookingontology.owl
	http://eresearch.griffith.edu.au/ANDS/vitro/ANDS-VITRO.owl
	http://vivoweb.org/files/vivo-core-public-1.2.owl

Source: authors' own presentation

1.2.4. Module of Ontological Entities Disambiguation

The previous module associates a collection of ontologies to each of the attributes extracted from the WSDL descriptor. The purpose of the *ontological entities disambiguation* is to identify which ontological entities (out of the set of *classes*, *properties* and *instances* each collection of ontologies comprises) may be used for annotating the functional service attributes. This disambiguation procedure starts by identifying the ontological entities matching the service attribute to be annotated (matching entities), along with their *parent* and *children* entities—according to the concept hierarchy in each ontology. Later, in order to estimate the semantic relatedness between the service attribute and the ontological entities, this module proceeds to disambiguate the matching entities, establishing their senses regarding the ontology they belong to.

An ontology reasoner supports the syntactic matching procedure (between service attributes and entities) allowing the search and extraction of ontological entities. There several tools that enable reasoning on top of ontologies. However, considering the use of OWL ontologies in the context of our proposal, tools like *FACT++*, *HermiT*, *Pellet* or *RacerPro* are suitable inference engines for serving this purpose (Dentler et al., 2011).

Particularly, we use the *Pellet* reasoner for performing the syntactic matching procedure, since (*i*) it natively supports OWL, including a subset of OWL Full, (*ii*) offers a wide support for XML Schema types, and (*iii*) Pellet is an open-source tool in a continuous development/improvement process (Parsia et al., 2005). Additionally, Pellet is the only framework offering a native interface for *Jena*, which eases the handling of OWL and RDF (Sirin et al., 2008) ontologies.

Next subsections detail the artifacts composing the module of ontological entities disambiguation:

1.2.4.1. Ontology Reasoner: allows identifying the ontological entities whose label (name) matches the attribute to annotate (*$token_i$*). It also gathers the *parent* and *children* entities for each one of the matching entities, which later becomes their linguistic context used for disambiguating them. Let's assume for instance, the *token* "*weather*", and the ontology located at http://www.scs.ryerson.ca/~bgajdero/msc_thesis/code/ontologies/weather-ont-t2.owl. The ontology reasoner identifies a high semantic relatedness between the entity *WeatherReport* (matching entity) and the "*weather*" token. Likewise, the reasoner extracts the *parent* and *children* entities from *WeatherReport*: *DatedWeatherEvent* and *TimedWeatherEvent*.

1.2.4.2. Ontological entities disambiguation: this sub-module disambiguates the matching entity associated to *token*$_{i,}$ according to the set of entities comprising its linguistic context, by using a WSD technique similar to the one introduced in the *attribute disambiguation module* (section 1.2.2).

This disambiguation procedure is performed by running an open-source tool called *WordNet::SenseRelate::WordToSet (SR-WTS*) (Michelizzi & Pedersen, 2008), which receives as an ontological entity as input (*target entity*) e_{t1} and a set of *n* entities $\{e_1, e_2, \ldots, e_n\}$ composing its linguistic context. The disambiguation procedure generates a pair *entity-sense* $\{e_i,s_i\}$ for each ontological entities being disambiguated. At the end of this procedure, a set of pairs $\{(e_{1,} s_1), (e_2,s_2), \ldots, (e_k,s_k)\}$ is generated for each ontology.

In the example being discussed in the above section, the target entity (e_{t1}) is *WeatherReport* while its context entities are *DatedWeatherEvent and TimedWeatherEvent*. After running the disambiguation procedure, this sub-module obtains the following *entity-sense* pair: (*WeatherReport, "Summary of the weather conditions. It often includes the forecast conditions for a specific area"*).

1.2.5. Module of Selection of Ontological Entities

This module receives a pair *entity-sense* $\{e_i, s_i\}$—obtained from the above module—and a pair *token-sense* $\{t_{ti},s_{ti}\}$—delivered by the *attributes disambiguation module.* It is in charge of identifying the most semantically similar ontological entity to each *token*$_i$ (disambiguated service attribute). In order to do so, this module estimates the similarity measure proposed by Wu & Palmer (WUP) (Wu & Palmer, 1994), between the input pairs: $\{t_{ti},s_{ti}\} \text{ x } \{e_i, s_i\} \rightarrow \mathbb{R}$, which quantifies their semantic relatedness (*R*). This measurement is made on each ontological entity matching the *token*$_i$, while adding the *R*-value computed for entities belonging to the same ontology. This way, the module estimates the overall value of *R* for each matched ontology. The ontologies with the higher *R*-value are then submitted to a quality analysis for deciding which is the most suitable ontology for annotating the service attribute.

The module of *selection of ontological entities* comprises two components:

1.2.5.1. Filter of ontological entities: computes the semantic relatedness (*R*) between each of the service attributes (*token*$_i$) and their associated ontological entities. This sub-module dismisses those ontologies with lower overall value of *R*.

1.2.5.2. Quality Analyzer: this sub-module performs an analysis for estimating the quality of the ontologies that the above filter delivers. Such analysis is supported on the *Pellet reasoner* (Sirin et al., 2008), which allows identifying misapplied OWL constructs and misspelled terms and typos within the ontologies. Additionally, Pellet allows building the taxonomic structure that arranges the domain concepts and terms, in a comprehensive and consistent way.

When detecting inconsistencies, Pellet locates the involved ontological entities at the bottom of the concept tree turning them into subclasses of owl:Nothing. Finally, Pellet calculates the number of well-formed concept in every single ontology, so that those ontologies with the lower number of concepts are discarded.

1.2.6. SAWSDL Generator Module

This module implements a procedure that generates a SAWSDL document (W3C, 2007), specifying the association between each attribute of the WSDL descriptor and ontological entities, by using the modelReference schema attribute.

We use SAWSDL as the language for encoding the semantic annotations, since it has been adopted as a W3C recommendation, defining a mechanism for attaching semantic information to WSDL elements. In the example below, the annotation attached to the attribute *MyGlobalWeatherService* (service label) is presented:

```
<xsd:service
    name=”MyGlobalWeatherService” sawsdl:modelReferen
    ce= “http://www.scs.ryerson.ca/~bgajdero/msc_thesis/code/ontolo-
    gies/weather-ont-t2.owl#WeatherReport”>
</xsd:service>
```

Where the label <xsd:service> denotes the WSDL element, and sawsdl: modelReference specifyies the ontological entity that annotates the service attribute.

2. Experimentation

In order to measure the relevance of the annotations generated by the AA-CS platform, we adopted the *benchmarking* methodology (Blakeman, 2002), using as *gold standard* a collection of manually annotated Web and Telco services descriptors (WSDL documents). This way, we contrasted the annotations from the gold standard against those generated by the AA-CS platform. The coming sections describe the service test collection, the performance measures we used and the results obtained from the experimental evaluation of our approach.

2.1. Service Test Collection

The gold standard used as baseline for contrasting the annotations that our platform generates consists of 90 WSDLs belonging to Web services, and 4 WSDLs describing telecommunications services, i.e. 94 service descriptors. These descriptors were classified into 13 domains, 9 of them related to Web services: Communications (10 WSDLs), Economy (10), Education (10), Food (10), Geography (10), Health care (10), Simulation (10), Traveling (10), and Armament (10); and 4 domains regarding telecommunication services: MMS (1), Payment (1), SMS (1), Terminal location (1).

2.2. Performance Measures

For estimating the quality of the annotations associated to the service descriptors processed by the AA-CS platform, we adopted statistical measures widely used in characterizing the performance of information retrieval systems (e.g. search engines): *Precision (p)* (Ec. 1)*, recall (r)* (Ec. 2)*,* and *F-measure (f)* (Ec. 3) (Yatskevich, 2003). When contrasting the automatic annotations against those provided in the test collection three sets of annotations were identified:

- *True positives* (*VP*) or correctly assigned annotations.
- *False positives* (*FP*) or misassigned annotations.
- *False negative* (*FN*) or semantic annotations that the AA-CS platform misses, despite their relevance.

Based on the cardinality of these sets the expressions below define the abovementioned quality measures:

$$p = \frac{|TP|}{|TP| + |FP|} \ (1) \quad r = \frac{|TP|}{|TP| + |FN|} \ (2) \quad f = (2 * r * p) / (r + p) \quad (3)$$

The precision measure estimates the reliability of the annotations made by the AA-CS platform, while *recall* specifies the ratio of correctly assigned annotations to the overall number of relevant annotations and *F-measure* determines the overall quality of the annotation (Yatskevich, 2003).

The above measures were estimated on each of the 94 descriptors comprising the test collection. For computing the general precision and recall of the whole annotation process, the macro-average (Lewis, 1992) method was used as follows:

$$P = \frac{\sum_{i=1}^{n} p_i}{n} \ (4) \quad R = \frac{\sum_{i=1}^{n} r_i}{n} \ (5)$$

Where ***n*** is the overall number of matches made.

2.3. Evaluation Results

The experimental evaluation ran on a server with a 3.20 GHz dual-core processor, 8192 MB in RAM and Debian Squeeze as operating system.

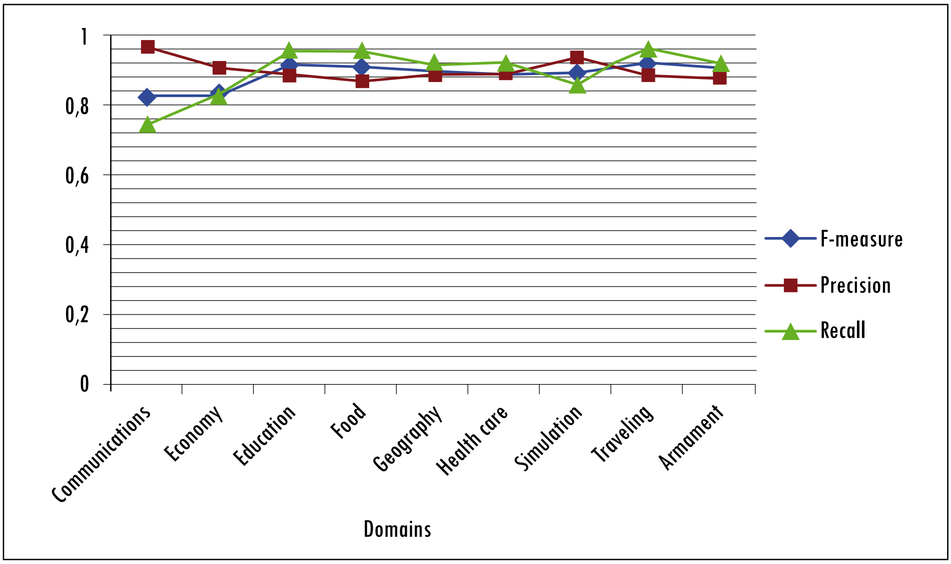
The analysis of the obtained results began by estimating the performance measures for each one of the 13 service domains defined in section 3.1. The estimation of these measures involves contrasting the annotations made by AA-CS against those annotation provided in the service test collection. Figure 4a illustrates the results for services descriptors belonging to Web services domains, while Figure 4g presents the performance measurements for Telco service domains.

The results depicted in Figure 4a, evidence a high *precision* value: 88% average over all domains. The traveling domain—comprising WSDL descriptors with fewer attributes (ranging between 2 and 14) in contrast to other domains—presents the best performance of the Web service domains: its corresponding F-measure is over 92%, which indicates that for this particular domain, the AA-CS platform properly assigns semantic annotations to the attributes of the services comprising it.

On the other hand, the less favorable performance of the platform corresponds to the communication, geography and health care domains. Services belonging to those domains feature a large amount of attributes (ranging from 57 to 66 attributes). Thus, it is possible to state that the performance of the implemented annotation procedure is reduced as the number of service attributes grows.

The results outlined in Figure 4b, regarding service belonging to Telco domains also evidence a high *precision*, being 94% average. The MMS domain features the best performance measures, which may be due to the low amount of attributes (6) extracted from the only service this domain contains, when compared with the quantity of attributes characterizing the rest of the services belonging to the other three domains (which range from 13 to 18 attributes). In this particular case, the F-measure reaches 88%, implying that the AA-CS platform properly associates semantic annotations to the attributes extracted from services belonging to the MMS domain.

Figure 4a. Precision, Recall and F-measure (Web Services)



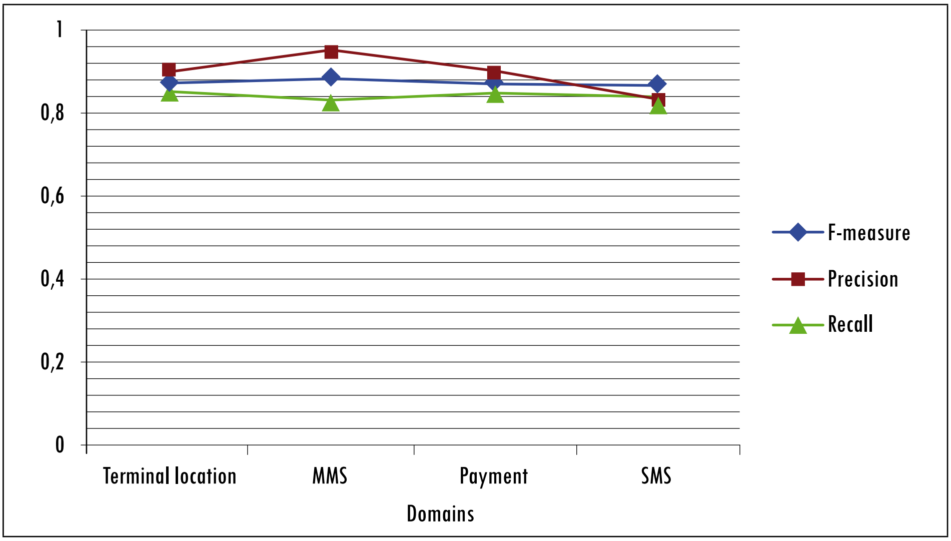
Source: authors' own presentation

Furthermore, the SMS service domain presents the smallest value of *precision* (83%), while holding the highest number of attributes (18 in total) in contrast to the remaining domains. This confirms the abovementioned rationale stating that the quality of the annotation made by the AA-CS platform is inversely related to the amount of service attributes being annotated.

In general, the system's performance is substantially favorable for service descriptors with attributes ranging from 2 to 34, featuring an average F-measure between 87% and 92%. Processing descriptors with 35 to 66 attributes causes a slight reduction in the quality of the annotations associated by the platform exhibiting an average F-measure between 82% and 87%.

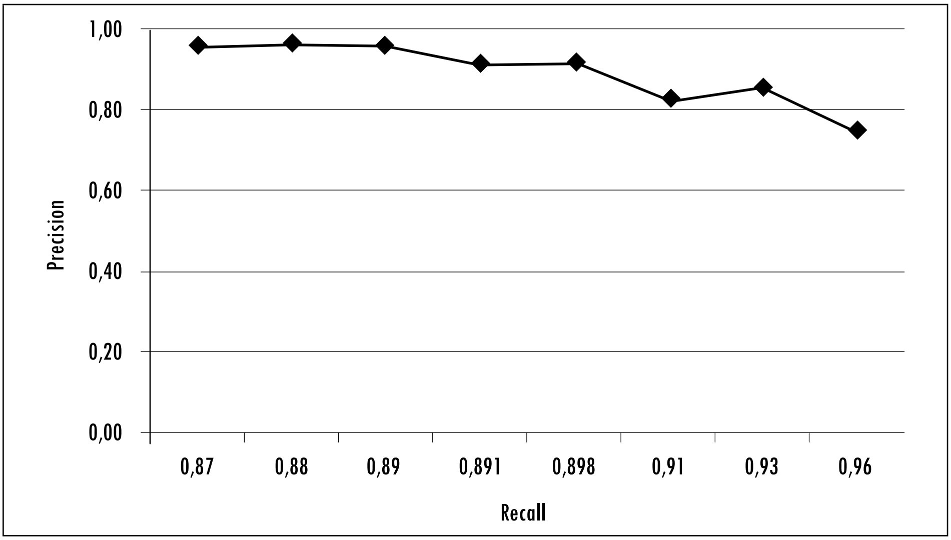
The slight drop in the overall platform performance when annotating WSDL descriptors with a large number of attributes is due to the rising error likelihood of the disambiguation mechanism when processing a wider context window. In such situation the context tends to include noise terms hindering the proper assignation of senses to the service attributes. This limitation may be addressed by setting a restriction on the number of attributes being disambiguated, i.e. when receiving a service descriptor featuring more than 34 attributes, the processing should be splitted and parallelized into various context windows.

Figure 4b. Precision, Recall and F-measure (Telco Services)



Source: authors' own presentation

Figure 5. Precision *vs*. Recall



Source: authors' own presentation

Figure 5 shows the relationship between the measures of *precision* and *recall*. This graph evidences the capability of the AA-CS platform for attaching highly relevant semantic annotations to each service attribute in the test collection, since all data points lie in an area of high *precision* (up to 95%) and *recall* (over 87%).

Conclusion

Existing approaches for supporting the semantic description of web service and resources require for the developers to mediate in the whole annotation process, which generally implies high resource consumption and a considerable effort from them. Most of those approaches even though associate ontological entities to service descriptors in a semi-automatic way, perform such annotation neglecting the meaning of the service attributes regarding the linguistic context they are in (i.e. the content of the service descriptor). Thus, the quality of the annotations in these kind of approaches could be low to the extent that they might not match the domain to which the services belong.

Unlike related approaches, the platform we introduced in this paper, is supported on WSD techniques for disambiguating both service attributes and ontological entities, allowing the semantic annotations to be consistent with the service domain. One major contribution of this work is the automated search of online available ontologies, promoting this way the reuse of a wide range of existing vocabularies specified under an agreed terminology by domain experts.

The results derived from the experimental evaluation performed on the AA-CS platform—in terms of *precision*, *recall* and *F-measure*—demonstrate that both the mechanisms implemented for disambiguating the functional attributes of Web and Telco services, and the technique for associating ontological entities to those attributes, are closely related to the judgment of humans when performing the same annotation process by hand. This evidences the benefits of using WSD techniques for articulating an automated mechanism capable of assigning highly related ontological entities to services attributes.

Further work in our research aims to adapting the proposed mechanism for assigning semantic annotations to web resources other than service descriptors, such as html documents and multimedia files.

Acknowledgement

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

AKSOY, C. et al. *MATAWS: A multimodal approach for automatic WS semantic annotation.* In: *International Conference on Networked Digital Technologies.* Springer, 2011. pp. 319-333.

ASSWAD, M.; DE CESARE, S. de and LYCETT, M. A query-based approach for semi-automatic annotation of web services. *International Journal of Information Systems and Social Change (IJISSC).* 2011, vol. 2, no. 2, pp. 37-54.

BLAKEMAN, J. *Benchmarking: definitions and overview* [document on line]. 2002. <http://www4.uwm.edu/cuts/bench/bm-desc.htm>. [Cited: 01-11-2013].

BOUCHIHA, D. and MALKI, M. *Semantic annotation of web services.* n. d: Sidi Bel Abbes, 2012.

CARDOSO, J.; MILLER, A. and EMANI, S. Web services discovery utilizing semantically annotated WSDL. *Resoning Web.* Venecia: Springer, 2008. pp. 240-268.

CHABEB, Y.; TATA, S. and BELAÏD, D. Toward an integrated ontology for Web services. *Fourth International Conference on Internet and Web Applications and Services, 2009. ICIW '09.* 2009, pp. 462-467.

CHRISTENSEN, E. et al. *Web service description languaje* [document on line]. 2001. <http://www.w3.org/TR/wsdl> [Cited: 12-27-2012].

DENTLER, K. et al. *Comparison of reasoners for large ontologies in the OWL 2 EL profile.* s. d.: IOS Press, 2011.

FALLERI, J. et al. *Automatic tag identification in web service descriptions* [document on line]. 2010. < http://www-sop.inria.fr/members/Zeina.Azmeh/webist10.pdf>.

FARRELL, J. and LAUSEN, H. *Semantic annotations for WSDL and XML schema* [document on line]. 2007. <http://www.w3.org/TR/sawsdl/> [Cited: 11-12-2012].

GOOGLE. 2012. *Custom search* [document on line]. 2012. <https://developers.google.com/custom-search/docs/start?hl=en#overview> [Cited: 11-02-2012].

JIE, G. et al. *Applying recommender system based mashup to web-telecom hybrid service creation*. Piscataway, NJ: IEEE Press, 2009, pp. 3321-3325.

KHALID, E.; AHMED, H. and PATRICK, M. *Clustering WSDL documents to bootstrap the discovery of web services*. Washington: IEEE Computer Society, 2010, pp. 147-154.

LEWANDOWSKI, D. The retrieval effectiveness of search engines on navigational queries. *Aslib Journal of Information Managemet.* 2010, vol. 63, no. 4, pp. 354-363.

LEWIS, D. *Representation and learning in information retrieval.* Boston: University of Massachusetts, 1992.

LY, P.; PEDRINACI, C. and DOMINGUE, J. *Automated information extraction from web APIs documentation.* Berlin: Springer-Verlag, 2012.

MICHELIZZI, J. and PEDERSEN, T. *WordNet-SenseRelate-WordToSet* [document on line]. 2008. <http://search.cpan.org/dist/WordNet-SenseRelate-WordToSet/doc/README.pod> [Cited: 12-13-2012].

MILLER, G. WordNet: a lexical database for English. *Communications of the ACM.* 1995, vol. 38, no. 11, pp. 39-41.

OSMAN, T.; THAKKER, D. and AL-DABASS, D. S-CBR: semantic case based reasoner for web services discovery and matchmaking. *IEEE International Conference on Web Services (ICWS 2006).* 2006. Chicago, 18-22 september. *IEEE Computer Society*. 2006, pp. 29-36.

PARSIA, B.; SIRIN, E. and KALYANPUR, A. *Debugging OWL ontologies.* s. l.: ACM, 2005.

PEDERSEN, T. and KOLHATKAR, V. WordNet: Sense relate: allwords - a broad coverage word sense tagger that maximizes semantic relatedness. *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics.* Colorado, May 31-June 5, 2009: The Association for Computational Linguistics, 2009, pp. 17-20.

PEDERSEN, T.; BANERJEE, S. and PATWARDHAN, S. Maximizing semantic relatedness to perform word sense disambiguation. *Minnesota Supercomputing Institute Users Bulletin.* 2005, vol. 2, no. 1.

PEDERSEN, T.; PATWARDHAN, S. and MICHELIZZI, J. *WordNet: Similarity - Measuring the Relatedness of Concepts.* s. l.: AAAI Press/The MIT Press, 2004.

SIRIN, E. et al. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web.* 2008, vol. 5, no. 2, pp. 51-53.

SMITH, K. *OneAPI: A standard to simplify cross-network development.* n. d., 2009.

VERMA, K. et al. METEOR-S WSDI: A Scalable P2P infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management.* 2005, vol. 6, no. 1, pp. 17-39.

W3C. Semantic annotations for WSDL working group. *SAWSDL Recommendation.* [document on line]. 2007a. <http://www.w3.org/2002/ws/sawsdl/> [Cited: 02-05-2013].

W3C. *Web services description language (WSDL) Version 2.0 Part 0: Primer* [document on line]. 2007b. <http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/> [Cited: 01-11-2013].

WU, Z. and PALMER, M. Verb semantics and lexical selection. *32nd. Annual Meeting of the Association for Computational Linguistics.* Las Cruces, New Mexico, 27 de Junio- 30 de Junio 1994 : Association for Computational Linguistics, 1994, pp. 133-138.

YATSKEVICH, M. *Preliminary evaluation of schema matching systems.* Trento: University of Trento, 2003.