Uncovering WSDL Specifications' Data Semantics

George A. Vouros, Alexandros Valarakos, Konstantinos Kotis

AI-Lab, University of the Aegean, Karlovassi, Samos, Greece
{georgev, alexv, kotis}@aegean.gr

Abstract. The work[1] reported in this article aims towards computing web services' data semantics. The paper provides extensive experimental results towards the automatic semantic annotation of WSDL specifications. Specifically, this paper reports on combinations of state-of-the-art methods for automatically mapping the part elements of the WSDL input and output messages to ontology classes: These combinations result to a specific method for Uncovering web Services Data Semantics (USDS). We study the performance of USDS even in challenging cases where lexical items are rather scarce and misleading. Experimental results are being thoroughly discussed, showing the potential and limitations of USDS.

Keywords: semantic annotation, mapping methods, WSDL specification, data semantics

1 Introduction

The automatic discovery of Web services is a highly important service manipulation task. WSDL is mainly focusing on operational and syntactic details regarding the implementation and execution of Web services. The lack of explicit semantics in WSDL specifications makes them insufficient to satisfy the requirements for flexible and effective Web service 'manipulation' tasks, as they force the relevant mechanisms to be based mostly on keyword matches. While we may relate different types of semantics to the web services, we can distinguish two widely recognized types of semantics: (a) Data semantics (introducing the semantic signature of services: semantics of input/output messages of service operations), (b) Functional Semantics (function of operations and of the service itself). We may further add to this list: Protocol semantics, execution semantics, non-functional semantics (security, QoS), and others. This paper focuses on data semantics, which is generally accepted to be one of the most critical aspects regarding services' semantic description. Although other important aspects exist with respect to web services' signature (e.g. goals, pre/post conditions [10], services' classification [2]), mainly due to their strong dependence on data semantics, they have not received the attention that data semantics do.

[1] This work has been supported by the European Commission project Grid4All under grant number FP6, project IST-Grid4All-2006-034567

The main objective of this paper is to study the automatic semantic annotation of WSDL specifications, given ontologies related to the services' domains. Towards this objective, we devise a method for mapping input/output messages' part names to ontology classes and study its performance to variations of the OWL-TC corpus 2 [6]. This method, aiming to uncovering the data semantics of web services (USDS), combines state-of-the-art string similarity methods and vector-based methods, aiming to mitigate difficulties and limitations of other approaches, even in challenging cases. The aim is for services to be automatically translated to semantically enriched specifications, and thus be registered to semantic registries automatically.

As pointed in [11], the process of semantic annotation of services requires input from multiple sources, including the source code of the service, the API documentation and description, as well as "external" information sources such as users' license agreements and sources of background knowledge [2]. The main assumption behind this article is that these information sources, in conjunction to service specification elements, can provide valuable information for the automatic annotation of web services. It is within the goals of this work to reduce human intervention to the subsidiary tasks that humans can do better: introduce descriptive documentation of WSDL specifications and validate results.

This paper is structured as follows: Section 2 states the problem that this article deals with. Section 3 presents related work and motivates the proposed method. Section 4 presents in detail the mapping techniques used and the overall proposed method. Section 5 presents the experimental setup and results, and section 6 concludes the paper, sketching future work.

2 Problem Statement

The problem that this paper deals with is as follows: "*Given (a) a WSDL specification and (b) an ontology O= (S,A), S being the set of terms lexicalizing ontology elements and A the ontology axioms, provide mappings of WSDL messages' part names to ontology classes with respect to the intended semantics of the service*".

We deal only with services' signatures specified in WSDL (i.e. with data semantics). More specifically we consider the following specifications:

- The signature of a service *s* specifies a set of input/output messages. These are denoted <*service_id, message_type, message_id*>, where *message_type* can be *input* or *output*.
- Each message <*service_id, message_type, message_id*> has one or more parameters: <*service_id, message_id, name, data_type*>, where *name* is the name of the parameter, and *data_type* specifies the (atomic or complex) data type of the parameter.
- Each parameter is associated with textual annotations: <*service_id, message_id, name, annotation_type, text*>, where the type of annotation is *description* or *comment*, and *text* is the actual annotation text. Each parameter may be associated with more than one annotation.

Computed mappings between WSDL messages' part names and ontology elements are of the form <*service_id, message_id,name,element,rel*>where *name* is the name of the parameter, *element* is a term in the ontology signature S, and *rel* is the assessed

relation between *element* and *name*: This may be *equivalence*, *subsumes* or *subsumed*. For the purposes of this paper we restrict our attention to the equivalence case.

We have to notice that the above problem statement follows the WSDL 1.1 specification (also addressing the great number of existent services): However, this can be easily restated for WSDL 2.0 specifications considering that no message part names exist.

3 Related Work and Motivation

3.1 Related Work

The work reported in this article, aims to provide services with data semantics by exploiting of domain ontologies, services' WSDL specifications and textual annotations. Close to the aims of this work are efforts that exploit textual descriptions of services for the annotation, classification, and for the assessment of similarities between web-services.

The METEOR-S Annotation Framework [1] is one of the most prominent approaches, aiming at semi-automatically marking up web service descriptions with ontologies, by means of a schema matching algorithm: Ontologies and XML schemata used by WSDL specifications are converted to SchemaGraphs that are compared by computing linguistic similarities between elements and structural similarities between the schemata. For the purposes of linguistic similarity, the proposed algorithm consults WordNet to find synonyms. In this paper we emphasize on the exploitation of textual information for uncovering the semantics of WSDL parts. In contrast to our approach which aims, among others, to compute latent features that explicate the semantics of WSDL parts, METEOR-S exploits "shallow" features concerning XML schema and ontology elements' names: This for instance affects methods' efficacy in cases where polysemous terms appear, or in cases where specifications are at different granularity levels, as far as the conceptualization of the domain is concerned.

The work reported in [2] aims at assessing WSDL specifications' similarity by exploiting the structure of data types and operations of services, as well as the semantics of natural language descriptions and identifiers. Although the aim of this work is to support query-by-example discovery of services, the emphasis on semantic matching given textual descriptions of services, and the exploitation of identifiers' semantics, brings this work close to our work. This approach uses a vector model, exploiting the textual descriptions of services, and consults WordNet to calculate the semantic distances between identifiers of WSDL elements. While semantic matching is restricted to the exploitation of identifiers, identifiers' senses are not disambiguated, making the calculation of semantic distances rather problematic. Disambiguation is a vital task in our research since we aim to match WSDL elements to specific ontology concepts, explicating their semantics.

Aiming to show how content-based approaches can contribute to semantic matching of OWL-S service specifications, OWLS-MX [3], aims to exploit the implicit semantics of any part of OWL-S service description by representing it as a

weighted category-index term vector. Index terms are stemmed lexical items from a shared minimal vocabulary. This vocabulary results from the canonical unfolding in an underlying ontology of the words or concepts that exist in text categories that correspond to OWL-S elements (hasInput, ServiceName, TextDescription etc). Although the aims of this work are quite different from our aims, it provides firm evidence towards our conjecture: That the use of textual descriptions of service parts in conjunction to their specifications can help to uncovering their implicit semantics. However, as shown in [4], there are pitfalls to the logic-based and syntactic matchmaking methods (due to granularity of ontology specifications, surjective mapping of concepts, and incomplete coverage of service semantics) that require explicating the semantics of services' input/output parameters, before these are being compared by logic-based matchmakers.

Further evidence is provided by experiments with the ASSAM's annotation wizard [9]. ASSAM casts the problem of classifying operations and datatypes in a Web Service as a text classification problem. The tool learns from Web Services with existing semantic annotations. Given this training data, a machine learning algorithm can generalize and predict semantic labels for previously unseen Web Services. The approach described in the current paper does not require the pre-existence of semantic annotations to decide the mapping of WSDL elements to ontology concepts.

3.2 Motivation and overview of the method

Our work is being motivated by the view that, for web-service matchmakers to perform accurately, they need the precise, intended meaning of web services' signature in a fine-grained way: This means that parts of web service messages must be mapped to specific terms that are being axiomatized in a formal ontology.

To do so, we need to mitigate pitfalls related to phenomena concerning synonym terms, homonym terms (i.e. polysemy), typographic variations of terms, differences between the granularity of services and ontological specifications.

In conjunction to these pitfalls, we need to take also into account the "nature" of WSDL specifications, which are being produced from program code, with few and, in most of the cases, misleading comments, descriptions, and "tricky" names of the parameters being involved, with improper or faulty use of domain terminology.

To mitigate these pitfalls and avoid difficulties that are inherent to the specifications of web services, we employ the combination of different methods towards a system for uncovering the data semantics of web services (USDS): The core configuration of this system comprises two state of the art methods: COCLU [8], and LSA-based-mapping [13]. Specifically, COCLU is expected to tolerate typographic variations of terms, assessing similarities between terms whose appearance is quite similar, even if one of them is an abbreviation or a concatenation of the other term parts: These are variations that edit-distance measures are hard to capture.

The LSA-based-mapping aims to mitigate problems concerning synonym and homonym terms, as its aim is to disambiguate the meaning of web service parameters, mapping them to WordNet senses that best capture their intended meaning, according to their own lexicalization, their associated types, as well as according to descriptions

and comments given. The same method maps ontology classes to WordNet senses that are assessed to capture the human-intended meaning of the formal specifications: Having done these mappings, and in case the intended meanings (WordNet senses) of an ontology class and of a service parameter are related, then these can be mapped. More specifically, in case the corresponding WordNet senses coincide, or they are related via a synonym relation, then the method assesses an exact match. In case their corresponding terms are being related via a hyponym relation, then given that there is a short distance between them (in terms of the number of hyponym/hyperonym relations between them) then the method may assume that there is a subsumption relation between the parameter term and the ontology class. Doing so, the method facilitates tackling problems related to having specifications (WSDL and ontology) at different granularity levels, and relating elements with a "semantic distance". However, in this paper we only consider cases of exact matches.

To evaluate USDS, we provide extensive experimental results with different configurations of string-matching based and vector-model based methods in different sets of WSDL specifications.

4 Semantic Annotation of WSDL specifications

4.1 Annotating WSDL

As pointed out, we consider that the overall semantic annotation of WSDL specifications comprises three distinct stages: The annotation stage, the mappings stage, and the validation stage. However, for the purposes of this paper we require human intervention only in the annotation stage.

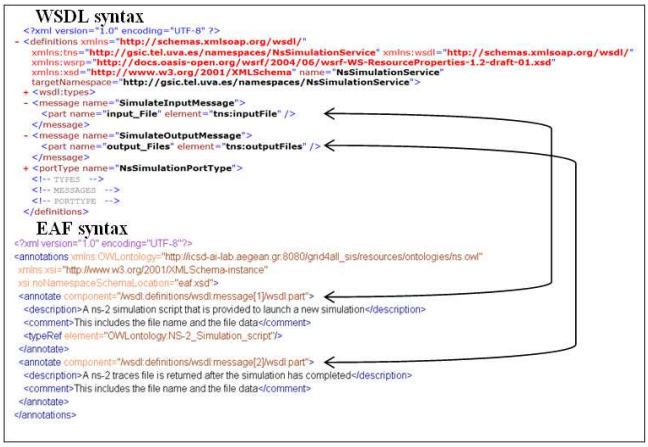

Fig. 1 Annotating WSDL specifications

During the annotation stage, humans provide textual descriptions for elements of the WSDL specification. The annotation stage takes as input the WSDL specification and produces an external xml annotation file (EAF) based on a specific annotation

schema that we have specified for this purpose. This is an annotation-template file that provides "slots" for the description of WSDL elements: For the service itself, for each interface, operation and input/output messages' elements it provides elements for "comments", "description", as well as support for mapping mechanisms between WSDL elements and ontologies. As Fig.1 shows, the EAF is aligned with the WSDL specification via XPATH expressions. The annotation schema can be extended for supporting other types of textual information that are necessary for the annotation of WSDL elements.

Although we plan to incorporate SAWSDL [5] into our framework, we do not commit to the use of SAWSDL at this stage, emphasizing mostly on the use of textual descriptions/comments for WSDL elements.

4.2 The USDS Semantic Annotation Method

The USDS semantic annotation system takes as input a WSDL document together with the corresponding xml annotation file, as well as a domain ontology. As already specified in section 2, the output of this system is a set of assessments concerning exact mappings between messages' part names and ontology classes. USDS, in addition to the part names of the input/output messages, exploits services' types' specifications, as well as textual descriptions and comments in the annotation file.

As already pointed out, the core of USDS comprises the combination of two state of the art methods: COCLU, and LSA-based-mapping. COCLU is a compression-based clustering algorithm (COmpression-based CLUstering). The algorithm is based on the assumption that different lexicalizations of a term (typographic variants) use a common set of `core' characters. Therefore, typographic variants that `mostly' use this set are potential alternative lexicalizations of the same concept, while those ones that are `far' from this set are potentially related to the lexicalization of a different concept. Further details for this model and partition-based clustering algorithm are provided in [8].COCLU has been used for the comparison of (a) the description of each WSDL input/output message part, with the labels of ontology classes, (b) the elements in the atomic/complex types specifications of messages' part elements with the name and labels of ontology classes, and (c) the comment of each WSDL input/output massage part with the comments of ontology classes.

The LSA-based-mapping [13] aims to disambiguate the meaning of WSDL messages' part names, mapping them to WordNet senses that best capture their intended meaning, according to their lexicalization, their associated types, as well as according to descriptions and comments given. As already said, the same method maps ontology classes to WordNet senses that capture the human-intended meaning of the formal specifications: Having done these mappings, and in case the intended meanings (WordNet senses) of a class and of a service parameter coincide, then these are mapped. The Latent Semantic Indexing (LSI) method assumes that there is an underlying latent semantic space that it estimates by means of statistical techniques using an association matrix (n×m) of terms-documents: Documents in our case correspond to WordNet senses. Terms are selected from the vicinity of WordNet senses (i.e. from the senses themselves and from their hyponym/hypernyms). Latent Semantic Analysis (LSA) computes the arrangement of a k-dimensional semantic

space to reflect the major associative patterns in the data. This is done by deriving a set of k uncorrelated indexing factors, which may be considered as “latent concepts”. Then, each term and document is represented by its vector of factor values, indicating its strength of association with each of these “latent concepts”. By virtue of dimension reduction from the N terms space to the k factors space, where k<N, terms that did not actually appear in a document may still end up close to the document, if this is consistent with the major patterns of association in the data.

When one searches an LSI-indexed database of documents, it provides a query (i.e. a pseudo-document), which is a list of terms. The similarity between two documents is computed by means of the dot product between the corresponding representation vectors. Doing so, LSI returns a set of graded documents, according to their similarity to the query.

In our case the semantic space is constructed by terms in the vicinity of the senses S1, S2,…Sm of the WordNet entry matching an ontology class name C, or a WSDL message part name C. The set of terms in the semantic space include:

- The term C’ that corresponds to C. C’ is a lexical entry in WordNet that is a linguistic variation of C.
- Terms that appear in C’ WordNet senses S1, S2,…Sm.
- Terms that constitute hyperonyms / hyponyms of each C’ sense.
- Terms that appear in hyper(hyp)onyms of C’ senses.

As far as ontology classes is being concerned, each query is being constructed by extracting terms from the label, and the comment of an ontology class, as well as from the names of its properties’. Concerning the WSDL specifications, for each input/output message part element we extract terms from its name attribute, from the names of the elements defined in the complex types of the type attribute, as well from its related annotation elements (description and comments). Specifically, terms may result from the tokenization of phrases and may be either simple terms, or compound terms.

To combine the above mentioned methods we have used the AUTOMS-F framework [7]: Specifically, in our implementation the combination of methods produces the union of the mappings produced by each of them. This improves the recall of the final method, but it may result to less precise results. Although more sophisticated types of methods’ combinations have been tested, these have given less encouraging results: Future work concerns the thorough investigation of these techniques.

Two additional string-matching based methods have been also employed. These methods compare the names of ontology classes and the part names of the service messages. Comparisons take place for every potential mapping pair. The better matches (those with the largest similarity values) are selected for every WSDL part. The Exact String Matching (ExactString) identifies a match if the compared names are exactly the same. This allows us to show the “difficulty” of our cases, given that such a simple method fails to exhibit effective performance.The Levenshtein matching method incorporates a distance measure that specifies the minimum number of operations needed to transform one string into another. The valid operations are: insertion, deletion, or substitution of a single character. Our implementation considers a match between two strings if and only if at most two operations are required for their transformation.

In addition to the above, we have also used a Vector Space Model – based (VSM) method, which computes the matching of documents pairs. Each document is represented by a vector of n weighted index terms. Index terms correspond to the simple terms that are extracted from all documents. Here we construct (pseudo-) documents that correspond to ontology classes and WSDL messages' part elements: As it is done in the LSA-based method, these pseudo-documents include terms from the label, and the comment of an ontology class, as well as from the names of its properties'. Concerning the WSDL specifications, for each input/output message part element we extract terms from its name attribute, from the names of the elements defined in the complex types of the type attribute, as well from its related annotation elements (description and comments). The VSM-based method builds the vector of a pseudo-document by assigning to the weight of a term the frequency of its appearance in the document. The similarity between two vectors (each corresponding to a WSDL message part name and to an ontology class) is computed by means of the cosine similarity measure. This computation ranges in [0, 1]: A threshold (currently set to 0.35) is set for deciding when a match occurs.

5 Experiments and Discussion

5.1 Experimental Setting

To evaluate our approach, we have used the OWL-S Service Retrieval Test Collection (OWLS-TC) version 2 [6]. From the OWLS-TC collection, we have translated to WSDL a subset of 87 services, due to problems we faced with OWL-S-to-WSDL translation, and due to duplicate WSDL part elements. In total, we have been experimenting with 5 different domains and 6 different domain ontologies, which result to several different sets of experiments, with an initial given varied degree of difficulty. Additionally, we have produced additional experiments by creating variations of the corpus in order to test the robustness of the USDS different configurations.

Table 1. Information about the experimental domains, ontologies and services.

Domain	Domain Ontology (.owl)			# Services (WSD)	Total # of part elements.	Distinct part elements to be annotated
	Ontology	#concepts	#properties			
(1): Travel	Travel	34	6	9	22 (4)	10
	Portal	171	104	9	27 (4)	17
(2): Education	Books	60	11	8	19 (4)	19
	Portal	171	104	31	63 (3)	63
(3): Economy	Books	60	11	11	31 (4)	19
	Concept	17	3	11	31 (4)	12
(4): Weapon	SUMO	613	208	3	7 (3)	7
(5): Medical	Hospital Physician	64	45	5	39 (10)	39

Table 1 summarizes information concerning the initial sets of experiments and characteristics of the services and ontologies used. The first and the second column present the domain and the ontology used. The third and fourth columns provide information concerning the number of concepts and properties in each ontology. The fifth column provides the number of services (WSDL specifications) that exist in each

set, while the sixth one presents the total number of part elements that exist in a set, and the maximum number of part elements that exist in a WSDL specification. The last column specifies the total number of the distinct part elements that should be annotated in each set of experiments.

Concerning WSDL specifications for domains 1 to 4, message part names are mainly composed by a single-word term capitalized and an underscore character as a prefix (e.g. _COUNTRY). This variation slightly differentiates part names from ontology class names. For the domain 5, the messages part names are composed by multi-word terms, either separated with an underscore or with no separator, or using a combination of these (e.g. GetPatientMedicalRecords_AuthorizedMedicalRecords). Such terms are not included in the related domain ontologies, however, their substrings match to ontology class names (e.g. MedicalRecords). We handle individual, distinct terms of multi-word terms separately, only in cases these are separated by an underscore separator, which is one of the most generic case considered.

The characteristics of the domain ontologies are important for the experiments: These include the size of the ontologies, the annotation of classes/properties, i.e. labels and comment annotations, as well as the richness of specifications for each class/property, i.e. the number of subclasses or related properties, depth of hierarchy for each class, etc. Apart from SUMO, which is an upper, widely-accepted ontology, ontologies accompanying the OWL-TC corpus have been developed independently from OWL-TC. Our analysis for these ontologies shows that there are ontologies with some annotations for their classes (e.g. the Travel ontology provides only comments for the defined classes, and the Portal ontology provides comments or labels for some of the classes), and ontologies with no annotations at all (e.g. the HospitalPhysician ontology). As far as richness of specifications is concerned, there are ontologies whose elements are mildly interrelated, such as SUMO (max parents: 3, mean parents: 2, max siblings: 15, mean siblings: 7, all properties have a domain and range specified), but there are others not so rich, such as the HospitalPhysician (max parents: 1, mean parents: 1, max siblings: 8, mean siblings: 4) or the Books ontology (none property have a domain and range specified).

WSDL part names have been manually annotated by human annotators that have adequate knowledge of the related domains. They have been advised to carefully choose the annotations in order to indicate as close as possible the intended meaning of the input/output message parts annotated. Where possible, annotators have been advised to get feedback from xsd-schema complex types included in the WSDL specifications. More specifically, we can identify 3 different annotation cases that have been applied in the corpus: (a) Annotations that are formed by "*free text including terms from xsd-schema types and from the WSDL message part name element*". (b) Annotations that are formed by "*free text including terms only from the WSDL message part name element*". (c) Annotations that are formed by "*a single term*". Annotators choose a single term without considering xsd-schema types or WSDL message part name element information. For instance, for the "*wsdl:part name="Capital_City"*", the annotator creates the description annotation "*<description> Capital </description>*", which is the intended meaning (or synonym) of the entity "Capital City".

The result of this process is a set of annotations with terms belonging in one of the following three categories: a) terms from xsd-schema types, b) terms from WSDL message part names, or c) terms chosen by human annotators.

The use of "free text" in combination with xsd-schema types' terms and/or WSDL message part names' terms, means that the annotator is allowed to form a natural language sentence: E.g. "*<description> The service requests accommodation using country information </description>*". Although the use of free text may distract the matching methods, the freedom that the approach gives to the annotator is important and realistic. In this example of annotation, the annotator has combined terms (e.g. "country") from messages' part names (case b). As another example, in the comment "*<comment> The country name is a string. A country is described with its capital, its currency, and its government </comment>*" the annotator has included terms (terms "capital", "currency", "government") from the xsd-schema type that corresponds to the specific message part name that is annotated (case a). Table 2 summarizes information concerning annotations per domain. It must be noticed that all annotations contain 5 to 7 "significant" terms ie. non stop-words that may drive the computation of the intended mappings. In addition to the above, motivated by our experience with vector-model-based ontology alignment methods (perform better with ontologies that have rich information, i.e. with ontologies that contain labels and comments for every ontology class), we artificially enriched the annotation information of ontology classes with textual information. All experiments ("*enriched ontologies*" cases) have been run with these "enriched" ontologies.

Table 2. Information concerning annotations per domain

Domain Id	Annotation case	Terms that match a class name
1	Descriptions: case (b), Comments: case (a)	Approx. 50%
2	Descriptions: case (b), Comments: case (a)	NA
3	Descriptions: case (b), Comments: case (a)	Approx. 70%
4	Descriptions: case (b), Comments: case (a)	Approx. 40%
5	Descriptions: case (c), Comments: case (a)	Approx. 70%

Furthermore, we have modified the WSDL specifications by replacing the name of their input/output message part elements with a unique random string which ranges in length between 9 and 11 characters. These new specifications double the number of experiments that have been contacted. Such a setting aims to unveil the importance of the natural language descriptions of the input/output message parts for our approach.

5.2 Results and Discussion

Due to space limitations, we present the precision and recall for all experiments contacted with the annotated WSDL specifications and the enriched ontologies (Fig. 2): These provide the best results for all experiments' configurations. Figure 2 shows that (as it was expected) the higher recall is achieved by the combination of all methods. The recall of an individual method for a specific domain/ontology pair may not be high due to the characteristics of the specific domain/ontology pair (e.g. due to the compound terms in name values of the WSDL part elements). For instance, the recall for the Domain5/hospitalPhysician.owl pair is low even for combined methods

(e.g. LSA+COCLU). So even if the composition of the "core methods" of USDS (LSA+COCLU) achieves recall equal to 0.8 for all the domains (in average), it achieves 0.1 for this specific domain/ontology pair.

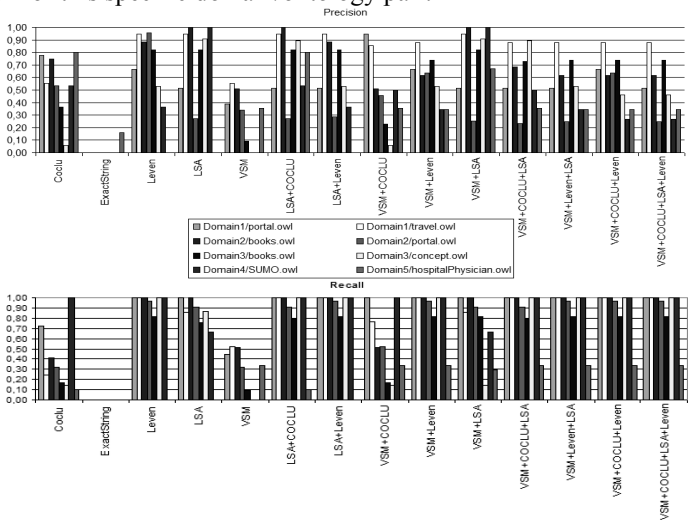

Fig. 2. Precision and recall for different USDS configurations using annotated WSDL specifications and enriched versions of ontologies.

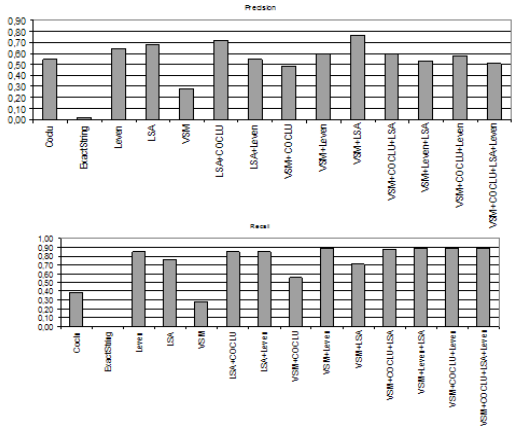

Fig. 3. Average precision and recall for each configuration

The higher precision is achieved by the composition of LSA and VSM. Specifically, the VSM+LSA configuration gives the higher average precision for all the domains. This is shown in Fig. 3. However this configuration achieves low precision in cases that most of the other methods do so as well. For instance, this is the case for the Domain2/portal.owl pair shown in Fig. 2, where most of the other methods achieve precision equal to 0.3. It seems that some methods are much better for specific ontologies. This can shape the conjecture that the performance of the method chosen for the automatic annotation of services is closely related to the choice of the domain ontology used. This may also help us to reach a criterion for the

selection of the most "proper" ontology to annotate a service. However this is part of our future work.

The exact matching method fails to deliver valid results in almost any set of experiment, as it was expected. Amongst the other two string matching methods, Levenshtein (Leven) outperforms COCLU in all sets of experiments (Fig. 3). This is mainly due to the existence of compound terms with many different characters.

It must be noticed that the precision achieved by string matching techniques is larger than this achieved by the individual VSM method. This is due to the fact that VSM is being misled by the artificial annotations created for the OWL classes. This process adds one or two terms to a class's virtual document, which may lead the VSM method astray. This supports the argument that ontology classes should be commented with rich and accurate natural language annotations for this method to perform adequately. However this adds extra effort to the ontology developers. As far as the LSA-based method is concerned, it must be noticed that WSDL annotations that include descriptions irrelevant to WordNet senses distract in most of the cases this method from finding the correct mappings due to the inclusion of non-relevant terms.

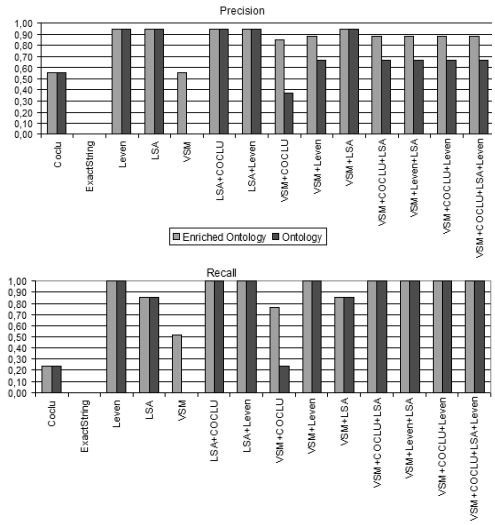

Fig. 4. Precision and Recall for all configurations in experiment Domain1/travel.owl with an artificially enriched version of the ontology

As far as the enrichment of ontologies with informative labels and comments is concerned, as it is depicted in Fig. 4 for the case concerning the Domain 1/travel.owl - a representative case -, the enrichment affects the precision and the recall of the USDS configurations that use virtual documents in the computation of their similarity function. Such an experiment indicates that state-of-the-art mapping methods that are used for the automatic semantic annotation of WSDL documents are more effective when domain ontologies are rich with descriptive information.

Concerning the cases where random strings replace part element names, the lexical mapping methods cannot perform effectively for obvious reasons: strings such as "1qazxsw2" cannot be directly mapped to an OWL class. Even the LSA method that relies on WordNet cannot perform well. This is due to the fact that part names do not

have WordNet entries. In contrast to this, the VSM-based mapping method that does not rely on an external lexicon, can overcome this problem resulting in a mildly good performance. This is shown in the experiment configuration Domain3/Books.owl depicted in Fig. 5. This shows that the adequate and rich annotation of WSDL specifications is vital to the effective performance of the most adequate mapping methods. This is further evidenced by the results provided by VSM in the domain 5, where the "complexity" of the terms involved provides major obstacles to the other methods.

As a final conclusion of the above results, we can state that the USDS configuration that results from the composition of different complimentary methods seems to be the most promising one.

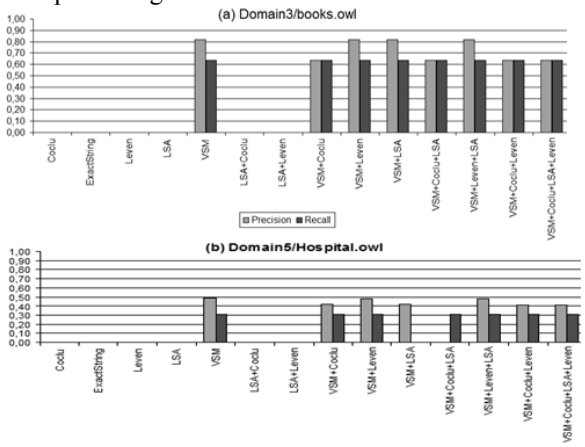

Fig. 5. Precision and Recall for experiments (a) Domain3/books.owl and (b) Domain5/HospitalPhysician.owl with random strings (e.g. "1qazxsw2") as part names

5.3 Top-*k* evaluation

Although we have so far presented a quite extensive set of experiments showing the potential of the approach to support the effective automatic annotation of WSDL specifications, we have in addition measured the effectiveness of the approach to providing assistance to human annotators [17]. Therefore we have measured the accuracy of the approach. Results are presented and discussed in the following paragraphs.

The top-*k* evaluation method has been used for the evaluation of Web information retrieval approaches (e.g. [14, 15]) and also in the evaluation of semantic services' matchmaking systems (e.g. [16]). The top-*k* (or topK) evaluation method measures the percentage of correct results in the top *k* results in a list of ranked results (accuracy of returned ranked results). In our case, WSDL part names are matched against ontology classes, and the most relevant pairs are returned as suggested mappings, together with their ranking. The higher ranked pairs of each pair are being assessed to provide the "correct" mappings. In case of conflict, the method that provided the first mapping wins. Of course this is not a proper policy. Rankings of different methods need to be normalized appropriately so as to rich a global list of

ranked pairs. However, even in this case, a sophisticated policy for reconciling conflicts is necessary: This is something to be further investigated. Therefore, top-*k (accuracy)* evaluation method identifies if the "correct" *pair (i.e. the one provided by the gold-mappings)* is among the *k* ranked suggested mappings. *Accuracy* is a widely accepted metric in the Information Retrieval community and has been successfully used for the evaluation of many systems. An example of related work concerning the support of human annotators is reported in [17]. The work is more focused on the simplification of the manual annotation task of services rather than on the validation of top-*k* ranked semantic annotations. Furthermore, the work facilitates the detecting of mistakes in existing annotations, discovering however a relatively small number of annotations.

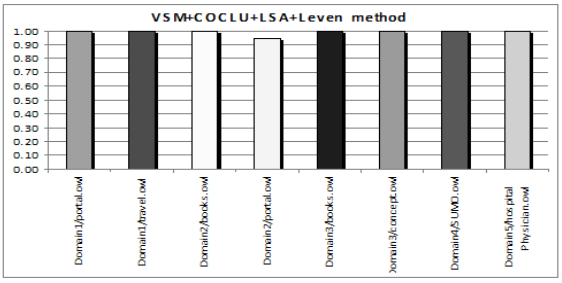

Fig. 6. Top-3 precision for VSM+COCLU+LSA+Leven configuration

The VSM+COCLU+LSA+Leven USDS configuration has produced the best precision/recall percentages in top-*1* experiments. Although recall was really high, the precision of the method can be further improved to support full-automation of the annotation process. Fig. 6 shows the top-*3* precision of this method. As it is shown, the top-*3* precision is 100% for 7 of the domain/ontology pairs and 95% for the Domain2/portal.owl pair. The method failed to retrieve the correct ontology class for a specific part name, although some of its constituent methods successfully retrieved it. This is due to the restriction in the number of retrieved ontology classes in combination with the absence of a global normalisation method amongst the match values produced by each method.

6 Concluding Remarks

Automatic service registration, to facilitate web services' semantic matchmaking, requires efficient and effective semantic annotation of services. In this paper we conjecture that this is possible when WSDL specifications are adequately annotated with textual annotations, and in cases where the suitable mapping method(s) are selected for mapping WSDL elements to ontology classes. Since the efficiency of a mapping method is influenced by the characteristics of the ontology(ies) and the WSDL annotations used in task, their careful selection and composition lead to better performance. In an application context, where users are being advised for the annotation of WSDL parts, the method resulting from the combination of lexical and vector-based methods (Coclu, VSM, LSA, Leven) proved to be extremely valuable

since it manages to return with high accuracy the "proper" class for annotating each of the WSDL message parts among the 3 highly ranked matching pairs.

In this paper we have presented extended experimentation cases with different configurations of the USDS method for uncovering web services' data semantics. We have reached a configuration where different complimentary methods achieve very good performance is a set of experiments of varying difficulty. Future goals that have been indicated throughout the sections of this article provide the many different facets of future work that this work entails.

References

1. Oundhakar, P. S., Sheth, A., Verma, K.: METEOR-S Web Service Annotation Framework, WWW 2004, 553-562 (2004)
2 Stroulia, E., Wang, Y.: Structural and Semantic Matching for Assessing Web-Service similarity, IJCIS 5:14, 1-30 (2004).
3. Klusch, M., Fries, B., Khalid, M., Sycara, K.: OWLS-MX: Hybrid OWL-S Service Matchmaking, AAAI (2005).
4. Klusch, M., Fries, B. Hybrid OWL-S service retrieval with OWLS-MX: Benefits and Pitfals, SMR2, 2007.
5. SAWSDL, http://www.w3.org/2002/ws/sawsdl/ (2002)
6. OWLS-TC, http://projects.semwebcentral.org/projects/owls-tc/
7. Valarakos, A., Spiliopoulos, V., Kotis, K., Vouros, G.: AUTOMS-F: A Java Framework for Synthesizing Ontology Mapping Methods, I-KNOW 2007 Special Track on Knowledge Organization and Semantic Technologies 2007, September 5, (2007)
8. Valarakos, A., Paliouras, G., Karkaletsis, V., Vouros, G.: A Name Matching Algorithm for Supporting Ontology Enrichment, In Proceedings of the 3rd Hellenic Conference on Artificial Intelligence (SETN04), LNAI, Vol. 3025, pp. 381-589, Springer-Verlag, Samos, Greece (2004)
9. Heß, A., Johnston, E., Kushmerick, N.: ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services, ISWC 2004, LNCS 3298, 320–334, (2004).
10. Keller, U., Lara, R., Lausen, H., Polleres, A., Fensel, D.: Automatic Location of Services, European Semantic Web Conference (ESWC'05) (2005)
11. Ringelstein, C., Franz, T., Staab, S.: 'The Process of Semantic Annotation of Web Services', In: J. Cardoso (ed.) (2007) Semantic Web Services - Theory, Tools, and Applications, Idea Publishing Group, USA, (2007)
12. OWL-S API, http://www.mindswap.org/2004/owl-s/api/ (2004)
13. Kotis, K., Vouros, G., Stergiou, K.: Towards Automatic Merging of Domain Ontologies: The HCONE-merge approach. Elsevier's Journal of Web Semantics (JWS), vol. 4:1, pp. 60-79 (2006)
14. Liu, J., Dong, X., Halevy, A.: Answering Structured Queries on Unstructured Data. WebDB '06 Chicago, Illinois USA (2006)
15. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., and Zhang, J. "Similarity Search for Web Services". *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)* (pp. 372-383), Toronto, (2004)
16. Basters, U. and Klusch, M.: RS2D: Fast Adaptive Search for Semantic Web Services in Unstructured P2P Networks. 5th International Conference of Semantic Web (ISWC 2006), Athens, GA, USA (2006)
17. Belhajjame, K., Embury, S. M., Paton, N. W., Stevens, R., and Goble, C. A.: Automatic Annotation of Web Services based on Workflow Definitions, 5th International Conference of Semantic Web (ISWC 2006), Athens, GA, USA (2006)