Enhancing the Discovery of Web Services: A Keyword-oriented Multiontology Reconciliation

Carlos Bobed and Eduardo Mena
IIS Department, University of Zaragoza, Zaragoza, Spain
{cbobed, emena}@unizar.es

Abstract— *The success of Web Services as a tool to decouple and distribute different processes is beyond any doubt. On the one hand, their distributed nature makes them perfect to deploy our applications in a network. On the other hand, the service abstraction provides an easy way to develop large scale solutions by using other's software components. However, to reach the full potential of Web Services, the need for an accurate search approach arises. Semantic ones use ontologies to describe what services do, but, what does happen when providers and requesters of services do not use the same ontologies, i.e., the same semantic vocabulary?*

In this paper, we propose an approach to enhance the discovery of Web Services based on the semantic reconciliation of providers and requesters before the discovery process itself. Our system reuses and integrates ontological information extracted from several online pools of ontologies to make possible to: 1) add semantics easily to existing non-semantic services; and, 2) perform a semantic keyword-based search independently of the ontologies used by the provider. Thus, our proposal bridges the semantic gap between requesters and providers.

Keywords: Distributed applications, software tools and environments for distributed platforms, Semantic Web Services

1. Introduction

Web Services are a useful tool to decouple and distribute different processes. Their apparition several years ago revolutionized the development of distributed systems, achieving a deep integration of distributed computing within the Web. Their distributed nature made them perfect to develop distributed solutions. Moreover, the possibility of offering our services and using services from others in a standardized and open way has made possible a new way of facing the development of large scale solutions. Web Services have enabled developers to reach their goals minimizing efforts and costs [1], and the standardization level that they have reached has made them easy to develop, deploy and use.

However, this open scenario in which everyone can publish his/her services has also its drawbacks: The full potential of Web Services cannot be achieved without providing suitable search and discovery methods. This implies to be able to find services with the exact functionality the requester is looking for. For this reason, Semantic Web Services (SWS) were proposed [2]. The main idea behind them is that making explicit (in a formal way) what services do would enable the automatic discovery and execution of Web Services.

In [3], they show two different paths to face the development of SWS (see Fig. 1): We can add semantics to existing Web Services already described in WSDL documents (path 1), or, once we have the proper semantics, we can develop the SWS directly (path 2).

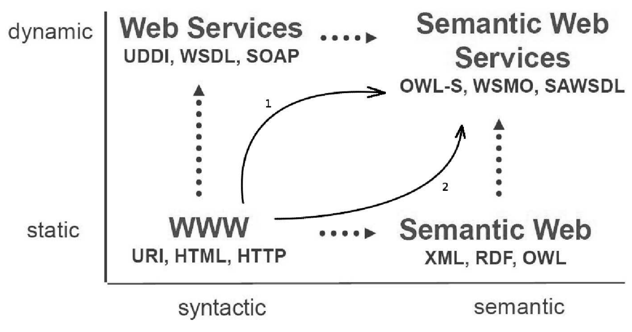


Fig. 1: Two possible paths to semantically describe Web Services.

Adding semantics to existing Web Services can be a time-consuming task. It involves to find or to build from scratch proper domain ontologies, to describe and annotate the service and, finally, to publish it. On the other hand, having a proper semantic description of the service does not ensure that it is going to be easily discovered. For example, there might be a semantic gap between providers and requesters due to the use of different ontologies (different semantics) to describe services and requests, respectively. This semantic gap is difficult to bridge and it is usually handled by the use of ontology mappings between the provider's and the requester's ontologies [4]. This assumption is too strong as it assumes pre-existing semantic mappings between any two given ontologies.

In this paper we propose an approach that enhances the discovery of Web Services. It aims at making independent the discovery process of the ontologies used in the description of published services. This is achieved by semantically reconciling the service descriptions and requests. From the point of view of the provider, our system has the following features:

- It is able to associate automatically to each service a set of *senses* (ontological terms along with their ontological context) by analyzing and processing the descriptions of the providers.
- It associates each published service with all the semantic information, related to the senses describing it, that can be obtained from several online pools of ontologies.

From the point of view of the requester, our system behaves as follows:

- It offers a keyword-based semantic search: keywords are matched automatically to the senses previously generated during the publishing process.
- It is able to use different third party matchmakers by integrating the knowledge about services in ontology modules.

Note that our approach is flexible enough to be applied to any other system which requires to publish and search remote services. The addition of semantics and the semantic reconciliation we propose is going to be needed in any scenario where an agreement between service providers and service requesters must be achieved.

The rest of the paper is as follows. In Section 2, we comment the problem of service discovery and the limitations of current description languages. In Section 3, an overview of the system is given from the points of view of both requester and provider. Section 4 explains the library of semantic descriptions of services. In Section 5, we explain how our system uses the information stored in the previous library to build ontology modules. Related works are presented in Section 6. Finally, conclusions and future work are presented in Section 7.

2. Discovering Semantic Web Services

In this section we overview the general problem for service discovery applied to SWS. In Figure 2, the widely accepted architecture for service discovery is shown. Three are the main agents in such an architecture:

- The service provider, the agent that offers the services.
- The service requester, the agent that looks for a desired functionality.
- The discovery agent, which is in charge of performing the search and matching the request with the descriptions.

One of the main problems of all the semantic approaches and platforms, independently of the subsequent methods of searching and matchmaking, arises from the use of different ontologies as background knowledge to describe the services (ontologies OS_p) and the requests (ontologies OS_r): Providers and requesters are likely to use different *domain vocabularies*. This problem is known as mediation at ontology level or *semantic reconciliation* [5].

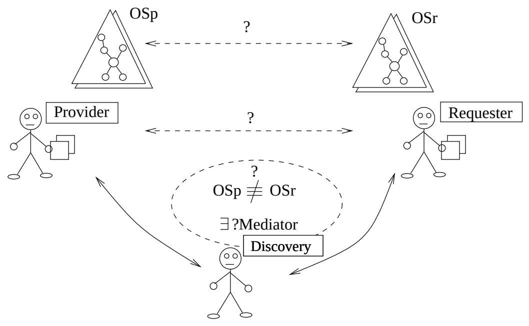


Fig. 2: Main agents in service discovering process

This drawback is handled by using ontology mappings in OWL-S (which establishes the equivalence relationship between terms from different ontologies) or by using ontology mediators in WSMO. However, the building of mappings and mediators between ontologies is a costly process to be done manually as we cannot foresee which ontologies the requester is going to use to specify his/her request: in open scenarios, users can express what they look for by using the semantics from thousands of possible ontologies. That is why the semantic reconciliation should be done in runtime, when the ontologies describing the request are known. Hence, describing the request using ontologies is not feasible for regular users. Therefore, a user-friendly mechanism to express service requests is needed, for example, keyword-based search. These user keywords must be matched automatically to ontological terms that comprises the semantics intended by the user.

3. Overview of the System

In this section, we explain our proposal to publish and search Web Services in a semantic way. To fully understand our approach, and before giving any further detail of the architecture, we have to introduce the exact meaning of *sense* in our system. A sense is the precise meaning of a keyword in a context, i.e. the surrounding keywords determine which meaning has the keyword. In particular, a sense is represented by a tuple formed by the term itself, an ontological context that comprises a list of possible synonyms (with their URIs) and ontological information about the term and a description in natural language. Each ontological context is built by integrating information from different ontologies. Figure 3 shows some possible senses for user keyword *star* retrieved from online ontologies.

For the process of obtaining the possible senses for the user keywords and the disambiguation of the different possibilities that could arise, we advocate using the techniques described in [6], whose main steps we summarize here: 1) for each user keyword, a syntactic matching with ontology terms in online ontology pools is performed, and a sense is built for each matching; 2) the system integrates similar

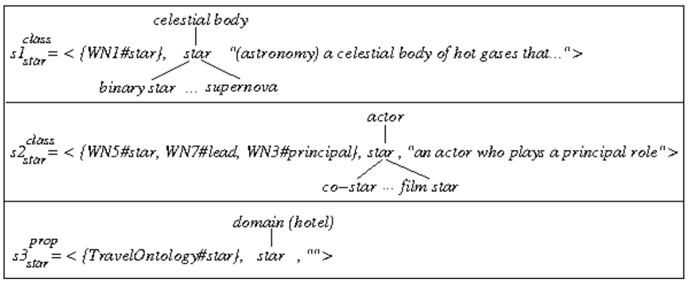

celestial body
s1 class star = < {WN1#star}, star "(astronomy) a celestial body of hot gases that..." >
binary star ... supernova
actor
s2 class star = < {WN5#star, WN7#lead, WN3#principal}, star , "an actor who plays a principal role" >
co−star ··· film star
domain (hotel)
s3 prop star = < {TravelOntology#star}, star , "" >

Fig. 3: Possible senses for keyword “star”.

enough senses or treat them as different ones when a certain similarity threshold is not reached, 3) the possible senses for each keyword are disambiguated considering the possible senses of the rest of the keywords. We refer the reader to [6] for further details on this topic.

This said, in this section we explain the architecture of our system (see Fig. 4). In order to obtain a better understanding of how the system works, we overview it from two separate points of view: the provider and the requester.

3.1 Publishing Services

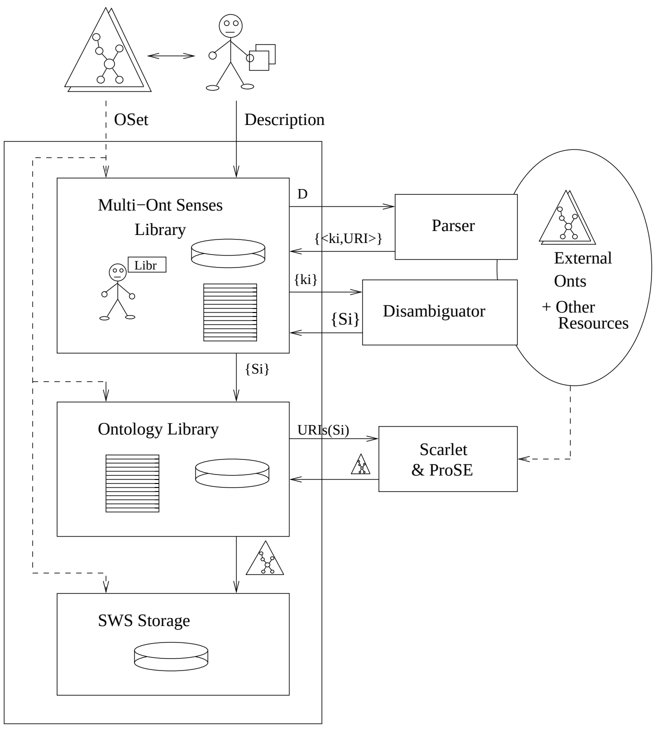


Fig. 4: Overview of the architecture of the system.

To make public a service in our system, the provider can give the service description in two different ways: As a set of plain keywords or a well-formed description in a knowledge representation language with a known and well-defined semantics. Taking the description as starting point, the main steps to publish a semantic web service in our system are (see Fig. 4):

1) *Multi-Ont Senses Library:* When the input is a well-formed description (D), the Parser is in charge of parsing it and obtaining the most important keywords while tracking their source ontologies[1]($< k_i, URI >$). The next step is to consult the senses library with the set of input keywords. This library contains an index of sets of keywords with their possible meanings in the form of senses.
   An intelligent agent, *Librarian*, decides when to build a new entry for the senses or to update and integrate the possibly existing ones. If the *Librarian* has to disambiguate the meaning of the keywords ($\{k_i\}$) or to widen the semantic information that it has for each one of them, it can use the Disambiguator and request the help of the provider to choose the most appropriate meanings ($\{S_i\}$). More details can be found in Sect. 4.
2) *Ontology Library:* Once the system has the senses attached to the input keywords, the Ontology Library integrates an ontology associated to the set of senses, gathering all the semantic information regarding the senses together. Its sources of information are the ontologies referenced in the senses. In this step, external services can be required to extract and discover more information (in particular, our system uses Scarlet [7] and ProSÈ [8]). In Section 5, the different possibilities to integrate the ontologies are detailed.
3) *SWS Storage:* Finally, the original description (if provided) is stored together with the ontological information in the SWS Storage.

Note that publishing the services in this system allows to add semantics to them even when no semantic description has been provided, i.e., taking as starting point a set of plain keywords.

3.2 Requesting Services

From the point of view of the requester, the process is quite different, as the system has to perform different matching operations in the different steps. The service search is mainly designed as being keyword based, although the Parser module can deal with higher-level requests descriptions by, as in the publishing process, parsing them looking for the most important keywords or ontological terms. The main steps of the search process in our system are:

1) *Multi-Ont Senses Library:* The system looks in the Multi-Ont Senses Library for a suitable interpretation of the keywords input by the requester. It tries to obtain different possible sets of senses with different probabilities of being the proper meanings. If there is not

[1]The source ontologies are those referenced in the service description.

such an appropriate interpretation currently available, the agent *Librarian* starts a process of disambiguation and insertion of new senses to make them available to the system.

2) *Ontology Library:* If the system has found an appropriate interpretation with the help of the requester, the system retrieves the associated ontology from the Ontology Library.
3) *SWS Storage:* Finally, a matchmaking process is performed over the SWS Storage. The most basic algorithm would be to retrieve all the services associated to each of the ontologies that have been retrieved in the previous steps, but more sophisticated approaches can be applied as plug-ins (for instance, the system can ask the requester information about the role that the keywords he/she has entered plays in the service he/she looks for, whether they are inputs, outputs, and so on, to focus the discovery process). The answer set of services can be ranked according to the probabilities of the meanings or different values calculated by the different possible matchmaking plug-ins.

3.3 Example of Use

To illustrate the different steps performed by the system, we have taken as reference the services offered by the test collection OWLS-TC3[2]. It consists of 969 service offers and 29 service requests in OWL-S covering different application domains such as travel, tourism and e-health. This collection has been used for the international semantic service selection contest S3 in 2009[3]. In particular, we have chosen a service which, given a book as input, returns its price. We have chosen this one due to the generality of the terms *book* and *price*. Looking for them in Watson[4], returns 3345 and 1239 semantic documents related to each of them, respectively.

Figure 5 shows how the system processes the description of the service *getPrice*. First, the Parser extracts the relevant keywords "book" and "price" out of the service description (we focus only on the inputs and outputs of the service as the main SWSs approaches have them as the most important elements, however, any part of the description of the service might be included in the process). Then, the Disambiguator builds the senses associated to each of them. Finally, the integrated senses are stored in the Multi-Ont Library. In the example shown in Fig. 5, and without loss of generality, we show the references to the domain ontologies used in OWLS-TC3[5] and to WordNet [9], to facilitate the traceability of the results.

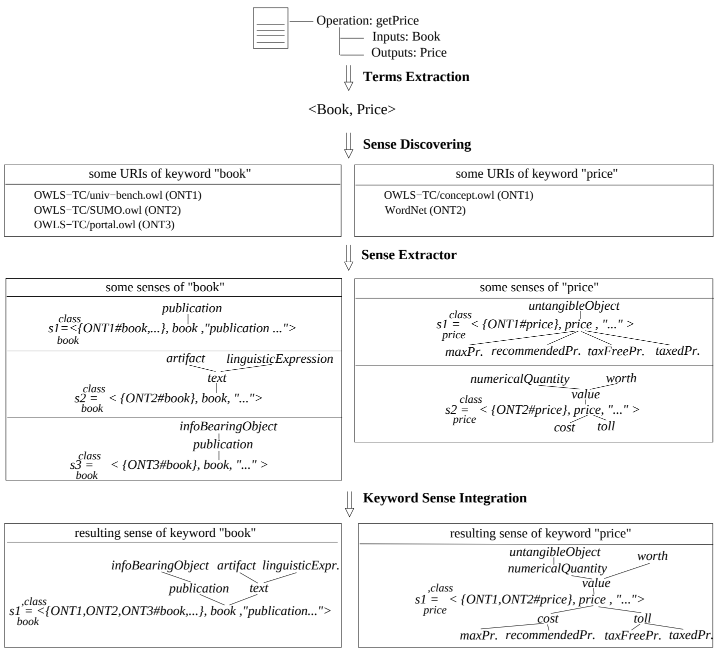
In the following sections, we detail other important modules in our proposal.

[2]`http://projects.semwebcentral.org/projects/owls-tc/`

[3]`http://www-ags.dfki.uni-sb.de/~klusch/s3/`

[4]`http://watson.kmi.open.ac.uk/WatsonWUI/`

[5]They are ontologies retrieved from various public sources on the Web, they are not made specifically for the contest.



Operation: getPrice
Inputs: Book
Outputs: Price
Terms Extraction
<Book, Price>
Sense Discovering
some URIs of keyword "book"
OWLS−TC/univ−bench.owl (ONT1)
OWLS−TC/SUMO.owl (ONT2)
OWLS−TC/portal.owl (ONT3)
some URIs of keyword "price"
OWLS−TC/concept.owl (ONT1)
WordNet (ONT2)
Sense Extractor
some senses of "book"
publication
s1 = <{ONT1#book,...}, book ,"publication ...">
artifact
linguisticExpression
text
s2 = < {ONT2#book}, book, "...">
infoBearingObject
publication
s3 = < {ONT3#book}, book, "..." >
some senses of "price"
untangibleObject
s1 = < {ONT1#price}, price , "..." >
maxPr.
recommendedPr.
taxFreePr.
taxedPr.
numericalQuantity
worth
value
s2 = < {ONT2#price}, price, "..." >
cost
toll
Keyword Sense Integration
resulting sense of keyword "book"
infoBearingObject
artifact
linguisticExpr.
publication
text
s1 = <{ONT1,ONT2,ONT3#book,...}, book ,"publication...">
resulting sense of keyword "price"
untangibleObject
numericalQuantity
worth
value
s1 = < {ONT1,ONT2#price}, price , "...">
cost
toll
maxPr.
recommendedPr.
taxFreePr.
taxedPr.

Fig. 5: Senses obtained from disambiguating "book" and "price".

4. Multi-Ont Senses Library

The Multi-Ont Senses Library is composed of two main blocks as shown in Fig. 6. There is also an intelligent agent *Librarian* which takes care of both.

The first block is the Disambiguation Storage. It contains the different results of disambiguating a set of keywords along with their different probabilities of being the proper interpretation. When a set of keywords is looked up, it returns a probability-ordered list of tuples formed by the probability and a list of corresponding references to the senses. In this search, the information about the synonyms of the keywords is taken into account to avoid missing any possible interpretation. The library tracks the senses with unique IDs, so homonym senses cannot be mistaken (otherwise, the disambiguation process would be useless).

The other block is the Sense Library. It is an storage for the senses maintained up to date by the *Librarian*.

The system can consult the Multi-Ont Library in two ways: By using sense references or plain keywords as input. When using the former only the Sense Library is accessed, while when using the latter the Disambiguation Storage is accessed.

When working with keywords as input, if they do not fully match any keyword set, then partial coverages are considered, and, if the access fails again (or the user declines all the offered interpretations), then the *Librarian* starts a new disambiguation process and the newly obtained senses are inserted. To insert a new sense:

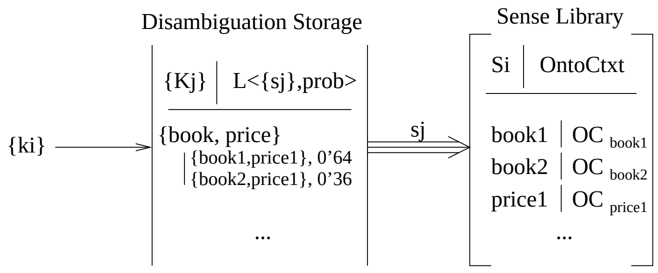


Fig. 6: Organization of the information managed by the agent *Librarian*.

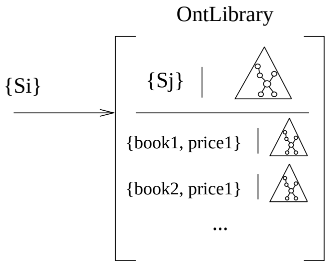


Fig. 7: The ontologies are stored and indexed by their originating senses.

1) The *Librarian* obtains possible synonyms of the newly built sense (the one to be inserted).
2) Then, it looks in the ontological contexts of the already inserted senses for matchings in the list of possible synonyms in order to avoid duplicates in the Sense Library. The result is a set of senses which are possibly equivalent.
3) In a parallel way, it checks whether the new sense and the candidates to be equivalent to it are so. If all of them are not, the new sense is inserted with a new unique ID. Otherwise:
   - The *Librarian* integrates the senses and inserts a new one in the Sense Library.
   - All the references to the former sense are updated and the ontologies in which the sense participates are tagged as obsolete ones.

By inserting new senses and update the existing ones, the system can minimize the effects of the evolution of the source ontologies, as the *Librarian* can take care of the senses being up-to-date. In the next section, we explain how our system uses the information stored in the previous library to build ontology modules.

5. Ontology Library

The different senses stored in the Multi-Ont Sense Library and their different possible combinations can lead to different knowledge sets about the domain they are attached. The Ontology Library is the component responsible for writing down that multiontology knowledge and storing it in different local ontologies in a way that it could be used for discovery issues. Each one of these ontologies are associated to the senses which have taken part in their creation, and are updated as the senses evolve in time (when introducing new senses in the Multi-Ont Sense Library, the already existing senses can be affected and modified). In Figure 7, the information stored is shown. As in the Disambiguation Storage previously explained, the integrated and stored ontologies are associated to a set of references to their conforming senses, and so, the system is able to take into account partial coverages of the inputs when retrieving the ontologies. To integrate the local ontologies, the system considers three levels of ontological information:

1) Sense information: The system considers the information in the ontological contexts. It is the skeleton of the resulting ontology, and, as it contains the original URIs of the terms involved in each sense definition, it allows to extract more information from the original source ontologies. This information is asserted altogether in one knowledge base and enriched with the information further retrieved. If no more information should be available, the resulting ontology would have to be considered as light weighted one (shallow ontology).
2) Simple inter-ontology relationships: The system can discover more inter-ontological relationships using Scarlet [7]. In particular, Scarlet is able to obtain disjointness, inheritance and named relations information (i.e. roles that has the pair of concepts as domain and range or vice versa). It also gives an explanation about how the relationship has been found, and our system only includes it if the reasoning path uses ontologies included in the ontological context.
3) Automatic modularization and ontology reuse: The highest level information is obtained by using ProSÉ [8]. Given a set of terms of an ontology, ProSÉ permits us to obtain a module that is equivalent to the whole ontology in which regards to what can be inferred about the set of terms given. This allows our system to obtain the complete definitions of the terms and all the existing relationships between terms coming from the same ontology.

The possible redundancies in the information (information of different levels may overlap) of the resulting ontology are removed with the help of a reasoner. By asserting all the information and classifying it, we obtain a final version of the ontology. These newly integrated ontologies can be used by the different discovery plug-ins that can be attached to our system. Together, the Multi-Ont Senses Library and the Ontology Library, can be seen as a real implementation of the system envisioned in the position paper [10].

6. Related Work

During the last years, many efforts have been done to ease the Web Services annotation and discovery process. The matchmakers presented in [11], [12], [13] have as assumption that both provider and requester share the ontology which is used to write the constraints for the desired service. Their proposal can be applied as a plug-in into our simpler matchmaker version. In [14], another proposal for matchmaking based on OWL-S is presented. In this system, when providers and requesters use different ontologies, the mappings have to be provided by the requester. In [15], the matchmaking algorithm presented also takes for granted the use of the same ontology or the existence of the mappings. The discovery step presented in [4] again assumes the existence of mappings between source and target ontologies. All these approaches could get rid of their assumption of using the same ontologies for both requester and provider (or having the mappings) by using the ontologies integrated with the information of the extracted senses.

WSMO initiative has its implementation in WSMX framework. The discovery service implemented in WSMX [16] follows the directives asserted in [17], which makes the assumption of the existence of mediators between the ontologies used by the provider and the requesters. Again, that assumption is one which our system get rids of. Another framework based on WSMO is IRS-III [18] but it is oriented to the publication and execution of services covering different aspects of WSMO such as choreography, orchestration and mediation (but at execution time, given the mediators).

METEOR-S WSDI [19] presents an organization for semantically enhanced UDDI registries in a P2P-fashion. The registries are categorized according to a shared ontology. In this way, the system balances the queries, consulting only the appropriate registries. The discovery process requires the user to use a service template using concepts from a domain specific ontology, which raises again the problem of sharing the ontology. In [20], the system is extended with a method for allowing providers and requesters to use different ontologies. This method only takes into account the two provided ontologies (provider's and requester's ones), and not the context of the search as ours do (in the disambiguation process, a dynamic pool of ontologies is consulted to build the disambiguated multi-ontology senses). Moreover, our method enables to add complete semantics to plain keywords search by building dynamic ontologies.

In [21], the authors propose a system to reach global agreement in semantic P2P networks by dynamically spreading and evaluating local mappings between the ontologies used by peers. Peers using the same ontology conform a semantic neighbourhood and mappings are only needed to forward queries beyond the borders of the neighbourhoods. However, authors assume the existence of those experts-made local mappings, and, moreover, the use of ontologies of multiple domains is not considered.

Regarding semantic annotation of services, in [22], the authors identify seven desirable requirements for general semantic annotation platforms. Our system holds explicitly six out of seven, namely: Use of standard formats, user centered/collaborative design, ontology support (multiple ontologies and evolution), support for heterogeneous document formats, annotation storage and automation. The seventh requirement is about document evolution (in this case, service description evolution), and is held implicitly by our system: When a change in a service description is needed, the new description is made available to the system and the senses are updated accordingly to the new information.

ASSAM [23] is a tool for annotating SWS that provides the user with suggestions on how to annotate the different elements of a WSDL document according to a previously known and shared ontology. It uses machine learning algorithms to obtain these suggestions out of a initial training set of Web Services with existing semantic annotation. Apart of the drawback of having to be trained for each ontology used to annotate, the annotations suggested are single-ontology sourced, which would lead again to the need of mappings between ontologies in the discovery process.

OntoMat-Service [24] takes a different approach: There is no discovery process, instead, the user browses the SWS which are advertised through user-friendly HTML pages. In the same browser, the framework allows the user to establish mappings between the ontology he/she is using and the services' ones. As opposite to our system, their framework requires the annotator to know and understand deeply both sets of ontologies, the provider's and requester's ones.

Finally, in which regards to semantic reconciliation, there are two approaches related to the automatic ontology mediation: ontology matching and the use of folksonomies. Ontology matching [25], [26] is aimed at finding the equivalences between two ontologies (source and target). We do not want to focus on just two ontologies as we want to widen the knowledge about the senses. In fact, the disambiguation process and the use of senses has the ontology matching as side-effect [27]. On the other hand, folksonomies [28], [29] are information structures that emerge from collaborative tagging. They can help to obtain a shared vocabulary, but they still have the problem of ambiguity and they lack of formal semantics. In [29], they state that although folksonomies can converge to stable tag distributions, they are not directly usable for the Semantic Web.

7. Conclusions and Future Work

In this paper, we have presented a system that provides a new approach to the discovery of Web Services. Based on the notion of *sense*, our system is able to discover, extend and integrate multi-ontology information about the published services, making it be ready for being used in the discovery process. The main features of our proposal are:

- It fills the semantic gap between providers and requester when they are using different ontologies to describe their services and their needs, respectively.
- It enables the provider to describe services in an informal way that would be automatically transformed into a semantic description.
- It enables the requester to perform keyword-based semantic searches.
- It can use and complement different existing discovery, matchmaking and ranking approaches.

Moreover, our approach is flexible enough to be adapted to any system in which semantic reconciliation might be needed. It makes it possible to achieve a semantic agreement between service providers and requesters even when they are not using the same set of ontologies or any ontology at all.

As future work, we want to study how the senses can help to improve the automatic composition of services. We also plan to evaluate our approach using different matchmaking systems against real third party services.

Acknowledgments

This work has been supported by the CICYT project TIN2007-68091-C02-02.

References

[1] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Commun. ACM*, vol. 46, no. 10, pp. 24–28, 2003.

[2] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[3] R. Krummenacher, M. Hepp, A. Polleres, C. Bussler, and D. Fensel, "WWW or What is Wrong with Web Services," in *Proc. of the 3rd European Conference on Web Services (ECOWS 2005), Vaxj, Sweeden*. IEEE Computer Society, November 2005, pp. 235–243.

[4] M. Alberti, M. Cattafi, F. Chesani, M. Gavanelli, E. Lamma, M. Montali, P. Mello, and P. Torroni, "Integrating abductive logic programming and description logics in a dynamic contracting architecture." in *Proc. of 7th Intl. Conference On Web Services (ICWS 2009), Los Angeles, CA, USA*. IEEE, July 2009, pp. 254–261.

[5] R. Studer, S. Grimm, and A. Abecker, *Semantic Web Services: Concepts, technologies, and applications*. Springer, 2007.

[6] R. Trillo, J. Gracia, M. Espinoza, and E. Mena, "Discovering the semantics of user keywords," *Journal on Universal Computer Science (JUCS). Special Issue: Ontologies and their Applications*, November 2007.

[7] M. Sabou, M. d'Aquin, and E. Motta, "SCARLET: SemantiC relAtion discoveRy by harvesting onLine onTologies," in *Proc. of the 5th European Semantic Web Conference(ESWC 2008), Tenerife (Spain)*. Springer, June 2008, pp. 854–858.

[8] E. Jimenez-Ruiz, B. Cuenca-Grau, U. Sattler, T. Schneider, and R. Berlanga, "Safe and economic re-use of ontologies: A logic-based methodology and tool support," in *Proc. of the 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain*. Springer, June 2008, pp. 185–199.

[9] G. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38(11), November 1995.

[10] H. Alani, "Ontology construction from online ontologies," in *Proc. of the 15th Intl. Conference on World Wide Web (WWW 2006), Edinburgh, Scotland, UK*. ACM, May 2006, pp. 491–495.

[11] M. Klusch, B. Fries, and K. Sycara, "Automated Semantic Web Service discovery with OWLS-MX," in *Proc. of the 5th Intl. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan*. ACM, May 2006, pp. 915–922.

[12] M. Klusch and F. Kaufer, "WSMO-MX: A hybrid Semantic Web Service matchmaker," *Web Intelli. and Agent Sys.*, vol. 7, no. 1, pp. 23–42, 2009.

[13] M. Klusch, P. Kapahnke, and I. Zinnikus, "SAWSDL-MX2: A machine-learning approach for integrating Semantic Web Service matchmaking variants," in *Proc. of 7th Intl. Conference On Web Services (ICWS 2009), Los Angeles, CA, USA*. IEEE, July 2009, pp. 335–342.

[14] J. Pathak, N. Koul, D. Caragea, and V. G. Honavar, "A framework for Semantic Web Services discovery," in *Proc. of the 7th Annual ACM Intl. Workshop on Web Information and Data Management (WIDM 2005), Bremen, Germany*. ACM, November 2005, pp. 45–50.

[15] S. Agarwal and R. Studer, "Automatic matchmaking of Web Services," in *Proc. of the IEEE Intl. Conference on Web Services (ICWS 2006), Chicago, USA*. IEEE Computer Society, September 2006, pp. 45–54.

[16] U. Keller, R. Lara, and e. Axel Polleres, "Semantic Web Service discovery, WSMX Working Draft," Tech. Rep., http://www.wsmo.org/TR/d10/.

[17] ——, "WSMO Web Service discovery, WSMO Working Draft," Tech. Rep., http://www.wsmo.org/2004/d5/d5.1/.

[18] L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, V. Tanasescu, C. Pedrinaci, and B. Norton, "IRS-III: A broker for Semantic Web Services based applications," in *Proc. of the 5th Intl. Semantic Web Conference (ISWC 2006), Athens, GA, USA*. Springer, November 2006, pp. 201–214.

[19] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, "METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of Web Services," *Inf. Technol. and Management*, vol. 6, no. 1, pp. 17–39, 2005.

[20] S. A. Oundhakar, K. Verma, K. Sivashanmugam, A. P. Sheth, and J. A. Miller, "Discovery of Web Services in a multi-ontology and federated registry environment," *Int. J. Web Service Research.*, vol. 2, no. 3, pp. 1–32, 2005.

[21] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "The chatty web: emergent semantics through gossiping," in *Proc. of the 12th Intl. Conference on World Wide Web (WWW 2003), Budapest, Hungary*. ACM, May 2003, pp. 197–206.

[22] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 1, pp. 14–28, January 2006.

[23] A. Heß, E. Johnston, and N. Kushmerick, "ASSAM: A tool for semi-automatically annotating Semantic Web Services," in *Proc. of the 3rd Intl. Semantic Web Conference (ISWC 2004), Hiroshima, Japan*. Springer, November 2004, pp. 320–334.

[24] S. Agarwal, S. Handschuh, and S. Staab, "Annotation, composition and invocation of Semantic Web Services," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 2, no. 1, pp. 31 – 48, 2004.

[25] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," in *Proc. of 17th National Conference on Artificial Intelligence (AAAI-2000), Austin, Texas, USA*. Austin, Texas: AAAI Press, July 2000, pp. 450–455.

[26] A. Gal, G. Modica, H. Jamil, and A. Eyal, "Automatic ontology matching using application semantics," *AI Mag.*, vol. 26, no. 1, pp. 21–31, 2005.

[27] J. Gracia and E. Mena, "Ontology matching with CIDER: Evaluation report for the OAEI 2008," in *Proc. of 3rd Ontology Matching Workshop (OM'08), at ISWC'08, Karlsruhe (Germany)*, vol. 431. CEUR-WS, ISSN-1613-0073, October 2008, pp. 140–146.

[28] A. Mikroyannidis, "Toward a social Semantic Web," *Computer*, vol. 40, no. 11, pp. 113–115, 2007.

[29] V. Robu, H. Halpin, and H. Shepherd, "Emergence of consensus and shared vocabularies in collaborative tagging systems," *ACM Transactions on the Web (TWEB)*, vol. 3, no. 4, pp. 1–34, September 2009.