On the Empirical Evaluation of Semantic Web Service Approaches: Towards Common SWS Test Collections

Ulrich Küster and Birgitta König-Ries
Institute for Informatics, Friedrich-Schiller-University Jena,
07743 Jena, Germany
ukuester|koenig@informatik.uni-jena.de

Abstract

Semantic web services have received a significant amount of attention in the last years and many frameworks, algorithms and tools leveraging them have been proposed. Nevertheless surprisingly little effort has been put into the evaluation of the approaches so far. The main blocker of thorough evaluations is the lack of large and diverse test collections of semantic web services. In this paper we analyze requirements on such collections and shortcomings of the state of the art in this respect. Our contribution to overcoming those shortcomings is OPOSSum, a portal to support the community to build the necessary standard semantic web service test collections in a collaborative way.

1 Introduction

In recent years semantic services research has emerged as an application of the ideas of the semantic web to the service oriented computing paradigm. Semantic web services (SWS in the following) have received a significant amount of attention and research spending since their beginnings roughly six years ago [8]. We found that from 2002 to 2006 within the sixth EU framework program[1] alone at least 20 projects with a combined funding of more than 70 million Euro dealt directly with semantic services which gives a good impression of the importance being currently put on this field of research. Research efforts so far have resulted in the proposal of numerous competing frameworks for SWS descriptions such as OWL-S[2], WSMO[3] or SAWSDL[4] and dozens of different matchmaking and discovery algorithms (see for instance [4] for an overview). Despite of this wealth of theoretical work surprisingly little effort has been spent towards the comparative evaluation of the competing approaches [7]. However, without relying on the results of objective and well-founded evaluations, on what basis should researchers decide which approach to enhance and base their following work on? Clearly, the gain achieved by the development of new alternative approaches is marginal, if it remains unclear whether or under which conditions the new approaches improve the old ones. Furthermore, without experiments that prove the applicability and relative advantage of SWS to real world problems, why should industry pick up a particular approach to SWS or the idea of SWS in general? Better evaluations are needed for the future enhancement of the field.

The main problem in evaluating SWS technology is the need for realistic test data, i.e. semantically annotated web services. Studies by Klusch et al. [6] have shown that these are not readily available today. Therefore, there is a strong need to provide suitable test collections. In this paper we investigate what the requirements towards these collections are, how they could be established, and finally present our contribution, OPOSSum. The rest of this paper is organized as follows. In Section 2, we take a closer look at the requirements towards test collections. We then (Section 3) discuss related work in the area and show its shortcomings. In Section 4, we present the design and implementation of OPOSSum, a portal to support the building of better test collections. In Section 5, we describe how we leveraged existing work by integrating the test collections SWS-TC 1.1 and OWLS-TC 2.2 into OPOSSum. Section 6 covers the current status of the project, envisioned usages, and aspects of future work followed by a summary in Section 7.

2 Requirements on SWS Test Collections

As mentioned above, serious evaluation of semantic web services frameworks is considerably hindered by the lack of realistic test collections. In this section, we are going

[1] http://cordis.europa.eu/fp6/projects.htm
[2] http://www.daml.org/services/owl-s/1.0/
[3] http://www.wsmo.org/
[4] http://www.w3.org/2002/ws/sawsdl/

to investigate what requirements towards such collections exist, i.e., we try to describe what the ideal test collection should look like.

2.1 Use Cases and Dimensions of Evaluation

There are a number of different goals the evaluation of a semantic web service framework might have. The requirements towards the test collection will vary depending on which goal is pursued. Among these goals are:

- For a given description language, compare the performance of matchmaking algorithms with respect to precision, recall, scalability, ...
- For a given framework, test its usability, ease of learning, ...
- Compare different frameworks with respect to expressivity, matchmaking capabilities, usability, ease of learning, ...
- All of the tests mentioned above could be done for different types of services, i.e., different domains (e.g., end-user vs. B2B), different choreographies (one step vs. complex), different settings (fully automated vs. only supporting the user), ...

If one looks at the requirements implied by these use cases more systematically, one can identify a number of dimensions that need to be spanned by any evaluation of semantic service frameworks and thus by any test collection:

- The *expressivity* of the formalism employed influences how well the web service descriptions represent the service instances offered by a provider and how well the needs of the client can be represented in a goal description.
- The *scope, supported use cases and level of automation* determines which of the different phases during service discovery [3] are supported by a framework, which level of guarantee is attached to a match and which level of automation can thus be supported. This is closely related to the expressivity of the formalism and the semantic correctness of the matchmaking, i.e. measures such as precision and recall.
- The *usability* of the framework regards the ease to create semantic descriptions, the tool support and the compliance with standards. It also covers the aspect whether the modeling of services follows the intuition of a user, which is not always the case [1]. This dimension also includes the *effort* of creating and maintaining the framework, e.g. to agree on common ontologies or update existing ones.
- *Performance and scalability* regards the runtime properties of a system.
- Finally, the *level of decoupling* measures the openness of a system, i.e. how closely the client and the provider are linked in the overall process and to what extent and how easily the goal and web service descriptions can be created independently of each other, still yielding correct results.

Obviously, these measures are correlated, some of them negatively. If natural language is used as the description formalism, expressivity will be high, usability excellent, the effort minimal, and the level of decoupling very good. On the other hand, runtime performance will be poor and automation will nevertheless be impossible to achieve. If, on the other hand, a controlled vocabulary of business classification identifiers is used, the effort, usability, level of decoupling, performance, and scalability will be very good but the expressivity will be poor only supporting very few and simple use cases.

Discovery frameworks balance in this design space of service discovery. Thus, the main task of an evaluation should be to make the chosen tradeoffs and the resulting consequences explicit, transparent and comparable.

2.2 Desirable Characteristics

The dimensions listed above translate into a number of desirable characteristics of the test collection:

- Expressivity: In order to evaluate expressivity of a framework, natural language descriptions of services and goals need to be provided, which are then expressed using the different frameworks under evaluation. The provided descriptions need to be as unambiguous as possible, in order to leave as little room as possible for interpretation by the service creator. Since different approaches might be good in modeling different types of services, the collection needs to contain services from different domains and with different characteristics.
- Scope, supported use cases, and level of automation: To illustrate and evaluate which approaches are advantageous in which settings requires a big but in particular diverse collection of scenarios and services covering as many as possible of the different envisioned use cases.
- Usability: Natural language scenario descriptions are required in order to evaluate the difficulty to semantically encode the involved services and goals within a particular framework. A set of pre-defined ontologies may be provided for some scenarios, while for others

the necessary ontologies may need to be created from scratch.

- Performance and scalability: To compare the performance of different frameworks, they need to run against the same set of services. Thus, offer and goal descriptions in different languages for the same set of services and requests are needed to compare systems across formalisms. To effectively evaluate the scalability of approaches, a large testbed is needed. However, caution has to be paid with regard to automatically generated testbeds. Depending on how well these reflect the variety encountered in real-world settings, scalability measures may or may not reflect real-world circumstances accurately. The composition of a testbed, e.g. whether the contained services are very similar to each other or vary greatly, may have strong effects on the performance of certain approaches.

- Decoupling: The test collections need to reflect different people's viewpoints and modeling approaches in order to effectively support testing for this aspect. Thus, preferably many people should contribute to their development.

In summary, an ideal test collection needs to be fairly big, be composed of contributions by many different people, cover different domains, and contain both, unambiguous, natural language descriptions of services and semantic descriptions for these in different formalisms. In the following section we will review the state of the art with these requirements in mind.

3 State of the Art and Related Work

As mentioned, Klusch and Xing recently investigated the question "Where are all the semantic Web services today?" [6]. They used a specialized crawler to search the surface Web for OWL-S, WSML, WSDL-S and SAWSDL services and discovered 1439 descriptions. Out of these, 1133 were found inside the OWL-S Test Collection 2.2[5], 241 were found in the SWS-TC 1.1, another OWL-S test collection[6], and only the remaining 65 descriptions (4.5%) were found outside of those two artificially synthesized collections. This sheds light on some problems.

First, as the authors remark, "the reported preliminary experimental result does not reflect the strong research efforts carried out in the SWS domain world wide in the past few years, independent from the status of maturity of SWS technology and implied low adoption by end users yet" [6]. Given the amount of effort involved in creating semantic web services as test data, the level of reuse and sharing of such data among researchers obviously needs to be improved.

Second, despite of significant research efforts outside the OWL-S community (e.g. on WSMO or WSDL-S) there is virtually no publicly available test data for SWS tools and algorithms relying on formalisms other than OWL-S (only 29 of 1439 descriptions were not specified in OWL-S).

Third, even though the situation for OWL-S services seems less critical than for other formalisms, the findings highlight that apart from the much smaller SWS-TC, OWLS-TC 2 is currently the only vehicle for quantitative evaluation of OWL-S based SWS technology. Unfortunately OWLS-TC 2 was not meant to be the standard collection for SWS evaluation: "Please note, that no standard test collection for OWL-S service retrieval does exist yet. As a consequence, OWLS-TC can only be considered as one possible starting point for any activity towards achieving such a standard collection by the community as a whole" (OWLS-TC 2.2 manual). OWLS-TC has been created by the effort of a single group and historically been developed to be balanced with respect to a particular matchmaking approach [5]. As a result of this it focuses on a specific use case, a specific modeling approach and a specific formalism. Furthermore, it contains a number of peculiarities and practical weaknesses [7]. Overall, OWLS-TC does not meet most of the requirements listed in Section 2 and is thus not yet ready to be used for a broad and general comparative evaluation of different matchmaking approaches. Even though OWLS-TC is creditably by far the best available starting point towards building a standard test collection, this standard test collection has yet to be built.

The previous section has shown that the requirements towards test collections are quite daunting. Our analysis above illustrates, that the only possible candidate for a standard test collection is far from meeting them. Building a collection that meets even only a significant subset of the listed requirements will require a lot of effort, to be more precise, it will require more effort that can be supplied by any single group. Even if there were a particularly resource-rich group, the need for impartiality would still make it undesirable to have this one group develop the test collection alone. As a consequence, community involvement is crucial for successfully building test collections.

However, even though the existing collections OWLS-TC and SWS-TC have been publicly available for quite some time now, they have not been improved and updated other than by their original authors. This illustrates that it is not easy to achieve community involvement. In our opinion, one prerequisite for obtaining the necessary contributions from the community is to offer appropriate tools that make contributing as easy and effortless as possible while offering a significant and obvious gain. These tools, of

[5] http://projects.semwebcentral.org/projects/owls-tc/
[6] http://projects.semwebcentral.org/projects/sws-tc/

course, need to support the entering of new services and their descriptions, but should also support the integration of existing collections. While not fulfilling all requirements towards an ideal test set, these collections are obviously important building blocks on the way to such a set and should therefore be seamlessly integrated into any new attempt.

Recently, the authors of OWLS-TC have set up the SWS-TC-Wiki[7] to foster community participation in improving OWLS-TC. This has raised the awareness of the problem, however, SWS-TC-Wiki is a fairly lightweight approach: it is agnostic to the contents or structure of service descriptions and does not provide more structuring than a file system would. As a consequence, it enables easy sharing of semantic descriptions but does not allow to edit existing descriptions easily. It also cannot support to search for descriptions with particular properties and to compose collections based on results of such searches. In the following sections, we will present our contribution to the solution of the problems discussed above: OPOSSum, the Online POrtal for Semantic Services, a tool that supports the community effort of building up high-quality SWS test collections.

It should be mentioned here, that in addition to efforts to build up test collections, a number of related efforts exist, that focus on the comparative evaluation of semantic service frameworks via given scenarios or tasks. These are the Semantic Web Services Challenge[8], the W3C SWS Testbed Incubator Working Group[9] and the S3 Contest[10]. All these initiatives acknowledge, that better test collections are needed for meaningful evaluations.

4 Design and Implementation

4.1 Design Goals

According to the discussion in the previous sections, the design of OPOSSum has been motivated by three main objectives:

Goal 1: Promote exchange and reuse of existing data. As mentioned above, there must be many more SWS descriptions around than were found by the experiment by Klusch and Xing. Despite of major projects in the field in Canada, Asia, or the pacific rim, for instance, Klusch and Xing did not find any public semantic web services outside of the US, Europe and Iran. Apparantly, most SWS descriptions developed within research projects remain hidden in private repositories. Given the amount of effort involved in creating SWS descriptions, existing data has to be shared more efficiently. Besides, test collections comprised of services from different sources would also increase the objectivity and relevance of evaluations performed with them. Thus, sharing, reusing, and editing existing data must become easier.

Goal 2: Improve structure, documentation, and usability. The few public as well as private SWS collections that the authors know of are generally poorly documented, poorly structured, and usually come in form of collections of flat files, which do not support convenient browsing nor powerful search. This limits their usability and keeps people from actually reusing them. Future collections must be improved in this aspect. To make this happen, this must be supported by tools.

Goal 3: Support reuse and comparisons across formalisms. Besides the effort to semantically annotate a given service, it is also far from trivial to come up with meaningful, rich, and diverse services in the first place (one of the obvious lessons learned from existing collections). Building a SWS test collection requires to gather (potentially fictitious) services and to semantically annotate them. Both steps are similarly challenging and time-consuming. Thus a collection of meaningful services in any description format (even natural language) is of great value when constructing a collection in a particular formalism. Furthermore, testbeds for different SWS description formalisms should not be isolated, as they currently are, but instead be closely interlinked. This supports the direct comparison of different description approaches for the same set of services, thereby allowing to investigate the trade-offs of the various approaches more easily.

4.2 Data Model

As a result of Goal 2, unlike existing file-centered collections, OPOSSum is built on top of a relational database. Figure 1 shows a slightly simplified picture of the data model of OPOSSum that was developed according to the goals listed above.

Unlike all existing collections, OPOSSum's data is structured around the notion of a *Service*, independent from a particular service description. This promotes the reuse and comparison of service descriptions written in different formalisms (Goal 3). Accordingly, a service in OPOSSum is first described by a natural language text. Services can be classified in possibly overlapping *Categories*.

To add more structure and support more precise searching, a service's *Parameters* (inputs and outputs) should be declared explicitly and described in natural language in addition to the general description of the service (Goal 2). To add more semantics without binding to a particular formalism, parameter types are mapped to WordNet synsets[11], thus

[7] http://www-ags.dfki.uni-sb.de/swstc-wiki/

[8] http://sws-challenge.org

[9] http://www.w3.org/2005/Incubator/swsc/

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

[11] WordNet (http://wordnet.princeton.edu/) is a semantic lexicon for the English language developed at Princeton University. It uses the notion of synsets to collect synonyms and disambiguates homonyms. Sense keys

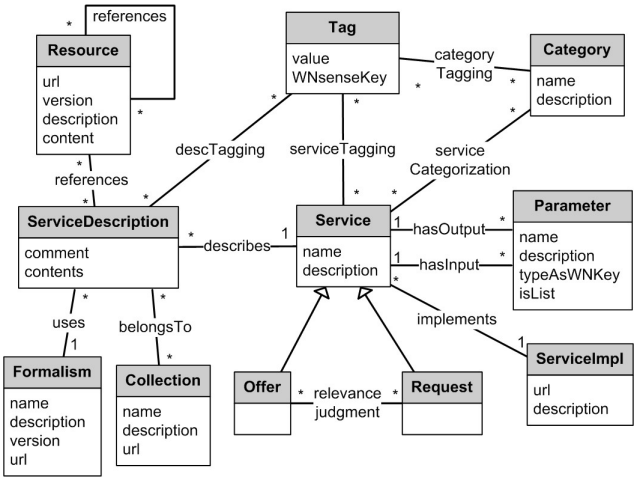

Figure 1. Simplified data model of OPOSSum

providing a kind of semantics that ensures an excellent compromise between being unambiguous, flexible, easily usable and language/formalism independent. We report on some experiences using WordNet in Section 5.

While some approaches to service matchmaking model service offers and requests alike (e.g. OWLS-TC), others model them differently. This enables to distinguish between more generic offers (like a flight booking service) and concrete requests (like a booking request for a particular flight). To accomodate both views, OPOSSum explicitly distinguishes between service *Offers* and *Requests*. OPOSSum's data model allows to store different user-based *Relevance judgments* between an offer and a request and uses a multi-valued metric for those judgments, which better reflects the capabilities of current matchmaking approaches than the binary relevance judgments used so far in the state of the art.

Pointers to *Service implementations* (e.g. a web service) may be listed for an OPOSSum service.

Service descriptions written in any *Formalism* (WSDL, OWL-S, SAWSDL, WSML, ...) are collected by attaching them to the service (request or offer) that they describe. As mentioned above, this should ease the creation of descriptions in different formalisms and support the comparisons of different descriptions for the same service (Goal 3). Service descriptions may be grouped to *Service collections*.

Resources like ontologies or schemas can be added to the system as an independent entity. Descriptions may then refer to the resources that they import. This allows to compute the set of necessary resources for a given set of service descriptions and thus to automatically assemble test collections with all necessary resources on the fly.

To enable users to mark services, OPOSSum allows adding *Tags* to services as well as service descriptions in a Web 2.0 fashion. To support more powerful searching, those tags may optionally be linked to WordNet sense keys to disambiguate their semantics.

4.3 Implementation

To enable easy sharing, reusing, and editing of existing data (Goal 1), OPOSSum has been implemented as a PHP-based web application[12]. This way – unlike with the existing collections – anybody willing can easily contribute to the collection hosted in OPOSSum online in a wiki-like fashion. Users may search for services or descriptions based on their properties, compose and download collections based on search results, add new services and service descriptions to the database, store relevance judgments for offers and requests, improve the documentation of existing data, update descriptions, fix errors and inconsistencies in referenced ontologies, etc.

Currently, OPOSSum already contains nearly 1400 descriptions for around 1300 services, making it the largest publicly available semantic service test collection. However, nearly all of those description have been integrated from the beforementioned existing test collections OWLS-TC 2.2 and SWS-TC 1.1. In the following section, we will describe how the collections have been integrated and share our experiences in doing so.

5 Integration of Existing Test Collections

To the best of our knowledge, OWLS-TC[13] and SWS-TC[14] are the only sizeable publicly available test collections of semantic web services. In this section we provide information about these collections and describe how they have been integrated in OPOSSum.

5.1 OWLS-TC

OWLS-TC is by far the largest publicly available collection of semantic web services. The current version 2.2 of OWLS-TC contains 1003 services written in OWL-S 1.1, 69 of which are also provided in OWL-S 1.0. OWLS-TC is organized as a collection of files, each containing a single service description. The OWL-S 1.1 services are classified into seven domains (communication, economy, education, food, medical, travel, and weapon). We parsed the 1003 service descriptions and extracted the following data from each of them:

are used to reference synsets, thus providing a unambiguous identifier of a particular semantic meaning.

[12] http://hnsp.inf-bb.uni-jena.de/Opossum/

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

[14] http://projects.semwebcentral.org/projects/sws-tc/

- domain the service was listed under
- resources (ontologies) imported by each description
- service name as specified in the profile
- textual description as specified in the profile
- service inputs and outputs:
  - parameter name
  - parameter type (as an ontological reference)
  - parameter description as specified in the label attribute of a parameter (although next to none were provided)

Overall, the 1003 services used 2939 parameters with 417 different parameter types. These were manually mapped to WordNet sense keys as used in OPOSSum. Complete information about the performed preprocessing and the resulting type mappings is available online[15].

From the data obtained from each of the 1003 service descriptions we inserted a corresponding service in OPOSSum and attached the original description for that service to the new service entry. The 69 OWL-S 1.0 service descriptions additionally contained in OWLS-TC 2.2 were later added to the corresponding service entry. The generated service entries were categorized according to the domains under which they were listed in OWLS-TC 2.2.

Unfortunately the generated entries do contain errors, mostly related either to incorrect mappings of parameter types to WordNet sense keys or to flaws in the original OWLS-TC 2.2 data. We will illustrate this by some examples. OWLS-TC 2.2 contains numerous services that were apparently created by slightly variating existing services. This results in some copy and paste errors. The service "award_scholarshipduration_SwissGovservice.owls" for instance, has a service name "GermanGovernmentAwardScholarshipService" and is described as a service that provides information about scholarships by the German government. The service "_price_Fishservice.owls" is described as a service that returns the price of dried fish offered by a company, but is named in the profile as "CanonCameraPriceService". Such trivial errors are aggravated by the fact that the services in OWLS-TC 2.2 are described with little detail. The above mentioned service delivering the price of dry fish for instance is described solely by its single output parameter of type price ([7] analyzes some more problems in OWLS-TC). While we created the WordNet type mappings, we encountered some parameter types which referred to non-existing concepts in the ontologies, mostly caused by confusing references to the SUMO and MILO ontologies[16]. We fixed those errors when we encountered them but we did not explicitly search for them. Most likely, the data still contains some erroneous references.

[15] http://hnsp.inf-bb.uni-jena.de/Opossum/index.php?action=faq

[16] http://www.ontologyportal.org/

In addition to the flaws contained in the original OWLS-TC data, more errors were likely inserted during the process of mapping ontological parameter types to WordNet sense keys. These result from the fact that those mappings were made based on the referenced ontological types independent from the actual services. As a result they do not always describe the parameter types very well. First, the ontological types did not always have a perfectly matching entry in WordNet. Second, the ontological types themselves did not always capture the semantics of the parameter types precisely in the first place. Obviously, these two sources of errors can potentially combine in an unfortunate way. We report on some of the problems encountered when creating the WordNet mappings below. The automatically generated service entries were added to a specific category ("Incomplete - OWLS-TC") to mark that they could be considerably improved, if a human took some time to manually edit them. However, due to the size of OWLS-TC 2.2 this can only be done by the community as a whole. It is one of the achievements of OPOSSum, that this is now easily possible online.

OWLS-TC 2.2 contains roughly two dozen different domain ontologies that are referenced by the services. During the process of creating the type mappings, we realized that these contain concepts which are defined multiple times in the different ontologies without being explicitly related to each other. For some settings, this situation may be more realistic than one with a single unified ontology, thus, this is not necessarily a flaw of OWLS-TC, but it is a feature worth noting.

5.2 SWS-TC

SWS-TC 1.1 is another test collection of 241 OWL-S services. Thus, SWS-TC 1.1 is much smaller than OWLS-TC, but it seems that the average quality of the descriptions is somewhat better (e.g. the level of documentation) and that it contains fewer services created by only slight variations of existing ones.

The services from SWS-TC 1.1 have been parsed and added to OPOSSum exactly like those from OWLS-TC 2.2. The WordNet mappings created for the SWS-TC 1.1 service's parameter types are available together with those created for OWLS-TC 2.2. Overall, the 241 services in SWS-TC 1.1 used a total of 594 parameters with 192 different parameter types, which illustrates the relatively higher variety of these services compared to OWLS-TC 2.2. Unlike OWLS-TC 2.2, SWS-TC 1.1 is based on a single unified domain ontology. Also, services in SWS-TC 1.1 are not classified in domains. Therefore, after their insertion to OPOSSum, a volunteer classified them manually in 12 overlapping categories (in contrast, the categorizations of the services from OWLS-TC 2.2 derived from the domain classification in that collection are disjoint).

5.3 Mapping Parameter Types to WordNet Sense Keys

The main task to integrate the existing collections to OPOSSum consisted of mapping the service's parameter types to WordNet sense keys. These mappings had to be created manually. We looked up the concepts in the ontologies, tried to estimate an English term which captured their semantics best and identified the corresponding WordNet synset using OPOSSum's WordNet API[17]. Although WordNet is likely the most complete and thorough ontology available, a couple of terms were difficult to map to WordNet synsets:

- Many compound terms are missing in WordNet, even though some of them are extremely common: email address, credit card number, account name, user name, dvd player, airport code, airline code, area code, arithmetic computation, full-time respectively part-time position, ...
- Many technical terms are missing, too: SLR (single lense reflex camera), APS (advanced photo system), analog SLR, ...
- A common feature in ontologies is a hierarchy of concepts where general concepts are further restricted by assumptions or conditions. Such concepts did not map well to WordNet which often contains only the base concept. Examples are: cell phone with camera, recommended price, price in Euro, price in Dollar, tax free price, taxed price, ...

In particular the absence of some very common compound terms (like email address or credit card number) is surprising. In some cases, however, it may also be, that we simply did not find the best matching term. The OPOSSum WordNet API performs a quite basic search in the WordNet dictionary, thus, it is often not trivial to come up with the right search term. The term "CryptographyKey" used in SWS-TC's concepts ontology for instance, was mapped to the WordNet sense 106492320 ("a word that is used as a pattern to decode an encrypted message") which is listed under the lemma "key word". A search for "key" lists 21 senses but does not list that particular one because the current API does not perform a substring search if precise matches are found. Also the search terms "cryptographic" or "cryptography" do not yield the above mentioned sense. There is clearly room for improvement here.

6 Status, Usage and Future Work

As described above OPOSSum has been implemented and the services from OWLS-TC 2.2 and SWS-TC 1.1 have been fully integrated into OPOSSum. It is possible to conveniently insert services into the database (including classifying, tagging and linking to WordNet sense keys) and to add descriptions for those services. Unlike with previous collections, services or ontologies in OPOSSum can be edited and their descriptions improved with very little effort.

Please note, that even though OPOSSum already contains nearly 1400 descriptions for roughly 1300 services, the hosted collection is still far from meeting the requirements identified in Section 2. As mentioned above, many of the services so far listed in OPOSSum need to be manually edited and corrected. The natural language descriptions automatically derived from the OWL-S descriptions in OWLS-TC 2.2 and SWS-TC 1.1 are generally fairly poor. Furthermore, we were unable to locate any public SWS collections written in formalisms other than OWL-S. In particular, OPOSSum does not yet contain any service descriptions written in WSML or SAWSDL. As argued, we believe that building a standard SWS test collection exceeds the capabilities of any single group. OPOSSum is meant to be a starting point and tool to help the community as a whole to do so. Everybody is strongly encouraged to go online and join this effort.

Besides the need to improve the quality of the already listed services and descriptions and to add more descriptions in formalisms other than OWL-S, to make it a real retrieval test collection, OPOSSum needs explicit instances of service requests and relevance judments. Service collections used to evaluate the retrieval performance of matchmaking approaches (e.g. in the spirit of the S3 Contest on Semantic Service Selection[18]) need a set of sample requests and relevance judgments identifying the set of services to be retrieved for each request.

Unlike SWS-TC 1.1, OWLS-TC 2.2 readily contains 29 service requests and – for each of these – the set of binary relevant services. To get started, these have been imported to OPOSSum, but ultimately, such binary relevance judgments are too coarse-grained. They ignore the important issue of ranking the relevant services beyond perfect matches and do not reflect the fact, that traditionally next to all semantic service matchmaking approaches apply a multi-valued matchmaking scale (e.g. [9] distinguishes between exact, plugIn, subsumes and fail). Tsetsos et al. deal extensively with the problems of binary relevance judgments in the context of semantic web service matchmaking [10]. Motivated by these problems, OPOSSum supports multi-valued instead of binary relevance judgments. However, for the existing data, these have not yet been created.

For a high quality test collection, relevance judgments need to be assessed manually by human experts. For large test beds, it is impossible to judge the complete set of services for any given query. Thus, before manually judging

[17] http://hnsp.inf-bb.uni-jena.de/opossum/index.php?action=wordnetAPI

[18] http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html

the set of available items, this set needs to be filtered automatically, preferably with a high recall[19] (at the expense of a lower precision[20]). Established evaluation endeavors in the field of information retrieval like in the context of the series of TREC conferences have traditionally relied on pooling approaches [2]. The basic idea is to build the union of the (top k) retrieved items from runs performed with different retrieval systems. The resulting pool of items is manually judged and all other items are assumed to be irrelevant. The quality of this approach correlates with the number of different systems and algorithms used to create the pool.

We are currently investigating how to set up a similar system and how to improve the interface that already allows to add relevance judgments in OPOSSum. This is the main directive of our ongoing and future work in the context of this paper.

7 Summary

This paper argued that more efforts regarding the evaluation of SWS technology are necessary for the further advancement of the field. The main obstacle towards such activity is the pressing lack of large and high quality SWS test collections in various formalisms. Requirements on such collections were investigated and the state of the art examined. It was found that the few existing collections are far from meeting the requirements on good test collections and that more community involvement is essential in order to build the desired better collections. The main prerequisite to obtain the necessary contributions from the community is to offer appropriate tools that make contributing as easy and effortless as possible and allow the community to benefit from such contributions immediately. Such tools have been lacking so far.

The main contribution of this work is a solution to this problem: OPOSSum, a web portal that supports the community effort of building up high-quality SWS test collections. To get started and leverage previous efforts, the existing public collections have been fully integrated with OPOSSum. OPOSSum's web interface enables the community to collaboratively update, correct, and improve the contained descriptions. Additionally, everybody can share his work and extend OPOSSum's dataset by adding new services and service descriptions online with as little effort as possible. The community benefits from a more efficient use of existing resources by having access to a large, public repository of semantic web services, that is well structured and can be much more efficiently searched and browsed than previous collections.

[19] Recall measures how many of the relevant items are retrieved.

[20] Precision measures how many of the retrieved items are relevant.

Acknowledgements

We would like to thank Andreas Krug who implemented essential parts of OPOSSum and Miriam Ries who performed the classication of the SWS-TC 1.1 services.

References

[1] S. Grimm, B. Motik, and C. Preist. Matching semantic service descriptions with local closed-world reasoning. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, June 2006.

[2] D. Harman. Overview of the first Text REtrieval Conference (TREC-1). In *Proceedings of the first Text REtrieval Conference (TREC-1)*, Gaithersbury, MD, USA, November 1992.

[3] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In *Proceedings of the Second European Semantic Web Conference (ESWC2005)*, Heraklion, Crete, Greece, May 2005.

[4] M. Klusch. Semantic web service coordination. In M. Schumacher and H. Helin, editors, *CASCOM - Intelligent Service Coordination in the Semantic Web*, chapter 4. Springer, 2008.

[5] M. Klusch and B. Fries. Hybrid OWL-S service retrieval with OWLS-MX: Benefits and pitfalls. In *Proceedings of the First International Joint Workshop SMR^2 on Service Matchmaking and Resource Retrieval in the Semantic Web at the 6th International Semantic Web Conference (ISWC07)*, Busan, South Korea, November 2007.

[6] M. Klusch and Z. Xing. Semantic web services in the web: A preliminary reality check. In *Proceedings of the First International Joint Workshop SMR^2 on Service Matchmaking and Resource Retrieval in the Semantic Web at the 6th International Semantic Web Conference (ISWC07)*, Busan, South Korea, November 2007.

[7] U. Küster, H. Lausen, and B. König-Ries. Evaluation of semantic service discovery - a survey and directions for future research. In *Postproceedings of the 2nd Workshop on Emerging Web Services Technology (WEWST07) at the 5th IEEE European Conference on Web Services (ECOWS07)*, Halle (Saale), Germany, November 2007.

[8] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[9] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference (ISWC2002)*, pages 333–347, Sardinia, Italy, June 2002.

[10] V. Tsetsos, C. Anagnostopoulos, and S. Hadjiefthymiades. On the evaluation of semantic web service matchmaking systems. In *Proceedings of the 4th IEEE European Conference on Web Services (ECOWS2006)*, Zürich, Switzerland, December 2006.