WSCOLAB: STRUCTURED COLLABORATIVE TAGGING FOR WEB SERVICE MATCHMAKING

Maciej Gawinecki, Giacomo Cabri
Department of Information Engineering, University of Modena and Reggio Emilia, Italy

Marcin Paprzycki, Maria Ganzha
Systems Research Institute, Polish Academy of Sciences, Poland

Keywords: Web service, Discovery, Matchmaking, Collaborative tagging, Evaluation.

Abstract: One of the key requirements for the success of Service Oriented Architecture is *discoverability* of Web services. Unfortunately, application of authoritatively defined taxonomies cannot cope with the volume of services published on the Web. Collaborative tagging claims to address this problem, but is impeded by the lack of structure to describe Web service functions and interfaces. In this paper we introduce *structured collaborative tagging* to improve Web service descriptions. Performance of the proposed technique obtained during the Cross-Evaluation track of the Semantic Service Selection 2009 contest is reported. Obtained results show that the proposed approach can be successfully used in both Web service tagging and querying.

1 INTRODUCTION

The key benefit of utilization of the Service Oriented Architecture (SOA) is a high degree of reuse of components, available as readily deployed Web services (Weerawarana et al., 2005). To achieve this goal, it is necessary to make Web services discoverable. In the traditional SOA vision, service providers register services using a service broker, while service requestors use the same broker to discover them. For the registered services to be usable, service broker must provide all necessary information about their functionality (Hagemann et al., 2007). Until January 2006, the role of service brokers was played mainly by the UDDI Business Registries (UBRs), facilitated by Microsoft, SAP and IBM (SOA World Magazine, 2009). Afterward, a large number of services has been published on the Web (in the form of WSDL definitions) and current service brokers, like SeekDa (SeekDa, 2009), use focused crawling to help facilitate their utilization (Lausen and Haselwanter, 2007; Al-Masri and Mahmoud, 2008a). However, no "central authority" categorizes indexed services according to their functionality (as it was in the case of UBRs). Therefore, WSDL definitions and the related documentation remain the only explicit information defining functionality of Web services. Unfortunately, methods based on the WSDL matching suffer from the *vocabulary problem* (Furnas et al., 1987; Dong et al., 2004; Wang and Stroulia, 2003) and the *intention gap* (Fernández et al., 2008).

In this context, it is often claimed that the *collaborative tagging* of Web services, used by the ProgrammableWeb (ProgrammableWeb, 2009) and the SeekDa service broker portals, has potential to address these issues. Here, the community of users is provided with mechanisms for annotating indexed Web services with keywords called *tags* (Meyer and Weske, 2006). It is a collaborative process because users can see how others tagged the same service and hence the semantics of a service emerges from annotations of the community. The main advantage of this approach is its simplicity. Users do not need to structure their annotation according to multiple, complex features of a service (as it is the case in formal semantic annotation models; e.g. OWL-S (OWL-S, 2009), SAWSDL (SAWSDL, 2009)).

However, for a class of services (including data-centric services, i.e. services that only provide or manipulate data) it would be useful to structure annotation and to explicitly split the functional categorization from the description of the service interface. Specifically, to differentiate between tags describing *input*, *output*, and *behaviour* of a service. We call this

Gawinecki M., Cabri G., Paprzycki M. and Ganzha M.
WSCOLAB: STRUCTURED COLLABORATIVE TAGGING FOR WEB SERVICE MATCHMAKING.
DOI: 10.5220/0002809200700077
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

approach *structured collaborative tagging*.

The paper is organized as follows. We describe the addressed problem in Section 2, and we formalize our approach in Section 3. In Section 4 we describe how the proposed approach can be used for service retrieval. In Section 5 we report results of the Cross-Evaluation track of the Semantic Service Selection 2009 contest (S3, 2009), where the proposed retrieval mechanism took the first place for its effectiveness and for its short query response time. In Section 6 we conclude the approach and present issues for further research in collaborative tagging of Web services.

2 PROBLEM EXPLICATION

We define the *Web service matchmaking* as a task of finding services that match user-specified criteria. We are looking for a Web service *classification schema* that supports this process.

2.1 Matchmaking Criteria

Inspired by the relevance judgment criteria for the Web service matchmaking (Küster and König-Ries, 2009), we identify the following user-specified criteria:

- **Functional Equivalence Criterion.** A user is looking for a specific functionality, for instance, a service that "geocodes" US city names. Such service may take the city name and the state code as input and return the geographic location of the city (longitude and latitude) as output. However, a service that provides geocoding of zip codes still approximates the desired functionality.
- **Functionality Scope Criterion.** A user is looking for a service realizing quantitatively all the required functionality. For instance, geocoding must be done not only for US cities, but for all countries around the world.
- **Interface Compatibility Criterion.** A user is looking for a service with a specific interface (not only of required functionality). For instance, service inputs and outputs should be available in the requested format; e.g. input utilizing zip codes.

Web service matchmaking heuristics should support finding a service with respect to these criteria.

2.2 Web Service Classification Schema

Service *categorization* is a way to find services of similar functionality or functionality scope. For instance, the aforementioned service for geocoding of US cities may belong to the Geocoding, US or Geocoding_US category. Note that it may be not trivial for a user to guess the right category for a required service. Alternatively, a user may define a service request in the form of a *required* service interface. In this case, it is assumed that services of similar interfaces realize similar functionality. This heuristics is called *function signature matching* (Zaremski and Wing, 1995) and used, for instance, by the Merobase software components registry (Merobase, 2009). Here, two services returning a distance in miles, for given zip codes, should realize the same functionality. Independently, for services that provide or manipulate data, the scope of functionality can be also described by their input or output, e.g. the USCity name of an input parameter. However, if service *X* provides the driving distance, whereas service *Y* delivers the straight line distance, they may have compatible interfaces but realize different functionalities. Alternatively, two services providing the driving distance—one for zip codes, and another for cities—are not interface-compatible, but still functionally equivalent. This is because a service interface does not capture the complete semantics of service functionality (Dong et al., 2004). Summarizing, categorization and function signature matching can be seen as complementary heuristics. However, no classification schema exists to provide data for both of them.

2.3 The Vocabulary Problem

When searching for a service, a user may need to know the right keyword(s) for the category this service should belong to, or the right keywords for the input and output parameters. Using the wrong word may result in failing to find the right service. This is the *vocabulary problem* (Furnas et al., 1987). It relates not only to variability of words used to name the same thing. Users may also perceive service functionality differently than how the service provider expressed it in the documentation (Fernández et al., 2008).

Categorization was initially thought as a way to classify re-usable software components. Software components were classified according to authoritatively defined controlled vocabulary. Similarly, UDDI Web service registries and specialized Web service portals (e.g. XMethods) used authoritatively defined taxonomies of business categories (e.g. (UNSPSC, 2009)). Both controlled vocabulary and taxonomies introduce common understanding of service functionality between a service provider (or broker) and a service requestor (user), but still does not resolve the vocabulary problem. The way

that objects have been classified by an authority is very often not obvious for the user (Shirky, 2005). Moreover, hierarchical taxonomies do not allow an object to belong to more than one category. This leads to dilemmas such as: *Whether a service on geocoding of USA addresses should be put into the geocoding or the USA category*. Hence, unless the user gains some intuition on how a particular taxonomy is designed, she needs to drill down through the hierarchy of categories, or guess the right name for it. Finally, an initial vocabulary may become incomplete as the collection of software components grows (Prieto-Díaz, 1991), while most taxonomies are not flexible; e.g. for the UNSPSC taxonomy used by the UDDI registries, it took up to 5 years for a new category to be added (Meyer and Weske, 2006).

The vocabulary problem applies also to the function signature matching heuristics based on the WSDL definitions. This is because, first, WSDL definitions are often sparsely documented (Al-Masri and Mahmoud, 2008b). Second, keywords of input and output parameter names are usually assigned by convention or by the preference of the provider. Hence, they can have related semantics but still be syntactically different (Dong et al., 2004). As a result, different approaches fail when trying to identify parameters meaning the same thing. Introducing a controlled vocabulary (in the form of shared ontologies (Paolucci et al., 2002)) to annotate both a *service offer* (i.e. description of a service in the registry) and a *service request* raises the same problems that the classification heuristics.

3 PROPOSED APPROACH

3.1 Addressing the Vocabulary Problem

The vocabulary problem and related issues occur in the context of Web services, because of: (1) large domain of objects to be categorized, (2) categories difficult to define, (3) lack of clear demarcating lines between them, and (4) lack of categorizing authority(ties). To address these issues, (Shirky, 2005) argued that collaborative tagging may be "better" than utilization of taxonomies and ontologies. Specifically, it was claimed that collaborative tagging allows for non-hierarchical categorization, solving the dilemma of the right category for an object. It also addresses the vocabulary problem, because users may generate large number of tags for an object, and they do it in a similar way they formulate queries (think about the objects) when searching for the object (Furnas et al., 2006).

3.2 Specifying Classification Schema

We follow the idea of (Meyer and Weske, 2006) to use collaborative tagging for classifying Web services but we allow both the behavior and the interface to be described by tags—to provide data for both function signature matching and categorization heuristics. However, an unstructured annotation does not specify if a given tag describes input, output or a behavior of a service. For instance, for a given service, tags find, location and zip are ambiguous. They do not specify whether the service finds a location for a zip code or a zip code for a location. To assess the way people handle such cases we have asked colleagues to tag a number of similar services from the geographic domain. Some resolved this problem by mimicking the interface structure in tags: location_to_zip, coordinates2ZIP, location_from_sentence and find_city. Such "free patterns" can be very difficult to be processed by a machine.

We address the problem by introducing *structured collaborative tagging*. Here, structured tagging implies: categorization of service functionality (*behavior* tags), description of a service interface (*input* and *output* tags) and identification of additional service characteristics, like the functionality scope (*behavior*, *input* and *output* tags). The proposed structure explicitly implies which facets of a service can be tagged, and give a uniform pattern for describing the interface. Note that in the proposed approach, two facets of a service can be tagged with the same tag. For instance, a user may tag both behavior of a service and its output with the distance tag. Hence, the sharp distinction between the behavior and the output, or the behavior and the input is not necessary. Moreover, the larger the number of facets by which a service has been tagged the more likely a user will be able to recall the tagged objects in retrieval (Xu et al., 2006).

3.3 Structured Tagging Model

Formally, we model service functionality utilizing three facets: *input*, *output*, and *behavior*. We describe each of them using formalization of emergent semantics introduced by (Mika, 2005) and adapted for Web service annotation by (Meyer and Weske, 2006).

Definition 1 . A folksonomy $\mathbf{F} \subseteq \mathbf{A} \times \mathbf{T} \times \mathbf{S}$ is a hypergraph $\mathbf{G}(\mathbf{F}) = (\mathbf{V}, \mathbf{E})$ with

- vertices $\mathbf{V} = \mathbf{A} \cup \mathbf{T} \cup \mathbf{S}$, where $\mathbf{A}$ is the set of actors (users and the system), $\mathbf{T}$ — the set of tags and $\mathbf{S}$—the set of services described by service descriptions (service landscape).

- hyperedges $\mathbf{E} = \{(\mathbf{a},\mathbf{t},\mathbf{s})|(\mathbf{a},\mathbf{t},\mathbf{s}) \in \mathbf{F}\}$ connecting an actor **a** who tagged a service **s** with the tag **t**.

In this way we have specified three folksonomies F_i, F_o, F_b for *input*, *output* and *behavior*. We call an *annotation* a single hyperedge (a,t,s) of a folksonomy. To manipulate relations F_i, F_o, F_b we use two standard relational algebra operators with set semantics. *Projection* (π_p), projects a relation into a smaller set of attributes p. *Selection* (σ_c), selects tuples from a relation where a certain condition c holds. Hence, π_p is equivalent to the `SELECT DISTINCT p` clause in SQL, whereas σ_c is equivalent to the `WHERE c` clause in the SQL.

3.4 Quality Tags

In social bookmarking systems, like `del.icio.us`, users are responsible for adding new objects to the system. On the contrary, for Web services, the significant contribution comes from focused crawling of Web and UDDI registries (Lausen and Haselwanter, 2007; Al-Masri and Mahmoud, 2008a). The role of a user limits to tagging services and bookmarking those she found relevant for her task. As a result, the system may contain a number of services without tags, which in turn cannot be recommended by a matchmaker. This is called a *cold-start problem* (Montaner et al., 2003). To address the problem services may be initially assigned with *system tags*. For instance, SeekDa generates system tags both (1) automatically, based on generic features that can be guessed from the endpoint URL of a service (Lausen and Haselwanter, 2007), and (2) manually, by a uniform group of people (authors of SeekDa portal and their colleagues). As a result, there is a minimal overlap between user queries and popular tags; and many services are described by the same combination of tags (Gawinecki, 2009a). We propose to assign system tags manually by a service broker either on the base of parameter names (*input* and *output* tags), or of the WSDL documentation (generic functional categories), like `geographical` (*behavior* tags). Usage of specific facets may lead to more specific and varied system tags.

The second issue we address is steering the community to provide quality tags. The difference between simple keyword annotation and collaborative tagging is the social aspect of the latter. Users can see how other people tagged the same object and can learn from that. (Sen et al., 2006) observed that pre-existing tags affect future tagging behavior. Unfortunately, some users do not tag because they cannot think of any tags or simply do not like tagging. Offering tag suggestions is thus a way to encourage more people to participate in tagging. Specifically, the system presents top 5 tags for a service; that have been already provided by at least two actors (including also the system). During tagging each user is also shown a set of tags she has already used. In this way we try to help user to utilize the same, consistent vocabulary.

4 SERVICE MATCHMAKER

A user may describe a service she searches for in terms of interface it exposes (*input*, *output* query keywords) and the category to which it belongs (*behavior* query keywords). Hence, we model a service request q as a service template: $q = (q_i, q_o, q_b)$, where q_i, q_o, q_b are sets of query terms describing three facets: *input*, *output* and *behavior*, respectively. The proposed matchmaker, called WSColab, (a) classifies service offers as either relevant or irrelevant to the service request, and (b) ranks relevant service offers with respect to their estimated relevance to the service request.

Note some coupling between users annotating services and users formulating queries is necessary to grant that the latter share the vocabulary used by the former ones. *Query expansion* is a recall-enhancing technique to satisfy this need. The query is expanded using a *query autocompletion* mechanism. As a user types query keywords, the system suggests matching tags (completions) for the given facet (maximally the top 15 commonly used tags).

4.1 Service Binary Classification

The matchmaker classifies a service offer as relevant to a service request if they share *input* and *output* tags (function signature matching), or if they share *behavior* tags (categorization). Formally, the results $r(q,(F_i,F_o,F_b))$ of the query q for the folksonomies F_i, F_o, F_b contain only service offers that satisfy the following condition:

$$r(q,(F_i,F_o,F_b)) = r(q_i,F_i) \cap r(q_o,F_o) \cup r(q_b,F_b),$$
$$\text{where} \quad r(q_b,F_b) = \pi_s(\sigma_{t \in q_b}(F_b)),$$
$$r(q_i,F_i) = \begin{cases} \pi_s(\sigma_{t \in q_i}(F_i)), & q_i \neq \emptyset \\ S, & q_i = \emptyset \end{cases}$$
$$r(q_o,F_o) = \begin{cases} \pi_s(\sigma_{t \in q_o}(F_o)), & q_o \neq \emptyset \\ S, & q_o = \emptyset \end{cases}$$

Empty set of query keywords for a given facet means that a user does not care about values for this facet.

4.2 Ranking Services

The matchmaker should rank higher those service offers that are both functionally equivalent and interface compatible to the service request. Service offers that satisfy only function signature matching heuristics or only categorization heuristics should be ranked lower. Degree to which a service offer satisfies signature matching heuristics is the similarity of input and output tags and input and output query keywords. Degree to which a service offer satisfies categorization heuristics is the similarity of behavior tags and behavior query keywords. Hence, combination of those two heuristics can be represented as the weighted sum:

$$sim(q,s) = w_b \cdot sim(q_b, \sigma_s(F_b)) + w_i \cdot sim(q_i, \sigma_s(F_i)) + w_o \cdot sim(q_o, \sigma_s(F_o))$$

of similarity scores for single facets: $sim(q_b, \sigma_s(F_b))$, $sim(q_i, \sigma_s(F_i))$ and $sim(q_o, \sigma_s(F_o))$. Our initial experiments have shown that categorization heuristics is more sensitive to false positives, because it may classify as relevant those services that have similar scope of functionality (*USA*), but are not functionally equivalent. Hence, we give more weight to the *input*/*ouput* facets ($w_i = w_o = 0.4$) than to the *behavior* facet ($w_b = 0.2$).

Similarity for a single facet is measured in the Vector Space Model (VSM). Specifically, tags/keywords for a facet of a single service offer/request are represented as an m-dimensional document vector ($m = |T|$). Tags are weighted using the TF/IDF weighting model (Salton and Buckley, 1988). Similarity between the query keywords and the tags is a cosine similarity between their document vectors. For instance, for the input query keywords q_i and the input tags $\sigma_s(F_i)$ of the service offer s we define the similarity as:

$$sim(q_i, \sigma_s(F_i)) = \frac{\sum_{t \in q_i} w_{s,t}}{W_s},$$

$$\text{where } w_{s,t} = tf_{s,t} \cdot idf_t,\ W_s = \sqrt{\sum_{t \in q_i} w_{s,t}^2},$$

$$tf_{s,t} = \frac{n_{s,t}}{N_t},\ idf_t = log\frac{|S|}{1+|S_t|}$$

where $n_{s,t}$ is the number of actors that annotated an input of the service s with the tag t ($|\pi_u(\sigma_{s,t}(F_i))|$) and N is the number of annotations all actors made for the input of the service offer s ($|\pi_u(\sigma_s(F_i))|$). $|S|$ is the number of all registered services and $|S_t|$ is the number of services having input annotated with the tag t ($|\pi_s(\sigma_t(F_i))|$). Term frequency ($tf_{s,t}$) promotes service offers with tags that many actors used to describe them. Inverse document frequency (idf_t) promotes service offers annotated with more specific tags (i.e. tags that have been used for description of a small number of services). The similarity score $sim(q_i, \sigma_s(F_i))$ promotes service offers sharing more tags with the query. Normalization of similarity measure by W_s allows the similarity score to be unaffected by the number of tags used to describe a facet of a given service offer.

5 MATCHMAKER EVALUATION

An experimental evaluation has been performed during the Cross-Evaluation track of the Semantic Service Selection 2009 contest (S3, 2009). The goal was to compare performance of matchmakers using different formalisms to describe the same test collection—Jena Geography Dataset (JDG50).

5.1 Experimental Setup

Below we briefly report the experimental setup of the contest and describe how the test collection has been annotated in our approach. The complete experimental setup is described in (Küster, 2010).

Performance Measures. Matchmakers have been evaluated using the Semantic Web Service Matchmaker Evaluation Environment (SME^2) (SME2, 2009). The relevance of Web service responses has been checked against *binary* relevance judgments and *graded* relevance judgments (Küster and König-Ries, 2009). Both types of judgments considered functional equivalence of the answer, functional scope and interface-compatibility. Due to limited space we report only the results for graded relevance judgments. Note, however, that the results were stable—the position of WSColab in the ranking of compared matchmakers does not change with respect to the binary relevance judgments.

The performance against the graded relevance judgments has been measured using the $nDCG_i$—a normalized Discount Cumulative Gain at the rank (cut-off level) i (Järvelin and Kekäläinen, 2002). Let G_i be a gain value that the i-th returned service gains for relevance. We define

$$DCG_i = \begin{cases} G_1 & ,i = 1 \\ DCG_{i-1} + G_i / log_2(i+1) & ,i \geq 2 \end{cases}$$

The Discount Cumulative Gain (DCG) realistically rewards relevant answers in the top of the ranking more than those in the bottom of the ranking. Calculated DCG_i is then normalized by the ideal possible

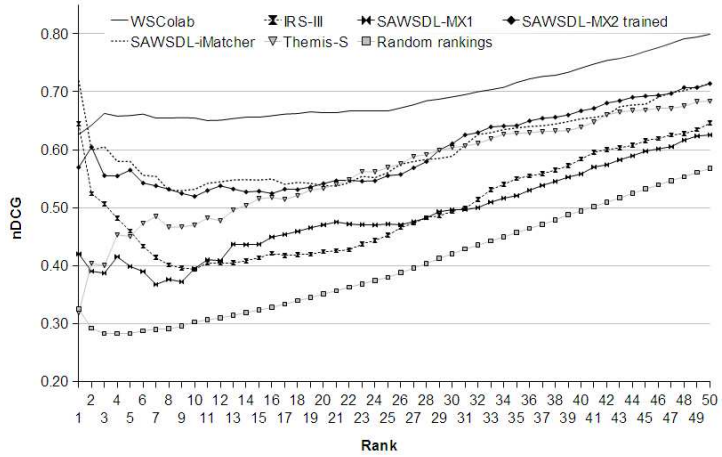

Figure 1: The normalized Discount Cumulative Gain (nDCG) curves for the six different matchmakers averaged over 4 different graded relevance judgments. Shown courtesy of the S3 contest organizers.

DCG_i to make the comparison between different matchmakers possible.

The discount factor of $log_2(i+1)$ is relatively high, to model an impatient user who gets bored when she cannot find a relevant answer in the top of the ranking. We also plot the *nDCG* curve, where the X-axis represents a rank, and the Y-axis represents a *nDCG* value for a given rank. An ideal matchmaker has a horizontal curve with a high *nDCG* value; the vertical distance between the ideal *nDCG* curve and the actual *nDCG* curve corresponds to the effort a user wastes on less than perfect documents delivered by a particular matchmaker.

The efficiency of matchmakers has been measured in terms of average query response time on an Intel Core2 Duo T9600 (2.8GHz) machine with 4GB RAM running Windows XP 32bit.

Test Collection. The test collection provided by the organizers contained *service offers* and *service requests*. The 50 service offers have been annotated by the community using our structured collaborative tagging model (see Section 3).

System tags were generated manually by the organizers of the S3 contest. To collect community tags we developed a collaborative tagging portal (Gawinecki, 2009b), where incoming users were given one of 10 prepared software engineering tasks. For each service in the portal each user has been asked to: (a) tag its *behavior*, *input* and *output*, and (b) classify it as either relevant for the task (potentially useful in the context of the task) or irrelevant. The tagging process has been completed in the open (non-laboratory) environment, where users could come to the portal any number of times, at any time. We invited to participate our colleagues, with either industrial or academic experience in Web services, SOA or software engineering in general. Furthermore, we have sent invitations to the open community, through several public forums and Usenet groups concerned with related topics.

The annotation portal was open for 12 days between September 16 and 27, 2009. Total of 27 users provided 2716 annotations. Our colleagues (17) have tagged 50 services, providing 2541 annotations (94%). The remaining 10 users have tagged 10 services, providing 175 annotations (6%). The contribution of users was significant: 46% to 61% (depending on a facet) of tags were new (not system).

The nine service requests have been annotated in the following way. Each service request was a *natural language* (NL) *query* that needed to be translated into a *system query*. However, our query language is not very restrictive and the same NL query can be translated into different system queries, depending on the query translation strategy used by a user. The choice of a user may have an impact on the final evaluation results of the whole system. Picking up a single neutral user formulating system queries for all the matchmakers (as it is done in (TREC, 2009) approach) addresses this problem. However, it introduces also potential variance in the performance of a single neutral user in using different systems formalisms. This is because for some matchmakers it is far from easy to formulate queries in their formalism. Therefore, we collected query formulations from as many users as possible and the performance of our matchmaker has been further averaged over all query formulations. The collection process has been performed in a more controlled environment than tagging of service offers, to avoid participation of persons who already have seen service descriptions. We extended our annota-

tion portal with a functionality of presenting service requests and collecting system queries from users. A user could not see any services in the registry nor results of her queries. The only information was shared by means of query autocompletion. Additionally, a user could also see the whole vocabulary that has been used during tagging phase to describe service offers. The vocabulary has been presented in the form of three tag clouds, one for each facet of the annotation. No information has been given about which service has been described by which tags.

We collected query formulations from 5 different users. Average length of a query per user ranged from 4.2 to 11.2 words. System queries of different users provided for the same service requests differed from 50% to 100% of words. We observed that users found non-system tags very valuable for describing service requests—the query keywords were system tags, for only 22% for the behavior facet, 26% for the input, and 34% for the output.

Matchmaker Implementation. WSColab indexes terms for each facet of a service using in-memory inverted files, implemented with the `HashMap` standard JDK (JDK, 2009) class. It uses Document-at-a-Time query evaluation algorithm based on accumulators (Zobel and Moffat, 2006).

5.2 Experimental Results

WSColab has been compared with five other matchmakers tested over the same test set and all the results reported here are courtesy of the S3 contest organizers. The competitors included 3 matchmakers based on the SAWSDL formalism, requiring each service to be annotated manually by ontological concepts: *SAWSDL-MX1* (Klusch and Kapahnke, 2008), *SAWSDL-MX2* (Klusch et al., 2009) and *SAWSDL-iMatcher3/1*. Two another were the *IRS-III* (Dietze et al., 2009), which uses the OCML (OCML, 2009) rules, and the *Themis-S* (Knackstedt et al., 2008) ranking service offered over the enhanced Topic-based Vector Space Model (eTVSM) and using the WordNet as its domain ontology.

Figure 1 shows the *nDCG* curves for the compared systems. The performance of the WSColab is the closest to the performance of an ideal one (with respect to the *nDCG* measure). It has a relative performance of 65-80% over most of the ranks while (except for the first two ranks) the remaining systems have a relative performance less than 55-70%. Here, the intuition is that a user needs to spend less effort to find relevant service with WSColab than using other matchmakers.

The average query response time of the WSColab is below 1 millisecond. The second top-efficient matchmaker is the SAWSDL-iMatcher3/1 with 170 milliseconds of average query response time. WSColab is very fast thanks to the simple indexing structure (inverted files). This can be vital for large volume of indexed services and can foster active interaction between a user and the system.

6 CONCLUDING REMARKS

In the context of software component reuse, the thoroughness of component description is limited by the user's willingness to formulate long and precise queries (Mili et al., 1995). We have shown that our model of Web service description and retrieval is a good trade-off between complexity of annotation and query language, and the retrieval quality. However, it is difficult to estimate annotation effort and scalability. First, because tags were generated for a small and specific collection of service offers. Second, because evaluation of collaborative tagging process in the open environment is a difficult problem. Nevertheless, the fact that most of annotations (94%) have been provided by our collegues, not by the open community, may be symptomatic for the Web services on the Web; e.g. 87% of services harvested by the SeekDa are without any tags at all (Gawinecki, 2009a). The process of tagging only selected services *may* be the sign of filtering only services that are valuable for the community. Whether this is the case must be validated in further research.

ACKNOWLEDGEMENTS

We would like to thank to: Ulrich Küster (for the organization of the Cross-Evaluation track), Patrick Kapahnke and Matthias Klusch (for their general support and organization of the S3 contest), Holger Lausen and Michal Zaremba (for providing SeekDa data), M. Brian Blake, Elton Domnori, Grzegorz Frackowiak, Giorgio Villani and Federica Mandreoli (for the discussion).

REFERENCES

Al-Masri, E. and Mahmoud, Q. H. (2008a). Discovering Web Services in Search Engines. *IEEE Internet Computing*, 12(3).

Al-Masri, E. and Mahmoud, Q. H. (2008b). Investigating Web Services on the World Wide Web. In *WWW*.

Dietze, S., Benn, N., Conconi, J. D., and Cattaneo, F. (2009). Two-Fold Semantic Web Service Matchmaking—Applying Ontology Mapping for Service Discovery. In *ASWC*.

Dong, X., Halevy, A. Y., Madhavan, J., Nemes, E., and Zhang, J. (2004). Similarity Search for Web Services. In *VLDB*.

Fernández, A., Hayes, C., Loutas, N., Peristeras, V., Polleres, A., and Tarabanis, K. A. (2008). Closing the Service Discovery Gap by Collaborative Tagging and Clustering Techniques. In *SMRR*.

Furnas, G. W., Fake, C., von Ahn, L., Schachter, J., Golder, S., Fox, K., Davis, M., Marlow, C., and Naaman, M. (2006). Why do tagging systems work? In *CHI*.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Commun. ACM*, 30(11).

Gawinecki, M. (2009a). Analysis of SeekDa Tags for Web Service Matchmaking. Technical report, University of Modena and Reggio-Emilia.

Gawinecki, M. (2009b). WSColab Portal. http://mars.ing.unimo.it/wscolab/.

Hagemann, S., Letz, C., and Vossen, G. (2007). Web Service Discovery - Reality Check 2.0. In *NWESP*, pages 113–118.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

JDK (2009). Java Development Kit. http://java.sun.com/javase/.

Klusch, M. and Kapahnke, P. (2008). Semantic Web Service Selection with SAWSDL-MX. In *SMRR*, volume 416.

Klusch, M., Kapahnke, P., and Zinnikus, I. (2009). SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. In *ICWS*, pages 335–342.

Knackstedt, R., Kuropka, D., Mller, O., and Polyvyanyy, A. (2008). An Ontology-based Service Discovery Approach for the Provisioning of Product-Service Bundles. In *ECIS*.

Küster, U. (2010). JGDEval at S3 Contest 2009 - Results. http://fusion.cs.uni-jena.de/professur/jgdeval/jgdeval-at-s3-contest-2009-results.

Küster, U. and König-Ries, B. (2009). Relevance Judgments for Web Services Retrieval - A Methodology and Test Collection for SWS Discovery Evaluation. In *ECOWS*.

Lausen, H. and Haselwanter, T. (2007). Finding Web Services. In *ESTC*.

Merobase (2009). http://www.merobase.com/.

Meyer, H. and Weske, M. (2006). Light-Weight Semantic Service Annotations through Tagging. In *ICSOC*, volume 4294 of *LNCS*, pages 465–470.

Mika, P. (2005). Ontologies are us: A unified model of social networks and semantics. *J. Web Sem.*, 5(1).

Mili, H., Mili, F., and Mili, A. (1995). Reusing Software: Issues and Research Directions. *IEEE Trans. Softw. Eng.*, 21(6):528–562.

Montaner, M., López, B., and De La Rosa, J. L. (2003). A Taxonomy of Recommender Agents on the Internet. *Artif. Intell. Rev.*, 19(4):285–330.

OCML (2009). http://kmi.open.ac.uk/projects/ocml/.

OWL-S (2009). http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P. (2002). Semantic Matching of Web Services Capabilities. In *ISWC*, pages 333–347.

Prieto-Díaz, R. (1991). Implementing faceted classification for software reuse. *Commun. ACM*, 34(5):88–97.

ProgrammableWeb (2009). http://programmableweb.com.

S3 (2009). Semantic Service Selection contest. http://www-ags.dfki.uni-sb.de/ klusch/s3/html/2009.html.

Salton, G. and Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.*, 24(5):513–523.

SAWSDL (2009). http://www.w3.org/TR/sawsdl/.

SeekDa (2009). http://seekda.com.

Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F. M., and Riedl, J. (2006). tagging, communities, vocabulary, evolution. In *CSCW*, pages 181–190.

Shirky, C. (2005). Ontology is Overrated: Categories, Links, and Tags. http://www.shirky.com/ writings/ontology_overrated.html.

SME2 (2009). Semantic Web Service Matchmaker Evaluation Environment. http://projects.semwebcentral.org/projects/sme2/.

SOA World Magazine (2009). Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort. http://soa.sys-con.com/node/164624.

TREC (2009). Trec REtreival Conference. http://trec.nist.gov/.

UNSPSC (2009). United Nations Standard Products and Services Code. http://www.unspsc.org.

Wang, Y. and Stroulia, E. (2003). Semantic Structure Matching for Assessing Web-Service Similarity. In *ICSOC*.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR.

Xu, Z., Fu, Y., Mao, J., and Su, D. (2006). Towards the Semantic Web: Collaborative Tag Suggestions. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*.

Zaremski, A. M. and Wing, J. M. (1995). Signature Matching: A Tool for Using Software Libraries. *ACM Trans. Softw. Eng. Methodol.*, 4(2):146–170.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6.