Structured Collaborative Tagging: Is It Practical for Web Service Discovery?

Maciej Gawinecki[1], Giacomo Cabri[1], Marcin Paprzycki[2], and Maria Ganzha[2,3]

[1] Department of Information Engineering,
University of Modena and Reggio Emilia, Italy
maciej.gawinecki@unimore.it, giacomo.cabri@unimore.it
[2] Systems Research Institute, Polish Academy of Sciences, Poland
marcin.paprzycki@ibspan.pl, maria.ganzha@ibspan.pl
[3] Institute of Informatics, University of Gdańsk, Poland

Abstract. One of the key requirements for the success of Service Oriented Architecture is *discoverability* of Web services. However, public services suffer from the lack of metadata. Current methods to provide such metadata are impractical for the volume of services published on the Web: they are too expensive to be implemented by a service broker, and too difficult to be used for retrieval. We introduce *structured collaborative tagging* to address these issues. Here, user tags not only aspects relevant for her but also suggested ones (input, output and behavior). Cost, performance and usability of the proposed technique obtained during the Semantic Service Selection 2009 contest are reported. Obtained results suggests that there is no "free lunch." While the method is user-friendly and supports effective retrieval, it still involves cost of attracting the community, and is practical only as complementary one. The analysis shows this is due to user's autonomy as to what, when and how to tag.

Keywords: Web service, discovery, collaborative tagging

1 Introduction

For an application developer there are two key benefits of exploitation of public Web services: (i) a high degree of reuse of software building blocks available as readily deployed services, and (ii) access to data, functionalities and resources that would be otherwise not available to her [40]. For a service provider the benefit of exposing her data, functionality or resources as API is a new distribution channel for her business, research or activity in general [27]. To achieve these goals, it is necessary to make Web services *discoverable*. In the traditional Service Oriented Architecture (SOA) vision, service providers register services using a service broker, while service requestors use the same broker to discover them. In this vision, development of the service registry boils down to implementation of the following methods:

- *Adding Service* as a link and an interface description (WSDL) of a service,
- *Adding Metadata* about functionality of added Web service,

– *Service Discovery* using the provided service metadata.

Until January 2006, the role of service brokers was played mainly by the UDDI Business Registries (UBRs), facilitated by Microsoft, SAP and IBM [36]. Afterwards, a large number of services has been published on the Web (in the form of WSDL definitions) and current service brokers, like `SeekDa.com`, harvest WSDL definitions from the Web and add them to the registry automatically [25,1]. However, there is no longer any "authority" to add/verify/standardize service metadata.

Many current approaches for adding metadata are impractical. They are too expensive to be implemented by the service broker or too difficult to be used for retrieval. The goal of our work is to provide a description method that is *practical*: *affordable* for a service broker, and *usable* for a software developer. The contributions of this chapter toward achieving this goal are as follows:

– We define practical criteria for a description method, illustrate the lack of practicality of current approaches against those criteria and identify challenges to make description method more practical (Section 2).
– We address the identified challenges by defining a practical method for providing metadata (Section 3), type and representation of the metadata (Section 4), and a discovery mechanism operating on those metadata (Section 5). Our method is a variation of *structured collaborative tagging*. The method allows 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 *behavior* of a service.
– We report evaluation results obtained for those criteria during our participation in the Cross Evaluation track of the Semantic Service Selection 2009 contest [33] (Section 6).

2 Problem Definition

Our goal is to define a method for building a Web service registry where finding a required service is not only effective, but also easy for a user and keeps service broker costs of adding metadata low. This means that besides traditional *performance* criteria for service retrieval, we should stress the *managerial* and the *usability* ones. Specifically, we consider:

Performance criteria To develop a registry that enables effective retrieval we need to consider the following criteria (adopted from the Semantic Service Selection (S3) contest [33]):

1. *Retrieval effectiveness*. This criterion reflects the relevance of returned results by the method.
2. *Query response time*. This criteria reflects the average query response time of a matchmaker for a single request.

Managerial criteria To develop a registry that is affordable for a service broker we need to consider the following criteria (adopted from from the related domain of software reuse libraries [29]):

1. *Investment cost.* This criterion reflects the cost of setting up a Web service registry that implements the given method, pro-rated to the size of the registry (tens of thousands of services, e.g in `SeekDa.com`).
2. *Operating cost.* This criterion reflects the cost of operating a Web service registry, pro-rated to the size of the registry.

Usability criteria To develop a registry that is usable for a software developer we need to consider the following criteria (again, from [29]):

1. *Difficulty of use.* This criterion accounts for the fact that the various methods provide varying intellectual challenges to their users.
2. *Transparency.* This criterion reflects to what extent the operation of the Web service registry depends on the users understanding of how the retrieval algorithm works.

2.1 Identifying Challenges

The following approaches to service discovery were compared against the proposed criteria:

- *Taxonomy-based retrieval*: Taxonomies of service categories are used by UDDI Web service registries and specialized Web service portals (e.g. XMethods).
- *WSDL-based signature matching* [39,6]: In signature matching the goal is to find a service operation with a matching signature (input/output parameter names and types) or more generally, a whole service (with a matching operation). Matching is based on syntactical or structural similarity and is used, for instance, by the `Merobase.com` software components registry.
- *Ontology-based service specification matching* [17]: Service specification determines the logic-based semantic relation between service preconditions (expected world state in which a service may be invoked) and effects (how world state is changed by service invocation). For instance, a service S may match a request, if the effect of S is more specific than required and preconditions of S are more general than required.

One can observe these approaches are impractical for building Web-scale registry of Web services for the following reasons (challenges).
A. Operating cost scales poorly with for the number of services in ontology- and taxonomy-based approaches; as they require a person skilled in a given metadata formalism to categorize (annotate) a Web service with respect to an already established taxonomy (ontology) and extend the taxonomy (ontology). In the first case the operation is expensive, if Web services are added automatically from the Web, because the categorization (annotation) cannot be delegated to the provider. In the latter case, if adding new service requires an extension of the taxonomy (ontology), the whole operation can be complex. It might involve reconsidering the complete taxonomy (ontology) and possibly reclassifying (re-annotating) all Web services within the revised taxonomy (ontology).
B. Open-domain registry requires large investment cost As noted, operating cost for ontology- and taxonomy-based approaches is high if the taxonomy

(ontology) is not defined carefully. Defining a taxonomy (ontology) for open range of domains in advance is expensive. To limit the cost, public UDDI registries reuse taxonomies proven in business categorization, e.g. UNSPSC [37]. In case of ontology-based approaches finding ontologies that cover sufficient range of domains with sufficient level of details and integrating them together is prohibitively costly.

C. Query formulation for formal description methods is difficult Overly complex encoding schemes are wasteful unless a service requestor is provided computer assistance in formulating equally complex queries [30]. Thus richer expressiveness of complex approaches like ontology-based specification matching might be not justified comparing to less complex (and thus easier to use) taxonomies and operation signatures.

D. Authoritatively defined descriptions may be not transparent This is often the case that there is a gap between how a user imagines a software component, a Web service in particular, and how it has been described in the registry [9,8]. For instance, the way the taxonomy structure has been defined by the authority, or the service has been classified by a service provider, is very often not obvious for the user [35]. Furthermore, input and output parameter names and types in WSDL definitions are usually assigned by convention or by the preference of the provider [6] that is unfamiliar to a user. Thus, in both cases, using the wrong query keyword(s) may result in failing to find the right service.

E. Supporting complementary retrieval criteria To achieve high effectiveness of service retrieval it is enough to encode, in service metadata, only information that is relevant to the searcher [30]. However, searchers differ in what information/criteria are relevant to them and it is impossible to envision all of them in advance, for instance situation when user will search for a "free of charge" service. Still, none of considered families of approaches support such *ad hoc* search criteria. On the contrary, it is assumed that searchers share a subset of common criteria: functional equivalence, interface compatibility and functionality scope equivalence [23]. Unfortunately, some methods support only part of them. For instance, signature matching considers only interface compatibility (input and output parameters) and ignores functionality equivalence (service behavior); service categorization with respect to a taxonomy is a way to group services of similar functionality or functionality scope but it does not support interface compatibility search.

F. Support for interactive search User often starts a search for a Web service with an unclear goal in mind and thus needs to interact with the system in subsequent query/results steps to find the right Web service. To support such active interaction, the system needs to return not only relevant results, but also to do it in short time. However, many ontology-based approaches employ computationally expensive matchmaking algorithms with implementations too slow to be useful [32].

G. Limited available information about Web service To encode information relevant for a searcher (Challenge E) it is necessary to have this information available. A service provider that has developed a service, has all the knowledge

available but is rarely the case he documents it completely in a WSDL definition [1]. Still, for Web services harvested automatically from the Web WSDL is the only source of information about the Web service, thus its content might be not sufficient to categorize a service or provide its specification.

3 Architecture of Collaborative Tagging Portal

To face the above identified challenges we propose application of collaborative tagging. The technique has been initially proposed for service annotation by [28] as an alternative to authoritatively defined taxonomies. Collaborative tagging relies on voluntary participation of the community. Thus operating cost of describing is limited by putting the burden of description work on the community (Challenge A). The vocabulary used by taggers is not predefined: the process of vocabulary expansion and tagging object with respect to expanded vocabulary is continuous [35], eliminating thus investment cost (Challenge B) and operating cost of re-categorization (Challenge A). The advantage of tags is they can capture aspects of a service that are important for the community but has not been encoded neither in the WSDL definitions from service providers, nor in the authoritative classification; moreover, they use vocabulary more relevant and intuitive to a developer [10,8]. Therefore, the distance between what the developer wants and how it is described in the registry can be minimized [38] (Challenge D).

We propose to put a Web service registry behind an online portal where the process of collaborative tagging and searching of Web services is tightly intertwined. Marlow et al. [26] and Halpin et al. [15] discuss different elements of generic tagging system architecture. Below we address those architectural elements that are challenging for making tagging of Web services practical.

3.1 Supporting User Motivation

Tagging relies on voluntary participation and thus important incentive must attract a user to participate. The works of [14,24,26,31] discuss the number of motivations to tag and incentives to contribute to collaborative online projects (like open source software or Wikipedia). The following appear relevant for the technical community of software engineers: future retrieval (organizational), communal collaboration, information sharing, reciprocity ("I help you hoping you will help me in the future"), and autonomy ("I decide what to tag"). We propose two alternative tagging models to support realization of different subsets of those incentives (summarized in Figure 1):

– *Social bookmarking model* (inspired by such social bookmarking systems as `del.icio.us`). The model supports search task through cooperative organization of the registry content. A user bookmarks a service during discovery to keep a reference to it for further recall. She tags her collection of bookmarks with keywords to organize it, i.e. to ease the process of re-finding a service. Finally, she shares her bookmarks with the rest of the community,

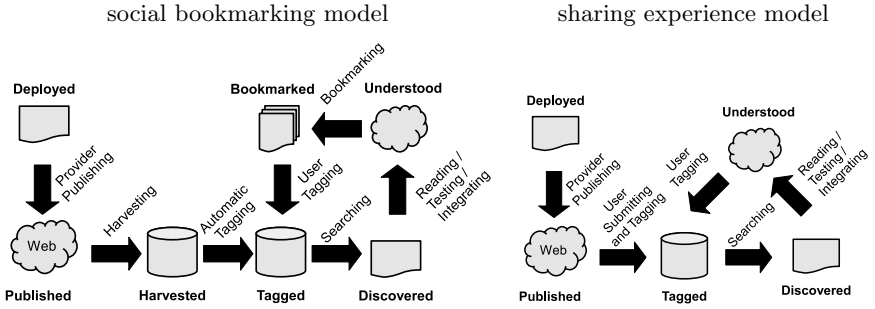


Fig. 1. Two tagging models show changing state of a service and user actions.

because she cannot describe all services in the registry alone and believes others will share their bookmarks with her.
– *Sharing experience model* (inspired by `ProgrammableWeb.com`, the community-driven registry of public APIs). In this model tagging is just one of the means a user has to share his experience about a service with the rest of the community. Other content includes: comments, ratings, additional descriptions, code snippets for invoking a service, applications/mash-ups using the service, how-to tutorials etc.

3.2 Supporting Service Understanding

Users tend not too tag Web services if they do not have any comprehension of them, for instance they have not tried them [4]. However, understanding what a service does in most cases is more complex process than understanding an article (for tagging at `del.icio.us`) or interpretation of a photograph (at `Flickr.com`). Given that initially WSDL definition is the only available source of information (Challenge G), we identified the following means to ease this process: (1) *names and natural language documentation* of service, provider, operations and their parameters, (2) *external site* of the provider (linked in WSDL), (3) *behavior sampling* to test how a service behaves (possible by generating client stub from the WSDL), and (4) *users experience* obtained from the previous means that can be documented in different form (see, the sharing experience model).

3.3 Tag Bootstrapping

A service needs to discovered before it gets tagged but untagged services cannot be discovered if tag-based search is the only search mechanism. This results in a deadlock. It can be prevented by bootstrapping a service with initial tags before it is actually added to the registry. Since services are harvested automatically from the Web, we propose to bootstrap tags using one of tags suggesting algorithms (e.g. [7]). Still, some services can be submitted manually by a user (in experience sharing model), and thus the same user may be encourage to submit also initial tags.

4 Metadata Representation

In our approach 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 (Challenge E). 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 aspects: *input*, *output* and *behavior*, respectively. For instance let us consider the following service request:

> I'm looking for a service to geocode a given US address. The expected service takes as input a structured US postal address (street, city, zip code) and returns a geographical coordinates: latitude and longitude.

One of possible formulations of the service request in our query language is:

$$q_i = (\texttt{street\_name}, \texttt{city\_name}, \texttt{state\_code}, \texttt{US}, \texttt{zip\_code})$$
$$q_o = (\texttt{geocoordinate}, \texttt{latitude}, \texttt{longitude})$$
$$q_b = (\texttt{geocoding}, \texttt{US\_address}, \texttt{usa})$$

To support such queries, the services in the registry should be annotated in a symmetric way. However, an unstructured annotation does not specify if a given tag describes *input*, *output* or *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.

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). Note, that experiments of [38] on retrieval effectiveness have shown that tags cannot be used alone for effective retrieval, but only as a complementary mechanism to traditional classification schemes. This might have been caused by the fact that traditional classification schemes focus on predefined aspects, while tagging on those that are actually important for individual users. Structured tagging supports tagging on both types of aspects (Challenge E).

4.1 Metadata: Structured Tagging Model

Formally, we model service functionality utilizing three aspects: *input*, *output*, and *behavior*. We represent each aspect as a folksonomy. A folksonomy F is a set of annotations (a, t, s), posted by an actor a (a user or the system) who tagged a service s with a tag t. In this way we have specified three folksonomies F_i, F_o, F_b for *input*, *output* and *behavior*. Figure 2 shows examples of such folksonomies. For instance, F_b is defined by the following annotations: (bob, `geographic`, ZipGeocoder), (bob, `geocoding`, ZipGeocoder), (bob, `find`, DistanceCalculator), (alice, `find`, ATMLocator), (alice, `location`, ATMLocator), (alice, `geographic`, ATMLocator), (alice, `geocoding`, ZipGeocoder). It can be seen that to classify a service aspects, an individual user provides one or more tags. Hence, many users may agree to tag a certain service part with the same tag. As a result the

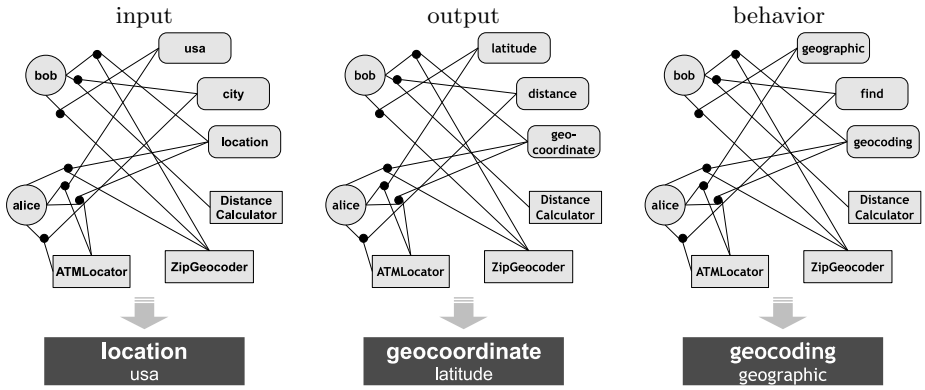


Fig. 2. Example folksonomies for three service parts and descriptions (tag clouds) of the ZipGeocoder service extracted from them. A circle depicts a tagging user, rectangle—a tagged service and rounded box—a tag used. Each black spot defines a single annotation. A tag cloud is a visual depiction of user-generated tags for a given service part. The bigger the tag, the more users provided it.

consensus on what a service does, consumes and returns emerges from annotations of the community. Figure 2 shows what consensus the community achieved on the ZipGeocoder service. For instance, two users agreed that a service does `geocoding`, and one of them classified it also as a `geographic` service.

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.

5 Service Discovery

In service retrieval the community tags play a number of roles: (i) provide a network of links to browse services [28], (ii) facilitate quick understanding about a service, without reading a complex documentation, and (iii) provide metadata for ranking-based search [3,2]. We propose an approach for service discovery that uses tags primary for the latter role. With regard to query formulation our method is easy to use, because it accommodates natural language queries (Challenge C). On the other hand, returned ranking of results may be imperfect (incomplete or irrelevant), thus delegating much of the tedious retrieval/selection/assessment activity to the user. This increases the difficulty of our method. With regard to transparency of our method we assess it as high, because our matchmaking mechanism is straightforward (Challenge D). Firstly, the representations of the query and of the service description are structured in the same way. Secondly, a user in the system can be both searcher and tagger and thus see the mapping between queries and services better. Thirdly, as we

shall see, matchmaking and ranking is based on simple concepts of tag overlap and tag popularity.

5.1 Service Matchmaker

The proposed matchmaker, called WSColab, (a) classifies services as either relevant or irrelevant to the service request, and (b) ranks relevant services with respect to their estimated relevance to the service request.

Service Binary Classification The matchmaker returns services that are either interface compatible (there is at least one matching *input* tag and *output* tag) or functionally equivalent (there is at least one matching *behavior* tag). Formally, the set of results of the query q is defined as $r(q,(F_i,F_o,F_b)) = r(q_i,F_i) \cap r(q_o,F_o) \cup r(q_b,F_b)$, where $r(q_b,F_b) = \pi_s(\sigma_{t \in q_b}(F_b))$ and

$$r(q_i,F_i) = \begin{cases} \pi_s(\sigma_{t \in q_i}(F_i)), & q_i \neq \emptyset \\ S, & q_i = \emptyset \end{cases}$$

($r(q_o,F_o)$ is defined analogously to $r(q_i,F_i)$). Empty set of query keywords for a given aspect means that a user does not care about values for this aspect.

Ranking Services The matchmaker should rank higher those services that are both functionally equivalent and interface compatible to the service request. Interface compatibility is estimated as the similarity between interface (input and output) tags and interface query keywords. Functionality equivalence is estimated as the similarity between behavior tags and behavior query keywords. Hence, the combination of those two heuristics may be represented as the weighted sum of similarity scores for single aspects: $sim(q_b,\sigma_s(F_b))$, $sim(q_i,\sigma_s(F_i))$ and $sim(q_o,\sigma_s(F_o))$:

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

Our initial experiments have shown that the functionality equivalence heuristics is more sensitive to false positives, because it may classify as relevant those services that have similar scope of functionality (e.g. `usa`), but are not functionally equivalent. Hence, we give more weight to the *input/ouput* aspects ($w_i = w_o = 0.4$) than to the *behavior* aspect ($w_b = 0.2$). To estimate similarity between the query keywords and the tags we borrowed a standard cosine similarity measure with TF/IDF weighting model [34] from information retrieval[4] For instance, for the input query keywords q_i and the input tags $\sigma_s(F_i)$ of the

[4] We evaluated also other similarity measures but they performed worse [12].

service s we define the similarity as:

$$sim(q_i, \sigma_s(F_i)) = \frac{\sum_{t \in q_i} w_{s,t}}{W_s},$$
$$w_{s,t} = tf_{s,t} \cdot idf_t,\ W_s = \sqrt{\sum_{t \in q_i} w_{s,t}^2},\ tf_{s,t} = \frac{n_{s,t}}{N_t},\ idf_t = log\frac{|S|}{1 + |S_t|}$$

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 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 ($|\pi_s(\sigma_t(F_i))|$). As a result, service s_1 is ranked higher than s_2 in a single ranking if: (a) s_1 shares more tags with the query than the s_2 does, (b) shared tags of the s_1 have higher relevance weights (more users proposed them) than those of the s_2, and (c) shared tags of the s_1 are more specific (less common among tagged services) than those of the s_2. We also assumed that more people agree on major features than on minor ones and thus the criterion (b) is introduced to avoid assigning high rank to services that share some minor features (e.g. `usa`), but not major ones (e.g. `location`). The criterion (c) is to prevent services with very common tags from domination in the ranking. Usage of ranking method well known in information retrieval gave us the chance to implement the mechanism using the fast query evaluation algorithms based on (in-memory) inverted files [42] (Challenge F).

6 Evaluation

The goal of the evaluation was to empirically validate whether structured collaborative tagging can be a practical solution for Web service description. Specifically, we evaluated our method against three types of criteria (defined in Section 2): managerial, performance, and usability.

6.1 Assessing Managerial Criteria

Here, the goal was to estimate what are the investment and the operating cost for our solution and how they are to scale for large Web service repositories.

Procedure To assess those costs we simulated the process of Web service tagging in terms of the social bookmarking cycle (see Section 3.1). We used the collection of data-centric Web services from the Jena Geography Dataset (JGD) [21]. The 50 services have been annotated by the community using our structured collaborative tagging model. System tags were generated manually by the organizers of the S3 contest (see [13] for details). To collect community tags we developed a collaborative tagging portal [11], where incoming users were given one of ten 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 only source of information about service functionality was documentation and parameter names extracted from the WSDL definitions and other people's tags. Note, that the authors of the JGD test collection had saved the taggers's cognition effort by documenting all missing but relevant information in original WSDL definitions (for this goal they used instruments described in Section 3.2). 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. The contribution of users was significant: 46% to 61% (depending on a service aspect) of tags were new (not system).

Results The only investment cost stemmed from approximately 40 man-hours spent to develop the annotation portal (Challenge B). The operating cost included almost 5 man-hours spent to invite taggers and promote the portal and about 3 man-hours on maintenance of the portal during the annotation process (fixing bugs) and answering to taggers' questions. This shows there is no free manpower in collaborative tagging: remaining operating cost was due to marketing actions required to attract the community (Challenge A). If operating cost is acceptable, a question remains whether such investment grants retrieval effectiveness. In case of tag-based search effectiveness depends on whether someone has described a service that a user searches for in a way she would describe it [35]. Figure 3a shows that people are very selective on which services to tag. Thus, for larger service catalogs, tags are likely to be distributed among services with respect to a power law, like in Web-scale tagging systems for other domains. Particularly, unpopular services may be found in a very long tail of little-tagged and untagged services and thus remain difficult to discover.

6.2 Assessing Retrieval Performance

An experimental evaluation has been performed during the Cross-Evaluation track of the Semantic Service Selection 2009 contest [33]. By participation in the contest we wanted to validate: (1) whether less formal descriptions do not result in worse performance than methods using Semantic Web Services, and (2) whether additional operating cost spent on attracting the community is justified by performance improvement with respect to methods with no operating cost (i.e. methods automatically describing services). We briefly report the experimental setup of the contest and results. The complete experimental setup and detailed results analysis are reported in [22,13].

Competing Matchmakers WSColab has been compared with five other matchmakers tested over the same collection of services and service requests. The competitors with more formal descriptions included: *SAWSDL-MX1* [18], *SAWSDL-MX2* [19], *SAWSDL-iMatcher3/1* [41], and *IRS-III* [5]. The family of methods based on automatically described services was represented by the *Themis-S* [20].

Test Data All participants were given services and service requests from the Jena Geography Dataset [21] to encode in their formalisms. For services we reused service tags collected previously (see Section 6.1). 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*. We collected query formulations from as many users as possible and the performance of our matchmaker has been further averaged over all query formulations. To avoid participation of persons who already have seen service descriptions, the collection process has been performed in a more controlled environment than tagging of services. We extended our annotation portal with a functionality of presenting service requests and collecting system queries from users. User could see neither services in the registry, nor results of her queries. The only information shared was the vocabulary used to describe services during the tagging phase. It was presented as: (1) query suggestions (through query autocompletion technique), and (2) three tag clouds, one for each aspect of the annotation. No information has been given about which service has been described by which tags.

Relevance Judgments The relevance of Web service responses has been checked against *binary* relevance judgments and *graded* relevance judgments [23]. Both types of judgments considered functional equivalence, functional scope and interface-compatibility of the answer. 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 did not change with respect to the binary relevance judgments.

Performance Metrics The retrieval effectiveness against the graded relevance judgments has been measured using the $nDCG_i$—a normalized Discount Cumulative Gain at the rank (cut-off level) i [16]. 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 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

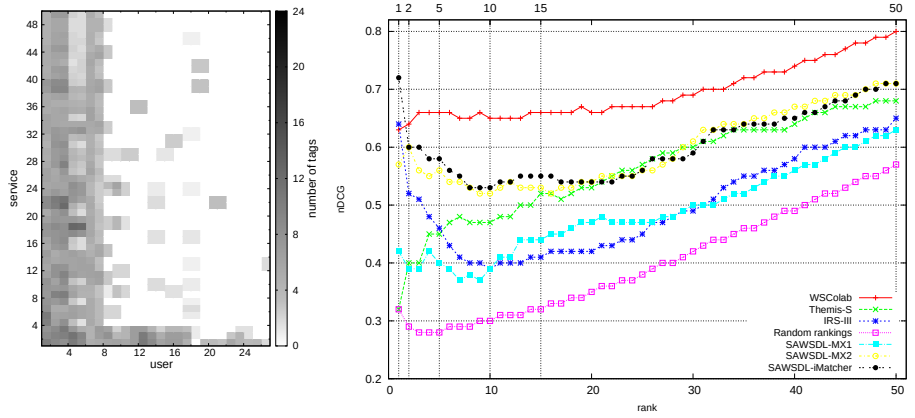
relevant answer in the top of the ranking. We also plot the $nDCG$ curve, where the X-axis represents a rank, and the Y-axis represents a $nDCG$ value for a given rank. An ideal matchmaker has a horizontal curve with a high $nDCG$ value; the vertical distance between the ideal $nDCG$ curve and the actual $nDCG$ 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.

Results All the results reported here are courtesy of the S3 contest organizers. Figure 3b 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. This not only justifies the operating cost of our description method, but also shows that our matchmaker can be competitive to matchmakers working with more formal service descriptions. Detailed results analysis (found in [13]) reveals the possible cause: authors of formal approaches have encoded only information about major features of the service. Encoding minor features (e.g. `miles`) would require spending additional effort on expending existing domain ontologies (Challenge A). Collaborative tagging does not require any additional effort when extending vocabulary and thus minor features have been encoded (Challenge E).

With regard to retrieval efficiency, 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 (Challenge F).

6.3 Assessing Usability Criteria

We asked how well our solution resolved usability-related challenges (C and D). For this goal we analyzed retrieval effectiveness of each query formulating user. Only one of the five users have used query language improperly by constantly putting the keywords related to the input and keywords related to output in the input field. He received the worst retrieval effectiveness (averaged over queries) among all users. This may suggest the we should make search interface more supportive in understanding the query language (Challenge C). With regard to the description transparency (Challenge D), 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 aspect, 26% for the input, and 34% for the output. This shows that most descriptions are transparent for a user. Still, in a few cases performance of users suffered from vocabulary and structure gaps: someone had described a searched service in a different way that the searcher would (e.g. used altitude instead of height, or put information about output in the behavior field).



(a) (b)

Fig. 3. Evaluation results: (a) user contributions in tagging the same services (darker area represents more tags provided by a given user for a given service); (b) the nDCG curves for the six different matchmakers averaged over 4 different graded relevance judgments. Shown courtesy of the S3 contest organizers.

7 Concluding Remarks

To be discoverable a service needs to be described. We asked whether structured collaborative tagging can be a practical approach for describing Web services based only on available WSDL information harvested automatically from the Web. Our work provides several suggestions in this direction. Structured collaborative tagging offers competitive retrieval performance with respect to more expensive (in terms of description) methods based on semantic annotation. It is easy to use and relatively transparent, thus usable for a searcher. However, it does not allow to eliminate operating cost completely. Participation in collaborative tagging is voluntary, and hence certain amount of time and resources must be spent on marketing to attract and to develop a community of developers around the portal. Most importantly, collaborative tagging suffers from the lack of *controllability*. Controlling the annotation process is necessary to get services with insufficient descriptions described but it cannot be reconciled with one of the major motivations of a user to participate: her *autonomy* as to what, when and how to tag [24]. Consequently, many services will not described at all, thus remaining non-discoverable. This suggests that structured collaborative tagging should be implemented by the service broker only as a description method complementary to traditional ones.

Acknowledgement 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, Gior-

gio Villani and Federica Mandreoli (for the discussion). Finally, we would like to thank voluntary participants of our experiment for their enthusiasm.

References

1. Al-Masri, E., Mahmoud, Q.H.: Discovering Web Services in Search Engines. IEEE Internet Computing 12(3) (2008)
2. Bouillet, E., Feblowitz, M., Feng, H., Liu, Z., Ranganathan, A., Riabov, A.: A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition. In: IEEE SCC. pp. 389–396 (2008)
3. Chukmol, U., Benharkat, A.N., Amghar, Y.: Enhancing Web Service Discovery by Using Collaborative Tagging System. In: NWESP. pp. 54–59 (2008)
4. Cuel, R., Oksana, T.: SeekDa Case. Tech. rep., Insemtives Project (2010)
5. Dietze, S., Benn, N., Conconi, J.D., Cattaneo, F.: Two-Fold Semantic Web Service Matchmaking—Applying Ontology Mapping for Service Discovery. In: ASWC (2009)
6. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: VLDB (2004)
7. Falleri, J.R., Azmeh, Z., Huchard, M., Tibermacine, C.: Automatic Tag Identification In Web Service Descriptions. In: WEBIST (2010)
8. Fernández, A., Hayes, C., Loutas, N., Peristeras, V., Polleres, A., Tarabanis, K.A.: Closing the Service Discovery Gap by Collaborative Tagging and Clustering Techniques. In: SMRR (2008)
9. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Commun. ACM 30(11) (1987)
10. Furnas, G.W., Fake, C., von Ahn, L., Schachter, J., Golder, S., Fox, K., Davis, M., Marlow, C., Naaman, M.: Why do tagging systems work? In: CHI (2006)
11. Gawinecki, M.: WSColab Portal. `http://mars.ing.unimo.it/wscolab/` (2009)
12. Gawinecki, M., Cabri, G., Paprzycki, M., Ganzha, M.: Trade-off Between Complexity of Structured Tagging and Effectiveness of Web Service Retrieval. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010 Workshops. pp. 289–300. LNCS 6385, Springer, Heidelberg (2010)
13. Gawinecki, M., Cabri, G., Paprzycki, M., Ganzha, M.: Evaluation of structured collaborative tagging for Web service matchmaking. In: Semantic Services: Advancement through Evaluation (2011), to appear
14. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32(2), 198–208 (2006)
15. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. In: WWW. pp. 211–220 (2007)
16. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. 20(4), 422–446 (2002)
17. Klusch, M.: Semantic web service coordination. In: CASCOM: Intelligent Service Coordination in the Semantic Web, chap. 4, pp. 59–104 (2008)
18. Klusch, M., Kapahnke, P.: Semantic Web Service Selection with SAWSDL-MX. In: SMRR. vol. 416 (2008)
19. Klusch, M., Kapahnke, P., Zinnikus, I.: SAWSDL-MX2: A Machine-Learning Approach for Integrating Semantic Web Service Matchmaking Variants. In: ICWS. pp. 335–342 (2009)

20. Knackstedt, R., Kuropka, D., Müller, O., Polyvyanyy, A.: An Ontology-based Service Discovery Approach for the Provisioning of Product-Service Bundles. In: ECIS (2008)
21. Küster, U.: Jena Geography Dataset. http://fusion.cs.uni-jena.de/professur/jgd (2009)
22. Küster, U.: JGDEval at S3 Contest 2009 - Results. http://fusion.cs.uni-jena.de/professur/jgdeval/jgdeval-at-s3-contest-2009-results (2010)
23. Küster, U., König-Ries, B.: Relevance Judgments for Web Services Retrieval - A Methodology and Test Collection for SWS Discovery Evaluation. In: ECOWS (2009)
24. Kuznetsov, S.: Motivations of contributors to wikipedia. SIGCAS Comput. Soc. 36(2), 1 (2006)
25. Lausen, H., Haselwanter, T.: Finding Web Services. In: ESTC (06 2007)
26. Marlow, C., Naaman, M., Boyd, D., Davis, M.: HT06, tagging paper, taxonomy, Flickr, academic article, to read. In: HYPERTEXT. pp. 31–40. ACM, New York, NY, USA (2006)
27. Mashery: Selling beyond your site: Five key api strategies for retailers, business white paper (2010), http://www.mashery.com
28. Meyer, H., Weske, M.: Light-Weight Semantic Service Annotations through Tagging. In: ICSOC. LNCS, vol. 4294, pp. 465–470 (0 2006)
29. Mili, A., Mili, R., Mittermeir, R.T.: A survey of software reuse libraries. Ann. Softw. Eng. 5, 349–414 (1998)
30. Mili, H., Mili, F., Mili, A.: Reusing Software: Issues and Research Directions. IEEE Trans. Softw. Eng. 21(6), 528–562 (1995)
31. Preece, J.: Sociability and usability in online communities: determining and measuring success. Behaviour & Information Technology 20(5), 347–356 (2001)
32. Semantic Service Selection contest—Summary Report. http://www-ags.dfki.uni-sb.de/~klusch/s3/s3c-2008.pdf (2008)
33. Semantic Service Selection contest. http://www-ags.dfki.uni-sb.de/~klusch/s3/html/2009.html (2009)
34. Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. Inf. Process. Manage. 24(5), 513–523 (1988)
35. Shirky, C.: Ontology is Overrated: Categories, Links, and Tags. http://www.shirky.com/writings/ontology_overrated.html (2005)
36. SOA World Magazine: Microsoft, IBM, SAP To Discontinue UDDI Web Services Registry Effort. http://soa.sys-con.com/node/164624 (2009)
37. United Nations Standard Products and Services Code. http://www.unspsc.org (2009)
38. Vanderlei, T.A., Durao, F.A., Martins, A.C., Garcia, V.C., Almeida, E.S., de L. Meira, S.R.: A cooperative classification mechanism for search and retrieval software components. In: SAC. pp. 866–871. ACM (2007)
39. Wang, Y., Stroulia, E.: Semantic Structure Matching for Assessing Web-Service Similarity. In: ICSOC (2003)
40. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR (2005)
41. Wei, D., Wang, T., Wang, J., Chen, Y.: Extracting semantic constraint from description text for semantic web service discovery. In: ISWC. pp. 146–161 (2008)
42. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38(2), 6 (2006)