Chapter 11
Evaluation of Structured Collaborative Tagging for Web Service Matchmaking

Maciej Gawinecki, Giacomo Cabri, Marcin Paprzycki, and Maria Ganzha

Abstract Turning Web services into Semantic Web Services (SWS) can be prohibitively expensive for large repositories. In such cases collecting community descriptions in forms of structured tags can be a more affordable approach to describe Web services. However, little is understood about how tagging impacts performance of retrieval. There are neither real structured tagging systems for Web services, nor real corpora of structured tags. To start addressing these issues, in our approach, we motivated taggers to tag services in a partially controlled environment. Specifically, taggers were given application requirements and asked to find and tag services that match the requirements. Tags collected in this way were used for Web service matchmaking and evaluated within the framework of the Cross-Evaluation Track of the Third Semantic Service Selection 2009 contest. As part of the lessons learned, we explain relations between description schema (SWS, tags, flat document) and matchmaking heuristics, and performance of retrieval in different search scenarios. We also analyze reliability of tagging system performance as related to taggers'/searchers' autonomy. Finally, we identify threats to results' credibility stemming from partial control of the tags collection process.

M. Gawinecki (✉) · G. Cabri
Department of Information Engineering, University of Modena and Reggio Emilia, Modena, Italy
e-mail: maciej.gawinecki@unimore.it; giacomo.cabri@unimore.it

M. Paprzycki
Warsaw School of Management, Warsaw, Poland

Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
e-mail: marcin.paprzycki@ibspan.pl

M. Ganzha
Institute of Informatics, University of Gdańsk, Gdańsk, Poland

Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
e-mail: maria.ganzha@ibspan.pl

M.B. Blake et al. (eds.), *Semantic Web Services*, DOI 10.1007/978-3-642-28735-0_11,

11.1 Introduction

To build reliable applications in a short time, and for little money, a software engineer should reuse existing artifacts: functionalities, data and resources. Access to such artifacts is often provided through Web services. However, if services are not described with proper metadata, software developers cannot find them. Here, Semantic Web Services (SWS) are to provide service metadata and, thus, make retrieval effective. However, most of SWS research is devoted to effective matchmaking, neglecting costs of turning traditional services into semantic ones. The main obstacles are investment and operating costs, stemming from: (1) complex formalisms requiring skilled experts to annotate, (2) ontologies that are expensive to develop and maintain [7, 20], and (3) the lack of motivating scenarios for the community to participate in the description process [25]. Without eliminating these obstacles SWS may remain unused on large scale, similarly to many formal approaches for traditional software components repositories [14, 15].

Discovery based on community tags for Web services supports software developers, but has only modest requirements [2,4,5,13,23,27]. Software developer without background in SWS can provide metadata. She decides which aspects to capture, in which vocabulary without the need for a predefined ontology. Participation in tagging is embedded in the service search process: by tagging, the programmer organizes her collections of bookmarked services to ease further (re)discovery of services.

Given the promising low costs of implementation and maintenance of tagging (as an approach to getting service metadata), our goal is to try to understand whether it can be competitive when compared to the SWS. In particular, we are interested in answers to the following questions:

- In collaborative tagging users have much autonomy. For instance, taggers decide which service to tag. Can a tagging system provide satisfying retrieval performance for Web services? How much the autonomy impacts performance?
- Search scenarios differ in goals and constraints. For instance, one programmer may search for a very specific functionality, e.g., geocoding; while another may search for services providing *any* information about a given input, e.g., a geographic region. For which search scenario structured tags work better? Why?

Obviously, answering these questions is hard for one major reason. There are neither real structured tagging systems for Web services, nor real corpora of structured tags.

To start addressing these issues, in our approach, users worked in a partially controlled environment. Specifically, taggers were given application requirements and asked to find and tag services that match the requirements. Tags collected in this way were used in the WSColab [5], our Web service matchmaker for structured tags. Performance evaluation was completed within the framework of the Cross-Evaluation Track of the Third Semantic Service Selection 2009 contest [19]. Our approach was compared to SWS matchmakers (using experts annotations), and a matchmaker using automatically generated descriptions.

Given that the tags collection process was partially controlled, our results may *not* be useful for predicting performance of an uncontrolled tagging system. Nevertheless, our results can yield insights into *ideal* performance of tagging approaches and threats to this performance.

In summary, the contributions of this chapter are:

- Identification of factors that need to be controlled during tag and query collection processes (Sect. 11.3).
- Insight into relation between descriptions schema (SWS, tags, flat document) and matchmaking heuristics, and performance of retrieval in different search scenarios (Sect. 11.4.2).
- Analysis of reliability of tagging system performance as related to taggers'/searchers' autonomy (Sect. 11.4.3).
- Identification of threats to results credibility (Sect. 11.4.4). Particularly we compare tagging dynamics between our and real systems (from better studied domains [6, 18]).

11.2 Approach: WSColab

Originally, tagging was used in Web service and software component repositories as a classification schema *complementary* to traditional ones (e.g., faceted-based search) [4, 13, 23, 27].[1] In such form it is useful only to minimize the distance between how a service is described in the repository and how a user describes it [27].

In our approach, WSColab [5], structured tagging *replaces* existing classification schema. In this way it captures primary aspects, important for most of searchers but still allows inclusion of complementary aspects, relevant for individual users. For instance, taggers are encouraged to describe service input, output and behavior (e.g., `distance`), but can also add detailed information, e.g., `miles` as a length unit of the distance. Example of a tagged service is shown in Fig. 11.1.

Our matchmaker for tag descriptions of services supports:

- *Interface/Behavior Matchmaking*: To facilitate matching of services that act similarly but have different interfaces it returns services that are interface compatible, i.e., service input/output tags match at least one input query keyword and one output query keyword (interface matching heuristics) or behavior compatible, i.e., there is a match with a behavior tag (behavior matching heuristics).
- *Approximate search*: Matches are ranked to help a user to assess their relevance. Ranking is generated by combining rankings for each service part (input, output, behavior) that are created using traditional information retrieval techniques adopted for folksonomies.

[1] See also online registries, e.g., SeekDa.com, ProgrammableWeb.com

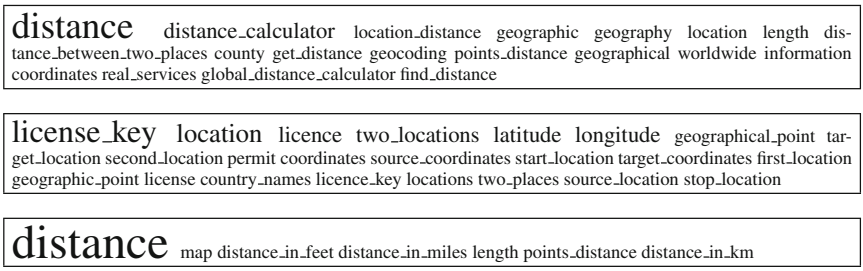

Fig. 11.1 Tag clouds for behavior, input and output of the DOTSGeoCoder_GetDistance service. The bigger the tag the more actors have used it to annotate the service

- *Primary/complementary aspects search*: It scores higher services with matching primary aspects than those with only complementary ones.
- *Incomplete information search*: A user with unclear goal in mind can formulate incomplete queries. Incompletely annotated services can still be matched.
- *Interactive search*: Fast query evaluation algorithm based on (in-memory) inverted indexes [29] enables a system to return services in short time.

11.3 Solution: Using WSColab for Annotation and Retrieval

11.3.1 Matchmaker Alterations

For participation in the contest, an altered version of the matchmaker was used – it returns the ranked matched services, followed by all non-matched services in a random order. This procedure allows results of the proposed matchmaker to be compared to the results of other matchmakers in the competition.

11.3.2 Configuration of Tag Collection Process

Collaborative tagging is a complex dynamic system where users interact with each other *indirectly* by sharing tags about common resources, e.g., a user X tags a service, while a user Y uses the same tags to pre-filter services for her. Next, when Y finds the relevant service, she tags it with her tags, probably based on the tags of X. This is a fragment of a *feedback cycle* [18] that allows semantics of a service to emerge from annotations of different users. The following factors affect the results of this process (we do not claim the list to be complete):

- *Tagging context*. Why a user decides to tag impacts *which services* get tagged and *what aspects* the tags cover [6, 12].
- *Participants*. Motivations to participate vary across different users [12].

- *Tagging system architecture*. What is the user's task that a system aims to support? Who has the right to tag? Which tags are visible? How tagging activity is supported? Are initial tags bootstrapped? [12, 18, 21].
- *Service types*. Different functionalities and data provided by services may be differently described by taggers.
- *Available information*. What does a user know about a service to tag? For instance, users tend not to tag Web services if they do not have any comprehension of them, for instance they have not tried them [3].

The two last factors were constrained by the Jena Geography Dataset [10] provided by the contest organizers. The dataset provides a collection of data-centric services from geography and geocoding domains. The information available about services includes WSDL definitions and initial categories/tags assigned by authors of the test collection (see Fig. 11.2). The authors had also saved the taggers' cognition effort by documenting all missing but relevant information in original WSDL definitions. To ease the process of service understanding we used: (1) names and natural language documentation of service, an operation and its parameters extracted from WSDL, (2) other users' understanding documented in form of tags (see tagging system architecture). The remaining factors have been controlled in the following way.

Tagging context. We have simulated the situation in which users take part in an early phase of software development with services [26]. Each incoming user was given a random use case scenario, describing a particular *applications goal* (why to search and to tag) and *search indications* (what to search for). We assumed that people would tag only services relevant to their specifications and thus we prepared ten different use cases to capture functionalities of all services in the test collection. We have performed tag collection in an open (non-laboratory) environment. We motivated people to tag by providing a form of a *social bookmarking* portal[2] that supports them in browsing and tagging. We encouraged users to focus on specific aspects by highlighting sections from documentation about input, output and behavior and prompting for tags those aspects explicitly in separate fields. The portal was opened for 12 days between September 16 and 27, 2009.

Tagging system architecture. The portal was seeded with automatically generated tags to avoid the *cold-start problem* [16]. *System annotations* were bootstrapped automatically from the service offers included in the JGD dataset. Furthermore, the interaction between taggers was based on tag sharing in two forms. By tag clouds a tagger could see the vocabulary that others used to tag services she or he classified as relevant, irrelevant or left to be yet classified. With tag suggestions a tagger could see the top five tags for a given service (provided by at least two actors, including also the system), i.e., *recommended tags*, and also her/his her *own most popular tags*. By the latter we tried to help a user to utilize the same, consistent vocabulary. Tags bootstrapping and recommendation have been shown in Fig. 11.2.

[2] http://mars.ing.unimo.it/wscolab/new.php

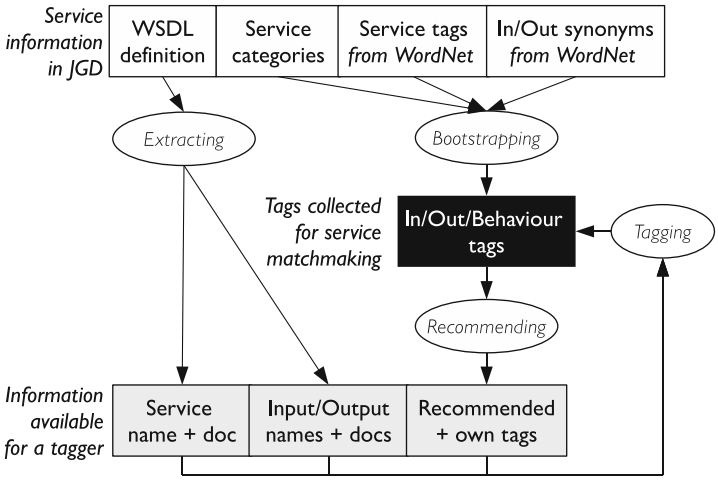

Fig. 11.2 Information flow in tag collection process

Participants. Our user base (27 users) consisted of two groups. A group of our 17 colleagues, with either industrial or academic experience in Web services, SOA or at least some experience in software engineering; and a group of 10 users from the open community, invited through several public forums and Usenet groups concerned with related topics.

11.3.3 Configuration of Query Collection Process

User comprehension in formulating queries, for a specific matchmaker, may cause a bias in performance towards this specific matchmaker. To avoid the bias, queries for the SWS-based solutions have to be formulated by experts in a given query language, while for the collaborative tagging solutions – by a "typical" software developer. Still, the bias towards a particular query formulating strategy may appear, because the query language for structured collaborative tagging is little constrained, and hence, the same service request can be formulated differently depending on a searcher. We addressed this problem on two levels. First, to obtain desirably diverse collection of strategies, we have asked as many users as possible to formulate queries for a given service request. Hence, the performance of our matchmaker could have been averaged over all query formulations. Second, to have control over the vocabulary used, we provided *query formulation assistance*. We have extended our annotation 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. Only the vocabulary, used during tagging phase to describe services, was shared. It was presented as: (1) query suggestions

(through query autocompletion, showing maximally the top 15 commonly used tags for a given service part), and (2) three tag clouds, one for each service part of the annotation.

To assure that searchers were persons that had not tagged service offers previously, the process of query formulation was done in a controlled environment.

11.4 Lessons Learned

11.4.1 Evaluation Results

11.4.1.1 Statistics of Tags and Query Collection Processes

Our tags corpus contains 4,007 annotations for 50 services from the smallest subset of the Jena Geography Dataset (JGD50). Community annotations are 68% (2,716) of all annotations. System annotations (boostrapped automatically, see tagging system architecture in Sect. 11.3.2) are 32% (1,291) of all annotations. Our colleagues (17) have tagged all 50 services, providing 94% (2,541) of community annotations, while the remaining 10 users came from the open community and have tagged only 10 services, providing only 6% (175) of community annotations. The new tags were added by taggers continuously during the collection process (Fig. 11.3a). Taggers were selective which service to tag and to what degree (Fig. 11.3b).

Our corpus contains 45 query formulations for 9 services requests from the JGD50 test collection. Query formulations were provided by five volunteers with background in software engineering or programming. Fortunately, with respect to service requests encodings our corpus has no bias towards a particular query formulation strategy: query formulations differed much in their verbosity and vocabulary. The length of a formulation (averaged over service requests) ranges from 4.2 to 11.2 words depending on the user. Formulations for a single service request shared 50–100% words. System tags, were found somewhat useful by users during the query formulation, but there was no strong bias towards reusing them in the queries. Majority of words used in queries were non-system tags (78%, 74%, 66% for a behaviour, input, output service part, respectively).

11.4.1.2 Contest Evaluation Results

We report effectiveness of Web service retrieval measured as the *normalized Discounted Cumulative Gain* (nDCG) for a result rank i ($nDCG_i$) [8] with respect to the graded relevance judgments [11]. Figure 11.4 shows the nDCG curves for the participating matchmakers. WSColab has a relative performance of 65–80% over most of the ranks (except for the first two ranks) while the remaining systems have

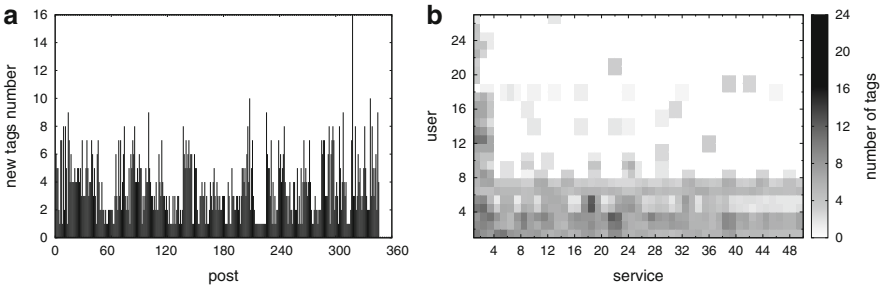

Fig. 11.3 Analysis of service annotation process and its outcome: (**a**) posts differs in the number of new tags added, (**b**) user contributions in tagging the same services (*darker* area represents more tags provided by a given user for a given service)

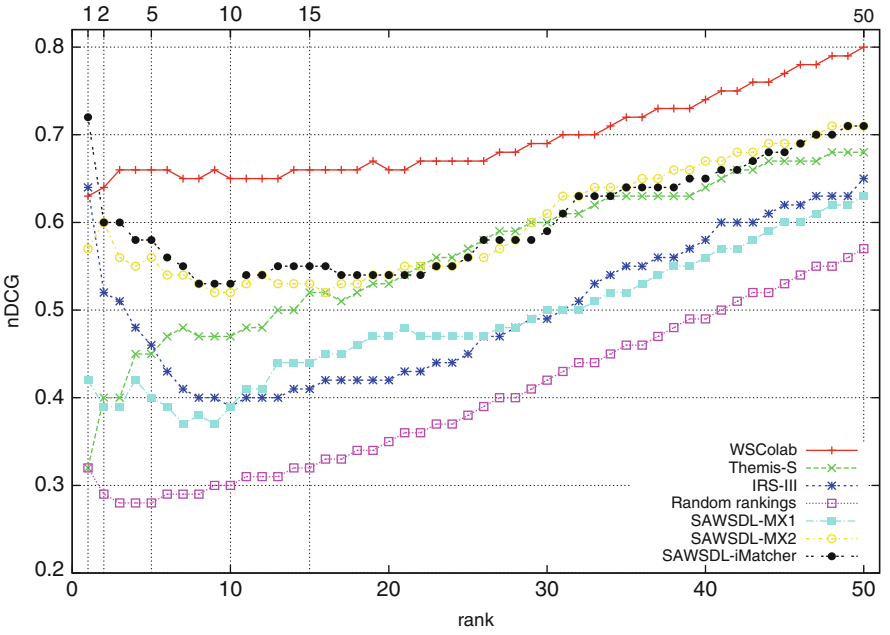

Fig. 11.4 The normalized Discounted Cumulative Gain (nDCG) curves of six different matchmakers for graded relevance judgments with all relevance criteria equally important

a relative performance less than 55–70%. Here, the intuition is that a user needs to spend less effort to find relevant services with WSColab than with other approaches. Figure 11.6 shows the changing effectiveness of retrieval for different searchers formulating queries. Labels U131, U142, ... are used to denote IDs of subsequent searchers. Labels Q5914, Q5835, ... are used to denote IDs of service requests from the JGD test collection. Figure 11.5 reports the effectiveness for different class of approaches: Semantic Web Services (excluding adaptive approaches),

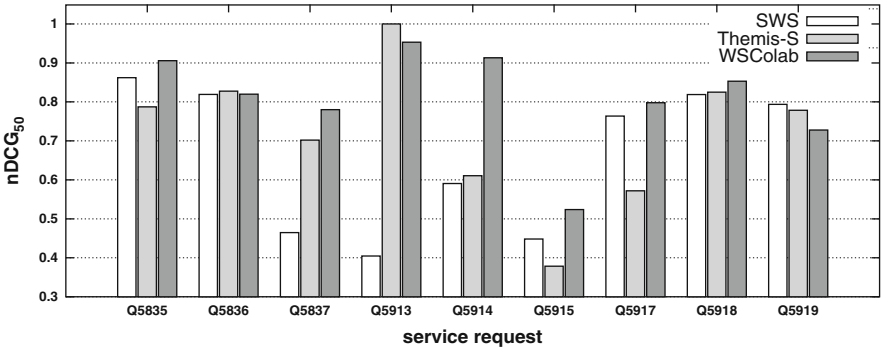

Fig. 11.5 Retrieval effectiveness of three different class of approaches for different service requests. Effectiveness of SWS has been averaged over only non-adaptive matchmakers: SAWSDL-MX1 and IRS-III. Measured with respect to graded relevance judgments with equal dimension settings

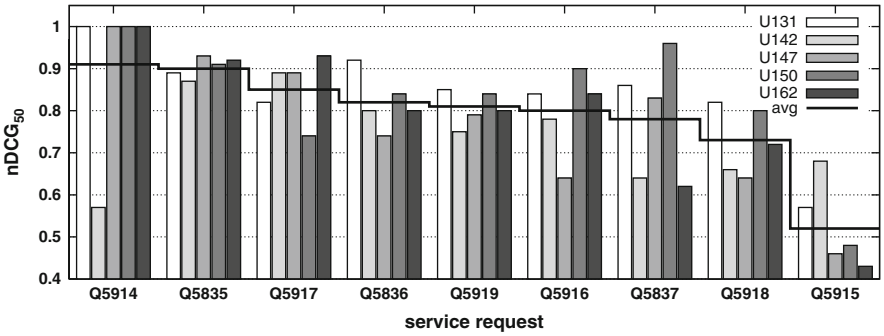

Fig. 11.6 Retrieval effectiveness of five searchers for nine different service requests. Effectiveness averaged over four settings for graded relevance judgments

based on monolithic/flat service descriptions (Themis-S), and collaborative tagging (WSColab) across different requests.

11.4.2 When and Why the Solution Fails and Succeeds?

The retrieval effectiveness of WSColab, on average, is the closest to the effectiveness of an ideal matchmaker (Fig. 11.4) but it varies across particular service requests (Fig. 11.5). For instance, for the Q5914 request the non-adaptive SWS approaches and Themis-S performed significantly worse than WSColab. In this subsection we asked what characteristics WSColab shares with losers and what – with winners, in particular queries.

We considered a solution to be a winner of a given request, if its retrieval effectiveness (with respect to $nDCG_{50}$) was significantly higher than of the others. An ideal winner has good services in the top of the ranking and bad ones in the bottom. Obviously, services can be good (or bad) to a different degree, but to simplify the analysis we consider a service to be either good or bad. This corresponds to binary relevance judgments. We mean a service to be "good" if it at least approximates the required functionality and covers at least part of the scope of required functionality.[3] A solution returns good and bad services in a wrong order, if it has "understanding" (model) of what is good or bad service different than the relevance judgments has. The difference in understanding is caused by two factors: (a) how much of original information about the request and the services has been encoded for a given solution, and (b) how this information is processed by the solution matchmaker. We analyzed both elements for the four requests that differ in losers and winners.

Services with only input or only output matching a request can be bad. Only WSColab could "understand" this. Why? For the Q5914 request only the good services have both input and output matching to the request. Bad services account for 96% of all services in the collection; many of them have only input or only output matching. Hence, many bad services will be included in the ranking, if a matchmaker accepts a match with only input or only output. We call this strategy partial interface matching. Conversely, only good services will be matched, if a matchmaker requires both input and output to be matched. We call this strategy complete interface matching. Both Themis-S and non-adaptive SWS approaches failed in this request probably because they use partial interface matching. WSColab succeeded, because it employs complete interface matching.

All solutions performed comparably when only input (or output) was constrained in the request. Why? The Q5836 request is a counter-example to the previous one. The expected functionality is very generic: it constrains mainly the input, and leaves a wide range of possible values for the expected output. Thereby, the request captures 76% of the services in the collection. Partial interface matching works in this case well, because there is little probability (24%) that a service, that has a similar input, is bad. Hence, Themis-S and non-adaptive SWS approaches performed pretty well (about 80% of the ideal matchmaker performance). Complete interface matching cannot match any service if only input (or only output) is constrained by the request. Therefore, in WSColab, complete interface matching rejected many good candidates.

Even a service having input like the required output can be (approximately) good. Only non-adaptive SWS approaches could not "understand" this. Why? For the Q5837 request some good services only approximates the required functionality. For instance, about 20% of good services do not provide a distance between

[3] It does not need to have interface compatible to the requested one. This corresponds to the binary relevance setting number seven in the contest [19].

two places, but return a list of locations within the specified range. Those services have a required output parameter (distance) as part of the input and one of required input parameter (location) as the output. Hence, those good services will be included in the results, if a matchmaker ignores the difference between input and output. We call this input/output indifferent strategy. Conversely, the same services will be excluded, if a matchmaker differentiates between input and output. We call this input/output aware strategy. The poor performance of SWS approaches can be explained by the use of input/output aware strategy. On the other hand, Themis-S operates on flat documents, i.e., is not aware of interface structure. Similarly, behavior matching heuristics in WSColab does not differentiate between input and output information.

A service having input like the required output can be bad. Only monolithic approaches (Themis-S) could not "understand" this. Why? The Q5918 request is a counter-example to the previous one. The bad services account for 66% of all services in the collection; many of them have output the same as the requested input, e.g., one of them provides a city for a given zip code, not the opposite. Hence, input/output indifference strategy will include many bad services in the results. Themis-S performed worse than other solutions because it implements only this strategy.

Overall, there is no winning strategy for all requests. Rather, the choice of a strategy depends on how strict and specific a searcher's expectation (a request) is, i.e., whether it also accepts approximate services or whether she has no constrains on expected service input (or output). A combination of partial interface matching and input/output indifference can provide more complete results for such relaxed constrains. We call this combined strategy relaxed matching, in opposition to strict matching: complete interface matching with input/output awareness. One must, however, accept, that completeness of relaxed matching is done for the price of precision: also more bad services can be included in the results.

Furthermore, there may be no winning strategy for a given request as summarized in Table 11.1. It shows that winning strategy might be very specific for a given collection of service/request pairs. For instance, having an input similar to the output of required functionality (or vice versa) is not a definite feature of being a good or bad service. Therefore, the winning strategy predicted only on such a test collection-specific feature will work for this collection, but *may* not work for another.[4] For a human searcher, this simply means that she must be more flexible and assess the relevance of returned results herself, and, if she is not satisfied, submit the same query with an alternative matching strategy.

Despite the lack of one universal strategy, WSColab performed relatively stable across different requests. Why? This can be explained by two facts. First, it combines strict matching with relaxed matching. The strict strategy is

[4]This motivated us to exclude adaptive approaches, which learn from relation between such feature and relevance judgments, from this analysis.

Table 11.1 Summary of identified winning strategies

Strategy		Winning strategy when	Used by
Interface matching	Partial	(a) A request constrains only input (or output), and (b) bad services do not have input similar (or output, respectively) to the request.	Themis-S, IRS-III, SAWSDL-MX1, WSColab
	Complete	Good services have both input and output similar, while bad ones – only one of them	WSColab
Input/output	Awareness	Many bad services have some input parameters like some requires output parameters	IRS-III, SAWSDL-MX1, WSColab
	Indifference	Many good services have some input parameters like some required output parameters	Themis-S, WSColab

implemented by the interface matching heuristics. The relaxed strategy – by behavior matching heuristics, because, behavior tags and behavior query keywords were used to describe interface. These heuristics were combined together using a logical alternative, i.e., a service is matched if it satisfies at least one of them. WSColab ranks results of the less precise heuristics lower and thus limits impact of errors on the *nDCG*.

Surprisingly, behavior tags played a different role in matchmaking than we expected. We introduced behavior tags to name service behavior and thus enable matching services acting similar, but having incompatible interfaces. However, users used behavior field to describe service input or output. Only a small fraction of behavior tags describe the behavior as expected,[5] e.g., `converter`, `get_location_info`, `geocode`, `geocoding`, `find`, `location_finder`, `compute_distance`, `find_zip_code`. Since these tags were yet less common in query formulations, many good services were ranked low.

11.4.3 How Reliable Is Structured Collaborative Tagging?

In traditional matchmakers the retrieval effectiveness is the function of test collection (services and requests) and a matchmaker algorithm. Adding human factor to the system makes it more realistic but also makes the function more complex, especially because both taggers and searchers in the system have much autonomy. In this subsection we consider retrieval errors caused by autonomy.

We analyzed in details results for two requests that had the highest standard deviation over all users in the $nDCG_{50}$: Q5914 (19%) and Q5837 (15%), both shown

[5] However, this phenomenon might be characteristic for data-centric services, for which behavior is often identified with the data it consumes and returns.

in Fig. 11.6. We also analyzed in detail the results of query Q5915; most difficult for all matchmakers (see Fig. 11.5).

We found that retrieval errors can be attributed to causes typical for tagging systems [9, 17]: *spam tags*, when a service has been tagged by irrelevant tags (either on purpose or accidentally by a "bad tagger"), and a *vocabulary gap* between the service encoding and the query formulation. Additionally, we have identified problems characteristic for structured tagging [1]: a *structure gap*, when a tagger describes a certain service aspect in a different part of a structure than the searcher does, and an *invalid usage of structure*, when a tagger/searcher use input to describe output or behavior, or output to describe input or behavior. The difference between the last two issues is that the structure gap is caused by vagueness of structure semantics: tags can be assigned to more than one part of a structure. Each tagger has his/her own point of view and it is his/her decision where to assign a tag. Finally, we found that some errors can be attributed to matching only *complementary tags*, which capture detailed information about a service.

Let us now see what retrieval errors those issues caused. In the case of considered data-centric services the community often identifies the behavior of a service with the data it returns. For instance, the searcher U142 could not find service described with a behavior tag `distance_in_miles` because he required this tag to be in the output of a service (structure gap). The same searcher also achieved the worst effectiveness on average ($nDCG_{50}$) for all queries, because he was constantly putting keywords related to the output as the input query keywords (invalid use of a structure), e.g., his formulation for the query Q5837 contained: `input:city_name1 input:distance behavior:distance`. The matchmaker was not able to compute the interface compatibility (both input and output are required); thus, it computed relevance of a service candidate based only on the behavior compatibility. The effectiveness of the U142 was also affected by the vocabulary gap. He looked for a service with the `sea_level_high` behavior, while the `altitude` and the `height`, as semantically related, were very common tags among service encodings. Complementary tags were not a problem themselves: if there was no vocabulary nor structure gap, complementary tags could help a ranking function to discriminate between services that act very similarly. However, in many cases, WSColab ranked bad service highly, because a complementary tags (e.g., `worldwide`) was the only one shared with the query.

We also asked, how significant was the autonomy impact on the effectiveness of retrieval. The impact of vocabulary and structure gap depends on how much similar taggers and searchers describe the same types of services [22]. On average the WSColab solution handled surprisingly well with provided service requests. Figure 11.6 shows that the standard deviation of the retrieval effectiveness among searchers for the same service request was on average only at 4%. Quantifying impact of spam is more difficult. However, the case of the Q5915 shows it can be a serious problem for our matchmaking heuristics. The request asked for services returning map and many bad services have been matched and ranked highly because their behavior or output have been tagged been with the wrong `map` tag. Consequently, effectiveness of all users was affected significantly by the spam.

11.4.4 Can We Trust Evaluation Results?

Threats from partial control of collection process. In real collaborative tagging systems distribution of tags over objects converge to a stable point over time. Thereby, the community achieves consensus on how objects should be described [18]. If such consensus has not been achieved yet, then the description can still change and so the reported retrieval effectiveness is not stable over time. Therefore, we asked whether the community in our simulated scenario achieved such consensus. Figure 11.3a shows it was not the case: new tags were continuously posted by users during the whole collection period. This, as we believe, was for the two reasons. First, the duration of tags collection process was too short; we stopped it after an arbitrary period of time. Second, taggers could not validate discriminatory power of assigned tags, because the feedback cycle (see Sect. 11.3.2) was incomplete. Particularly, the users could not search using provided tags, they could only iterate through services listed in the portal. In real collaborative tagging systems users often minimize the number of tags used to describe an object and they add new tags incrementally – only when existing tags do not allow to differentiate the object from similar ones [18]. Conversely, Fig. 11.3a shows that taggers in the WSColab have been providing large quantities of tags at once.

Threats from matchmaker non-determinism. Both the WSColab and the IRS-III are binary matchmakers, originally returning only a subset of matching services. To be able to compare them with other matchmakers, they were forced to return matching services followed by a randomly ordered "tail" of non-matching services (see Sect. 11.3.1). The random tail introduced non-determinism to their behavior, i.e., the order of services in the tail was determined by the factor external to a matchmaking algorithm. For instance, for WSColab this order was determined by the sequence in which services were indexed. This limited repetitiveness of the evaluation. We asked to what degree this non-determinism affected the reported results of retrieval effectiveness for the WSColab matchmaker. For each configuration (a service request and its formulation) we generated 50 random permutations of services to index. For each configuration we executed the matchmaker and calculated the $nDCG_{50}$. The retrieval effectiveness differed between two different permutations for up to 24%. However, on average (over all configurations) the span of changes to retrieval effectiveness was of 4% with standard deviation of 5%.

11.5 Summary

In this chapter we investigated whether collaborative structured tagging can be a reliable and competitive method to describe Web services (as compared to SWS-based approaches). Understanding pros and cons of this approach may be of great value to find a low cost method to provide metadata in classic Web service repositories. This can be also important for the SWS research, searching

for incentives to attract the community to annotate services in their approach [25]. Analysis of both the contest results and the tagging process has shown that increased autonomy in tagging, and formulating queries, was often the major reason of decreased performance for some users, or for all users for a specific request. Retrieval errors were often caused by: spam, vocabulary and structure gaps. Those issues are typical in open systems and thus well-known instruments exist to resolve them. One must, however, accept that some services will never be described by the community due to the lack of interest in their specific functionalities. In this sense, this makes collaborative tagging different from other annotation approaches that rely on voluntary participation of online users. Unlike tagging, they provide appropriate instruments to control labor distribution [24,28] and thus grant that objects that need to be annotated will be annotated.

Despite those challenges matchmaking based on structured tagging provided very encouraging results. First, retrieval effectiveness of matchmaking was competitive to other approaches, including adaptive ones (like SAWSDL-MX2) that were tuned toward this specific test collection with machine learning. Second, across different scenarios, retrieval performance based on tagged data was more stable than that of other non-adaptive approaches. This, was the effect of combining two strategies: strict matching and relaxed matching. The first differentiates between input and output and accepts only complete interface matching. However, it does not handle well less constrained requests and search for approximate services. Relaxed matching address those limitations, but is also more sensitive to errors. WSColab ranks results of relaxed matching strategies lower. Overall, returned results are more complete, while possibly bad services have smaller impact on effectiveness.

Nevertheless, continued research on tagging-based methods needs a more credible evaluation methodology. For instance, we observed that tagging behavior was not completely realistic and thus its output could not be. This was because a tagger could not verify the "search power" of her and other taggers' tags. To address this issue we plan to allow taggers to be also searchers during the tag collection process. Enabling a user to play a role of both a searcher and a tagger can be also a good instrument to eliminate retrieval problems caused by autonomy. Here, a user will be able to immediately notice the retrieval problems, and fix them with more and better tags. Finally, it will give users an instrument to control the sufficient level of details by which a service has been described.

There is still much to be concluded from the experimental results. We would like to understand whether our approach implements the graded functional relevance criterion correctly. For instance, whether a searcher can find more exact services ranked higher than services that only approximate the required functionality.

Acknowledgements We would like to thank to: Ulrich Küster (for the organization of the Cross-Evaluation track and discussion), Patrick Kapahnke and Matthias Klusch (for their general support and organization of the S3 contest), and to Elton Domnori and anonymous reviewers for valuable comments. Finally, we would like to thank voluntary taggers and searchers for their time.

References

1. J. Bar-Ilan, S. Shoham, A. Idan, Y. Miller, A. Shachak, Structured versus unstructured tagging: a case study. Online Inf. Rev. **32**(5), 635–647 (2008)
2. E. Bouillet, M. Feblowitz, H. Feng, Z. Liu, A. Ranganathan, A. Riabov, A folksonomy-based model of web services for discovery and automatic composition, in *IEE SCC*, Honolulu, 2008, pp. 389–396
3. R. Cuel, O. Morozova, M. Rohde, E. Simperl, K. Siorpaes, O. Tokarchuk, T. Wiedenhöfer, F. Yetim, M. Zamarian, Motivation mechanisms for participation in human-driven semantic content creation. Int. J. Knowl. Eng. Data Min. **1**(4) 331–349 (2011)
4. A. Fernández, C. Hayes, N. Loutas, V. Peristeras, A. Polleres, K.A. Tarabanis, Closing the service discovery gap by collaborative tagging and clustering techniques, in *SMRR*, Karlsruhe, 2008
5. M. Gawinecki, G. Cabri, M. Paprzycki, M. Ganzha, Structured collaborative tagging: is it practical for web service discovery? in *WEBIST*, Valencia, 2010
6. S.A. Golder, B.A. Huberman, Usage patterns of collaborative tagging systems. J. Inf. Sci. **32**(2), 198–208 (2006)
7. A. Heß, N. Kushmerick, Learning to attach semantic metadata to web services, in *ISWC*, Sanibel, 2003, pp. 258–273
8. K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)
9. G. Koutrika, E. Frans, Z. Gyongyi, P. Heymann, H. Garcia-Molina, Combating spam in tagging systems: an evaluation. ACM Trans. Web (TWEB) **2**(4), 1–34 (2007)
10. U. Küster, Jena Geography Dataset (2009), http://fusion.cs.uni-jena.de/professur/jgd, last accessed May 2012
11. U. Küster, B. König-Ries, Relevance judgments for web services retrieval – a methodology and test collection for SWS discovery evaluation, in *ECOWS*, Eindhoven, 2009
12. C. Marlow, M. Naaman, D. Boyd, M. Davis, HT06, tagging paper, taxonomy, Flickr, academic article, to read, in *HYPERTEXT*, Odense, Denmark, 2006, pp. 31–40
13. H. Meyer, M. Weske, Light-weight semantic service annotations through tagging, in *ICSOC*. LNCS, vol. 4294, Chicago, 2006, pp. 465–470
14. H. Mili, F. Mili, A. Mili, Reusing software: issues and research directions. IEEE Trans. Softw. Eng. **21**(6), 528–562 (1995)
15. A. Mili, R. Mili, R.T. Mittermeir, A survey of software reuse libraries. Ann. Softw. Eng. **5**, 349–414 (1998)
16. M. Montaner, B. López, J. De La Rosa, A taxonomy of recommender agents on the internet. Artif. Intell. Rev. **19**(4), 285–330 (2003)
17. I. Peters, K. Weller, Tag gardening for folksonomy enrichment and maintenance. *Webology* **5**(3), 1–18 (2008)
18. V. Robu, H. Halpin, H. Shepherd, Emergence of consensus and shared vocabularies in collaborative tagging systems. ACM Trans. Web **3**(4), 1–34 (2009)
19. Semantic Service Selection Contest (2009), http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html, last accessed May 2012
20. M. Sabou, C. Wroe, C.A. Goble, G. Mishne, Learning domain ontologies for web service descriptions: an experiment in bioinformatics, in *WWW*, Chiba, Japan, 2005, pp. 190–198
21. S. Sen, S.K. Lam, Al.M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F.M. Harper, J. Riedl, Tagging, communities, vocabulary, evolution, in *CSCW*, Banff, 2006, pp. 181–190
22. C. Shirky, Ontology is overrated: categories, links, and tags (2005), http://www.shirky.com/writings/ontology_overrated.html, last accessed May 2012
23. I. Silva-Lepe, R. Subramanian, I. Rouvellou, T. Mikalsen, J. Diament, A. Iyengar, SOAlive service catalog: a simplified approach to describing, discovering and composing situational enterprise services, in *ICSOC*, Sydney, 2008

24. K. Siorpaes, M. Hepp, Games with a purpose for the semantic web. IEEE Intell. Syst. **23**, 50–60 (2008)
25. K. Siorpaes, E. Simperl, Human intelligence in the process of semantic content creation. World Wide Web **13**(1–2), 33–59 (2010)
26. I. Sommerville, *Software Engineering*. International Computer Science Series, 8th edn. (Addison–Wesley, Harlow, 2007)
27. T.A. Vanderlei, F.A. Durão, A.C. Martins, V.C. Garcia, E.S. Almeida, S.R. de L. Meira, A cooperative classification mechanism for search and retrieval software components, in *ACM SAC*, Seoul, 2007, pp. 866–871
28. L. von Ahn, L. Dabbish, Designing games with a purpose. Commun. ACM **51**, 58–67 (2008)
29. J. Zobel, A. Moffat, Inverted files for text search engines. ACM Comput. Surv. **38**(2), 6 (2006)