An approach for Web service discovery based on collaborative structured tagging

Uddam Chukmol, Benharkat Aïcha-Nabila, Amghar Youssef

AN APPROACH FOR WEB SERVICE DISCOVERY BASED ON COLLABORATIVE STRUCTURED TAGGING

Uddam Chukmol, Aïcha-Nabila Benharkat and Youssef Amghar
Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, Villeurbanne, France

Keywords: Web Service Discovery, Structured Tagging, Semantic Annotation, Folksonomy, User-centricity.

Abstract: This research work presents a folksonomic annotation used in a collaborative tagging system to enhance the Web service discovery process. More expressive than traditional tags, this structural tagging method is able to reflect the functional capability of Web services, easy to use and very much accessible to users than ontology or logic based formalism of annotation. We describe a Web service retrieval system exploiting the aforementioned structural tagging. System user profiling is also approached in order to further assign reputation and compute tag recommendation. We present some interesting usage scenarios of this system and also a strategy to evaluate its performance.

1 INTRODUCTION

Web service discovery is a major operation in the development cycle of service-oriented applications. It consists of identifying services having functional capabilities that are relevant to user needs. The increasing number of Web services makes this task even more complicated.

The existing solutions to this problem are often based on information retrieval techniques (IR) which suffers due to poor textual description of Web services or semantic Web technologies. Keyword search applying IR technique seems to be inefficient because of the poor textual description of Web services. Semantic Web technologies, if they are applied within Web service discovery processes, require the extension of original Web service description with ontology-based annotations. The latter hinders not only the service providers but also the service users because the competence in ontology or cognitive engineering is necessary to master the process and to maintain the systems. Besides, very few attempts consider the post-usage information provided by service users to enhance incrementally the process of Web service discovery.

Our proposal is inspired by the user centricity and collaboration found in Web 2.0 environment that aims at generating descriptive data associated to a Web resource (image, document, video, etc.) and the fact that other users exploit this evolving data to share and classify online resources.

Within our research case, users can tag/annotate a Web service that he/she has used with his/her own vocabularies. By analyzing the functional characteristics of Web service through numerous WSDL contents, we propose a specific structural tag/annotation model that takes into account the capability of Web services and their usage domain. We can consider that our tagging approach is based on a folksonomic model that combines the participation of users and the richness of their vocabularies used in tagging process with the structural aspect of annotation met in semantic Web technologies.

Tagging provided by users is not used to extend literally the original content of Web services expressed in WSDL but associated to the invocation URL of the service. We propose consequently a Web service retrieval system that makes use of this data, aggregates them when they are issued from different users and offers a mechanism to search for Web services and rank the output result.

In order to reinforce the quality of discovery and help users in the selection of found Web services, a user profiling scheme is approached and it is served as information that our system further processes to propose user reputation and tag recommendation.

The paper is organized as follows. Section 2 presents the related work. Section 3 describes our system including tagging and query model, tags

aggregation, retrieval algorithm, result ranking and user profiling. Section 4 shows the strategy that we opt for evaluating our system performance and Section 5 concludes the research work and envisions its future perspectives.

2 RELATED WORK

The emergence of collaborative tagging system (Golder & Huberman, 2005) and the abandon of UDDI registry development by its initial creators benefit the increasing number of Internet Web service portals or catalogs (Chen Wu & Chang, 2007), (Al-Masri & Mahmoud, 2008) and facilitate a lot the service suppliers in terms of publication and maintenance. They also allow users to use a single common search interface to look up for different services published by various suppliers.

Traditional information retrieval (IR) or natural language processing (NLP) techniques perform poorly when they are used to solve Web service discovery problems due to insufficient textual descriptions of Web service WSDL contents (Garofalakis, Panagis, Sakkopoulos, & Tsakalidis, 2006). They also do not seem to be scalable enough and offer low search precision rate.

Besides, semantic Web community offers another alternative solution to this discovery problem by proposing several semantic annotation techniques based on ontology or logical formalisms such as WSMO ("Web Service Modeling Ontology"), OWL-S ("OWL-S 1.2 Release"), WSDL-S ("Web Service Semantics - WSDL-S"), SAWSDL ("Semantic Annotations for WSDL and XML Schema") and WSMO-Lite (Kopecky & Vitvar, 2008). They aim at allowing software agents to operate Web service discovery automatically at runtime (on-the-fly) when needed by primarily extending WSDL contents of Web services by semantic annotations. Without automatic semantics annotation process, they may bring more burdens to service suppliers. In effect, on one hand, they have to be competent in cognitive or knowledge engineering in order to master completely the semantics designs and annotation process and on the other hand, they have to choose among the existing aforementioned formalisms the one that suits them best without bringing more cost and effort in training to their service engineers. More effort is also needed to propose semantic-aware service portals or registries in which services can be published and Web service discoveries can be run seamlessly and successfully. Additionally, with multitude semantic annotation formalisms, rare, if not to say, none has proposed such semantic supported registries yet and users of such systems have to be apt in constructing their discovery queries conforming to the imposed semantic annotation formalism from the providers. Despite the theoretical efficiency proved by different semantic Web service discovery proposals, they are struggling at getting widely adopted by real world service industries (Shi, 2007).

Some discovery methods exploit the data offered implicitly or explicitly by users such as preference or feedback on services. Service usage data is collected throughout a framework based on *Implicit Culture* via a deployment of a client-server system and used to improve further Web service discovery in (Birukou, Blanzieri, D'Andrea, Giorgini, & N. Kokash, 2007). At the user side, a client application must be installed to report the way Web services are used to the server. This behavioral information is the processed to compute the similarity between user's request and the set of used and stored services. The system might fail due to the increasing number of users and can be sensitive to confidentiality and privacy policy at client side. Preference of user can be treated as non-functional information to help improve the quality of service discovery, such in (Kovacs, Micsik, & Pallinger, 2007) where Web services and user's requests need to be enriched with specific logic based formalism in order to compute the similarity through a Prolog-based inference engine. (Lamparter, Ankolekar, Studer, & Grimm, 2007) extend original Web service descriptions and discovery queries with semantic data expressed in OWL conforming to their developed preference ontology before proceeding to service-request similarity computing. However, they oblige users to express the queries in logic based or ontology format and finally increase effort, time and investment of service suppliers to extend the descriptions of their published services.

Users are allowed to express their service usage satisfactions as tags (keyword tokens) in (Leitner, Michlmayr, Rosenberg, & Dustdar, 2009). Those tags are used to qualify the non-functional constraints of services. These constraints are then exploited to help in the selection of retrieved and used services. Simpler method is adopted in (Averbakh, Krause, & Skoutas, 2009) by allowing users to rate (as a score) Web services referring to some queries. These ratings are then exploited to enhance the service discovery quality. These approaches seem strongly to be more appropriate to Web service selection rather than discovery because

no functional characteristics of services can be reflected and expressed within their proposals.

(Hagemann, Letz, & Vossen, 2007) recommend users to associate semantics to Web services using tags – those are considered more light-weight and accessible to real world use comparing to semantic annotations in ontology-based formalisms. Such annotation is introduced in (Meyer & Weske, 2006). Then, by taking into account the user collaborations, (Chukmol, Benharkat, & Amghar, 2008) propose a Web service discovery method based on such collaborative and folksonomic annotation.

Nonetheless, most of the latter makes use of *flat* tags without any structure, relation or hierarchy. Thus, it is still vague to consider that such tagging can be used to reflect the functional capability of Web service.

By considering all the drawbacks and advantages met in analyzed related work, we are convinced that through usage experiences, the quality of Web service discovery can be notably improved by exploiting post-experience data provided by users referring to services. This Web 2.0 style of lightweight semantic provision is not costly and can be processed in a bottom-up and minimalistic way. In effect, service suppliers are not forced to invest in such process of semantic annotation and more interesting semantics can be extracted from real user perception on services.

In this research work, we are going to propose a method for Web service discovery based on Web 2.0 collaborative tagging. Our tagging model is easy to use and respects a structure that can reflect the functional capability of services. A simple yet efficient aggregation of tags is also proposed in the overall system.

3 SOLUTION OUTLINE

A number of definitions related to Web service, tag, query and user are proposed in order to facilitate the further comprehension of our proposal.

3.1 Defining Web Service

Functionally, a Web service, through WSDL, can be briefly defined as:

Definition 1. A Web service *w* is a triplet *w<name, doc, ope>,* where *name* is the service name corresponding to the attribute *name* of *<definition>* tag of WSDL, *doc* is the functional description of Web service corresponding to the *<documentation>* tag of WSDL and *ope* is the list of operations of Web service corresponding to *<operation>* tags of WSDL.

Definition 2. An operation *ope* is quadruplet *ope<OName, ODoc, In, Out>,* where *OName* is the operation name corresponding to the attribute *name* of *<operation>* tag, *ODoc* is the description of the operation corresponding also to the *<documentation>* tag under *<operation>*, *In* is the list of input parameters of the operation corresponding to *<input>* tags and *Out* is the list of output parameters of the operation corresponding to *<output>* tags.

Each input and output parameter has a name and a data type.

Our tagging model aims at expressing with the maximal ease of use the aforementioned functional capability of Web services. We allow users to express the contexts of service within their tags also. In effect, it is important to bring forward the usage experience perception of the service and share them with other users. This may lead to more clarity in understanding the situation in which a service is successfully used and can respond at best to the requester's needs.

Through the above definitions, there can be several basic operations or functions a service can offer. Each operation can also be described by users as atomic function offered by the incorporating service. However, if we consider only such operation by itself, it can be used in different situation (e.g. an operation *GetCityWeather* that outputs the weather forecast report given a city name can be invoked in the situation of *"Holiday Preparation"* or "*Wedding party organization"*).

More detail can also be added to make an operation more precise. The operation profile can also described by a set of inputs and outputs.

3.2 Defining Tag and Query

Therefore, according to the aforementioned functional definition of Web service, we can conformingly model our structured tag as:

Definition 3. A structured tag *STag* is a quadruplet *STag<ctxt, funct, input, output>,* where *ctxt* contains the list of keywords separated by **;**, encapsulated in **""** and describing the situation in which the service is / can be used or tested. Its structure is a single dimension table indexed by keywords and each of them has a weight referring to term frequency, *funct* has the same format as *ctxt* but is for describing different operations within the service that can be

used or tested, *input* has the same format as *funct* but is for describing different input parameters that service operations require to execute and *output* has the same format as *input* but is for describing different output parameters that service operations produce.

Thus, when a service is tagged by a user respecting the *STag* structure, it is associated to 4 keyword bags (a.k.a. cloud) - see Figure 1.

Hypothesis 1. A query *q* has the same structure as the tag *STag* defined in Definition 3.

Our annotation or tagging model can be considered as a folksonomy *F* defined by $F \subseteq U \times T \times R$ where *U* refers to users, *R* refers to resource that is Web service description file in our case and *T* refers to tags (conforming to *STag* structure defined in Definition 3) provided by user *U* and associated to resource *R*.

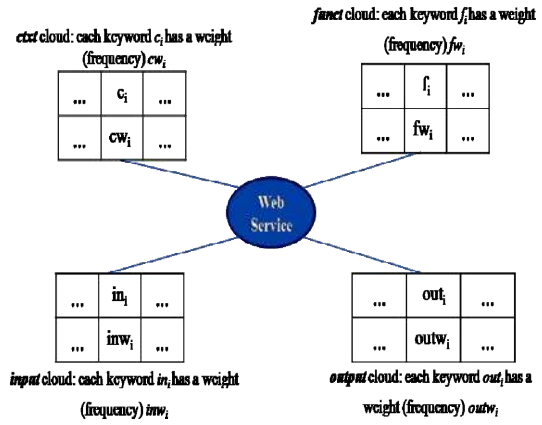

Figure 1: A Web service tagged by *STag*.

Our tagging process is collaborative which means that many users can annotate a Web service by using their own tags. The annotations are visible to other users in the system. Annotated Web services are also accessible to the system users. Therefore, in our system, each Web service is associated to 4 tag clouds and each of them represents accordingly each element of *STag* (context, function, input and output).

3.3 Tag Aggregation Algorithm

When there are several users annotating a Web service, these annotations are aggregated. Each element of *STag* structure is considered respectively and separately. Thus, the 4 tag clouds associated to a service evolves through the tagging activity from users.

The aggregation between two *STag*s can be expressed by the following algorithm:

- Let *STagA* and *STagB* two structured tags (*STag*) provided by two users *A* and *B* to annotate the Web service *ws*.
- Let *STagAgg* the structured tag resulted from the aggregation of *STagA* and *STagB*.

After the aggregation, *STagAgg* is the only tag that is associated to *ws*. We can define STagAgg as follows: `STagAgg=aggregate(STagA, STagB)`.

The *aggregate* function executes 4 merging operations on *ctxt* (respectively *funct, in* and *out*) elements of *STagA* and *STagB*. The *ctxt* (respectively *funct, in* and *out*) element of *STagAgg* is the result of the merging between the *ctxt* (respectively *funct, in* and *out*) elements of *STagA* and *STagB*. We define *merge* as the merging function. Therefore, each element of *STagAgg* can be obtained by:

- *STagAgg.ctxt=merge(STagA.ctxt,STagB.ctxt)*
- *STagAgg.funct=merge(STagA.funct, STagB.funct)*
- *STagAgg.input=merge(STagA.input, STagB.input)*
- *STagAgg.output=merge(STagA.output, STagB.output)*

We detail only the merging of *ctxt* elements of *STagA* and *STagB* because it is applied exactly to other elements.

We would like to recall that *ctxt* is a single dimension table indexed by keywords and each of them has a weight (see Definition 3). Suppose that c_i is a keyword and an index of *ctxt* and cw_i the weight of c_i; then $ctxt[c_i]=cw_i$. Therefore, merging *STagA.ctxt* and *STagB.ctxt* can be described by:

```
merge(STagA.ctxt,STagB.ctxt) {
      //initialize STagAgg.ctxt with
      //STagA.ctxt
STagAgg.ctxt.Init(STagA.ctxt);
Foreach c of STagB.ctxt {
      If c not in STagAgg.ctxt {
            STagAgg.ctxt.add(c);
            STagAgg.ctxt[c]=0;
      }
  STagAgg.ctxt[c]+=STagB.ctxt[c];
   }

}
```

3.4 User, Tag and Web Service Relation

We illustrate the different relations between each element (user, tag and Web service) of our folksonomy model with the goal to further exploit them in computing of similarity between a discovery request and the corpus of our tagged Web services.

The relation between Web services and tags or annotations in our system is the space of 4 two-dimension tables indexed by keywords and Web service identifier (access path to its WSDL file). Each table corresponds to every element or part of the *STag* structure. We can model them as:

- $TabCtxt_{WS}(Ctxt \times WS)$ where *Ctxt* is the list of keywords provided by users to annotate the contextual part of a structured tag associated to a service and *WS* is the list of annotated Web services. Given a keyword $c_i \in Ctxt$ and a Web service $ws_j \in WS$, $TabCtxt_{WS}[c_i, ws_j]$ is the frequency of the term c_i that is used to annotate the usage context of ws_j.
- $TabFunct_{WS}(Funct \times WS)$ where *Funct* is the list of keywords provided by users to annotate the operation part of a structured tag and *WS* is the list of annotated Web services. Given a keyword $f_i \in Funct$ and a Web service $ws_j \in WS$, $TabFunct_{WS}[f_i, ws_j]$ is the frequency of the term f_i that is used to annotate the operations of ws_j.
- $TabInput_{WS}(Input \times WS)$ where *Input* is the list of keywords provided by users to annotate the input part of a structured tag and *WS* is the list of annotated Web services. Given a keyword $in_i \in Input$ and a service $ws_j \in WS$, $TabInput_{WS}[in_i, ws_j]$ is the frequency of the term in_i that is used to annotate the input parameters of ws_j.
- $TabOutput_{WS}(Ouput, WS)$ where $out_i \in Output$ is the list of keywords provided by users to annotate the output part of a structured tag and *WS* is the list of annotated Web services. Given a keyword $out_i \in Output$ and a service $ws_j \in WS$, $TabOutput_{WS}[out_i, ws_j]$ is the frequency of the term out_i that is used to annotate the input parameters of ws_j.

The relation between users and tags is also a space of 4 two-dimension tables indexed by keywords and user identifiers because it refers to services tagged by users. We define the 4 tables as:

- $TabCtxt_{USER}(Ctxt \times User)$ where *Ctxt* is the list of keywords provided by users to annotate the contextual part of the structured tags associated to Web services and *User* is the list of system annotating users. Given a keyword $c_i \in Ctxt$ and a user $u_j \in User$, $TabCtxt_{USER}[c_i, u_j]$ is the frequency of the term c_i that is used by the user u_j to annotate the usage context of the system Web services.
- $TabFunct_{USER}(Funct \times User)$ where $f_i \in Funct$ is the list of keywords provided by users to annotate the operation part of the structured tags associated to Web services and *User* is the list of system annotating users. Given a keyword $f_i \in Funct$ and a user $u_j \in User$, $TabFunct_{USER}[f_i, u_j]$ is the frequency of the term f_i that is used by the user u_j to annotate the operation part of the system Web services.
- $TabInput_{USER}(Input \times User)$ where $in_i \in Input$ is the list of keywords provided by users to annotate the input part of the structured tags associated to Web services and *User* is the list of system annotating users. Given a keyword $in_i \in Input$ and a user $u_j \in User$, $TabInput_{USER}[in_i, u_j]$ is the frequency of the term in_i that is used by the user u_j to annotate the input parameters of the system Web services.
- $TabOutput_{USER}(Ouput \times User)$ where *Output* is the list of keywords provided by users to annotate the output part of the structured tags associated to Web services and *User* is the list of system annotating users. Given a keyword $out_i \in Output$ and a user $u_j \in User$, $TabOutput_{USER}[out_i, u_j]$ is the frequency of the term out_i that is used by the user u_j to annotate the input parameters of the system Web services.

The relation between users and services is the simplest because it can be modeled as a two-dimension table: $TabUser_{WS}(User \times WS)$ where *User* is the list of system annotating users and *WS* is the list of annotated Web services. Given a user $u_i \in User$ and a Web service $ws_j \in WS$, $TabUser_{WS}[u_i, ws_j]$ is the frequency number of how often the user u_i annotates the service ws_j.

However, based on the Hypothesis 1, a discovery request *Q* issued by user can be modeled as a space of 4 single dimension tables indexed by keywords such as:

- $TabCtxt(Ctxt)$ where *Ctxt* is the list of keywords provided by the requestor as part of his query on the contextual element of tagged Web service corpus. Given a keyword $c_i \in Ctxt$, $TabCtxt[c_i]=cw_i$ is the frequency of the keyword c_i in the table (or vector).
- $TabFunct(Funct)$ where *Funct* is the list of keywords provided by the requestor as part of his query on the operation element of tagged Web service corpus. Given a keyword $f_i \in Funct$,

$TabFunct[f_i]=fw_i$ is the frequency of the keyword f_i in the table.

- *TabInput(Input)* where *Input* is the list of keywords provided the requester as part of his query on the input element of tagged Web service corpus. Given a keyword $in_i \in Input$, $TabInput[in_i]=inw_i$ is the frequency of the keyword in_i in the table.
- *TabOutput(Output)* where Output is the list of keywords provided by requestor as part of his query on the output element of tagged Web service corpus. Given a keyword $out \in Output$, $TabOutput[out_i]=outw_i$ is the frequency of the keyword out_i in the table.

We are going to discus in more detail about how to discover Web services using collaborative structured tags.

3.5 Web Service Discovery based on Collaborative Structured Tagging

Let Q a discovery request expressed by a user and $aWS \in WS$ an annotated Web service in the tagged Web Service collection denoted by *WS*.

aWS is discovered by Q if and only if there is a similarity degree between Q and its structured annotation $\in [0, 1]$.

Therefore a Web service *aWS* is represented in this similarity computing by its structured annotation denoted as $STag_{aWS}$.

According to Hypothesis 1 and section 3.4, we can present Q and $STag_{WS}$ as follows:

- $Q <TabCtxt, TabFunct, TabInput, TabOutput>$
- $STag_{aWS} < TabCtxt_{WS}[Ctxt, aWS], TabFunct_{WS}[Funct, aWS], TabInput_{WS}[Input,aWS], TabOutput_{WS}[Output, aWS] >$

Each element of $STag_{aWS}$ is a single dimension table because it is a two-dimension table with a unique column indexed by *aWS*.

The similarity of a Web service *aWS* (represented by $STag_{aWS}$) and the query Q is the weighted sum of four basic similarities:

- $simCtxt(TabCtxt,TabCtxt_{WS}[Ctxt, aWS])$: the context similarity between Q and $STag_{aWS}$.
- $simFunct(TabFunct,TabFunct_{WS}[Funct, aWS])$: the operation similarity between Q and $STag_{aWS}$.
- $simInput(TabInput,TabInput_{WS}[Input,aWS])$: the input similarity between Q and $STag_{aWS}$.
- $simOutput(TabOutput,TabOutput_{WS}[Output, aWS])$: the output similarity between Q and $STag_{aWS}$.

Each basic similarity value is $\in [0,1]$. We use 4 coefficients associated to each similarity and their sum is 1: $coeff\_ctxt \in [0,1]$ is the coefficient for usage context similarity, $coeff\_funct \in [0,1]$ is the coefficient for operation similarity, $coeff\_input \in [0,1]$ is the coefficient for input similarity and $coeff\_output \in [0,1]$ is the coefficient for output similarity. Thus, the final similarity is computed by:

$$similarity = coeff\_ctxt*simCtxt + coeff\_funct*simFunct + coeff\_input*simInput + coeff\_output*simOutput; \quad (1)$$

According to the previous modeling of $STag_{aWS}$ and Q, each basic similarity can be computed by using the vector space similarity. Some analytic studies recommend using *term frequency* as the weight of each keyword representing the context (respectively operation, input and output part) of $STag_{aWS}$. In effect, keyword distribution in collaborative tagging systems often respects the Zipf's power law (Halpin, Robu, & Shepherd, 2007)(Robu, Halpin, & Shepherd, 2009) where a few representative tags cover most of the distribution. We grant more importance to most frequently used tags to represent the tagged resource.

Thus, the basic similarity between the context (respectively operation, input and output) part of the $STag_{aWS}$ and the context (respectively operation, input and output) part of the query Q is:

```
Float simCtxt(Q.TabCtxt,
STag_aWS.TabCtxt_WS[Ctxt, aWS]) {
   return cosine(Q.TabCtxt,
   STag_aWS.TabCtxt_ws,aWS];
}
```

The detail of the *cosine* measure of similarity between two term vectors can be consulted in (D. Manning, Raghavan, & Schülze, 2008) and (Markines et al., 2009).

In order to illustrate this basic similarity computing, we propose a case example as follows:

Let:

- $V_{WS}=STag_{aWS}.TabCtxt[Ctxt,aWS]$: the context part vector of the structured annotation associated to the service *aWS*.

Keyword	Frequency
Auto	5
Mobile	10
Transport	15
Vehicle	3

- $V_Q=Q.TabCtxt$: the context part vector of the discovery query Q

Keyword	Frequency
Transport	1
Vehicle	1

The cosine measure of similarity between V_Q and V_{WS} can be computed as:

$$\text{cosine}(V_Q,V_{WS})=\frac{5\times0+10\times0+15\times1+3\times1}{\sqrt{5^2+10^2+15^2+3^2}\times\sqrt{1^2+1^2}}$$

$$\text{cosine}(V_Q,V_{WS})=0.95.$$

3.6 Result Set Filtering

The result set of the discovery is ranked by the overall degree of similarity (see (1)). Users are allowed to filter this result set in order to keep the Web services that interest them only.

We propose two types of filtering: by threshold and by user implicit profile.

3.6.1 Filtering by Threshold

This is the simplest yet efficient way of filtering a retrieval or discovery result. It consists of eliminating any discovered Web service in the result set whose global similarity value given the requestor's query is below a limit value *threshold*. This latter can be defined by user him/herself or proposed by the discovery system.

3.6.2 Filtering by User Profile

A user signs up to the system by providing his/her personal information (name, address, email, signature, etc.) and his/her working domain or expertise domain. This latter can be described using a list of keywords (e.g. "networking"; "databases"; "administration").

While the user's personal information is mainly used by the system to authenticate a user, keyword set related to expertise domain is evolved according the way user employ the system (i.e. how often a user tags? What tags does a user add? How many Web services a user tags?, etc.)

The implicit user profile in our study case has the same structure as the annotations of Web service (see Definition 3). Therefore, a user profile is just a structured annotation that is generated implicitly and is changed throughout time and the tagging habit or behavior of the user.

We opt for simple method for generating such profile by taking into account the top *k* keywords used regularly by the user to tag Web services. *k* is an integer value that is computed automatically by the system and it is variable for different part of profile information (i. e. usage context, operation, input and output). In effect, an active user that tags more often different Web services with the converging set of vocabularies can be described or profiled by those frequent terms.

We discuss in more detail how *k* related to *usage context* part of the profile is computed and how this part of profile is generated as follows.

Assume that the profile of a user *u* is defined by $ImProf(u)=\langle TabCtxt[Ctxt, u], TabFunct[Funct,u], TabInput[Input, u], TabOutput[Output, u]\rangle$ where:

- $TabCtxt[Ctxt, u]$ is the usage context vector associated to the profile of user *u*, in which *Ctxt* is the list of keywords used to describe the usage context part of the profile and $TabCtxt[c_i,u]$ is the frequency of the keyword $c_i \in Ctxt$ used by the user *u* in his/her overall tagging activities on the usage context part of Web service collection.
- $TabFunct[Funct, u]$ is the operation vector associated to the profile of user *u*, in which *Funct* is the list of keywords used to describe the operation part of the profile and $TabFunct[f_i,u]$ is the frequency of the keyword $f_i \in Funct$ used by the user *u* in his/her overall tagging activities on the operation part of Web service collection.
- $TabInput[Input, u]$ is the input vector associated to the profile of user *u*, in which *Input* is the list of keywords used to describe the input part of the profile and $TabInput[in_i, u]$ is the frequency of the keyword $in_i \in Input$ used by the user *u* in his/her overall tagging activities on the input part of Web service collection.
- $TabOutput[Output, u]$ is the output vector associated to the profile of user *u*, in which *Output* is the list of keywords used to describe the output part of the profile and $TabOutput[out_i, u]$ is the frequency of the keyword $out_i \in Input$ used by the user *u* in his/her overall tagging activities on the output part of Web service collection.

We are going to illustrate only how to compute the user context part of the profile for the user *u*. The other parts of the profile are generated by the same way.

```
P=ImProf(u);
setEmpty(P.TabCtxt);
use TabCtxt_WS[Ctxt,WS];
use TabCtxt_USER[Ctxt,User];
k=0;
foreach c_i in Ctxt {
      foreach ws_j in WS {
            k+=TabCtxt_WS[c_i,ws_j];
      }
      k=k div length(WS);
```

```
if (P.TabCtxt_USER[c_i,u]>=k) {
P.TabCtxt[ci,u]=TabCtxtUSER[ci,u] ;}
```

After generating the profile, we consider it as a filtering query that is used with the result set of discovered Web services annotated by structured tags. The same similarity computing in (1) is used to calculate the distance between the requestor's profile and the set of discovered Web services.

3.7 Initiation of Discovery System

When the proposed discovery system is at its initial state, there are two particular tasks that need to be done automatically:

3.7.1 Creating Structured Tags Automatically

We can parse the WSDL content of each service and extract only the WSDL tags (e.g. <documentation>, <operation>, etc.) or their attributes that corresponds to each part of our *STag* structure. For the primary version of this proposal, we adapt the string tokenization method of (Natallia Kokash, 2006) to create keywords lists for different parts of the structured tagging associated to the corresponding service.

The initial structured annotation is evolved when the tagging activities of users increase.

3.7.2 Creating Initial User's Implicit Profile

When a user is signed up, except from prompting him from offering his personal information (name, address, mail, signature, etc.), a list of keywords describing the working or expertise domain of the user is required. These keywords are affected directly to the usage context part of the user's implicit profile because they are susceptible to describe best the situations in which the user can employ the services.

User is recommended to have a digital identifier to facilitate and reinforce his/her authentication with the system. Digital signature based on OpenID ("OpenID Foundation") seems strongly to be an interesting candidate model that we further take into account.

To partially sum up, the proposed system suits best semi-automatic and user-centric or user-assisted discovery of Web service where users can participate and collaborate with each other to make public knowledge on the usage of Web services emerge and exploitable to further enhance the discovery quality. It is recommended that users test or use at least once a service Web before annotating it with our structured tagging. By doing so, he/she has a clearer idea and perception of what feed back or annotation he/she should attach to the Web service.

4 APPLICATIVE SCENARIOS OF THE DISCOVERY SYSTEM

Web Service Discovery at Design Time. Service oriented application engineer can use our system to assist his/her task is finding services that are relevant to the functional requirement of clients. Such system allows engineers to browse different services in the enterprise collection by simple tags and formulating easily his/her request by just following a simple guide. Usage experience of services can be shared among engineer community to improve the reuse of locally available Web services, if those engineers care enough participating in the tagging activities after testing or using the services.

User-assisted Web Service Composition. When a user would like to have on-time effects or intervention on the composition of Web services, he/she can check the tag clouds at his/her disposal for atomic service that responds to his/her prompt need in order to create his composite service. The need formulation is easy and our tagging model favors the annotations on input/output of Web services which are habitually useful for the composition process.

User-oriented and Semi-automatic Web Services Clustering or Classification. With the increasing number of tags on functionality of services provided by users, the expression of actual usage of services can be deduced. By employing efficient text or data mining algorithms, the clustering or classification of such annotated service collection are more than feasible and can bring interesting outcome.

A prototype of our system is under development and we have proposed a strategic evaluation of this system that is discussed in the next section.

5 SYSTEM EVALUATION METHOD

The performance of our discovery system can be evaluated from several points of view, notably *(i)* ***Tags***: tag quantity can be the first parameter to let us know more about the quality of discovered services.

The evolution of tag quantity needs to be followed up for comparing the evolution of system performance *(ii)* ***Users***: user participation is to be considered in terms of tagging provision and frequency. This participation can influence the tag quantity and quality also. We would like to study the evolution of implicit user profile and determine if this profile can be stable through continuing tagging activities. It is also important to verify if the way we recommend using user's implicit profile can improve the discovery result set *(iii)* ***Basic Similarity Coefficients*****:** our system computes the overall similarity between a query and an annotated Web service based on 4 coefficients. A tuning mechanism of these coefficients is important so that at the initial state of the system, coefficients by default are suitably proposed to users *(iv)* ***Precision/Recall rate*****:** High precision and low recall rates signify good performance of the system. *(v)* ***Term Weight:*** term frequency is used as weight in our case but there are also many more ways to compute term weight such as shown in (D. Manning et al., 2008) and (Markines et al., 2009). We would like to define practically the best term weight computing that can yield the best precision/recall rate of discovery *(vi)* ***Similarity Measure*****:** *cosine* similarity measure is used in our system because in general case, this measure outperforms other measures. However, (Markines et al., 2009) recommends other measures also such as *Pearson*, *Dice* or *Jaccard*. We would like to verify if these three latter can outperform the *cosine* measure.

These are the specification that we follow to evaluate the performance of our under-development prototype. We particularly accentuate our effort in examining the precision/recall rates because they are obvious indicators of a retrieval system.

6 CONCLUSIONS AND FUTURE WORKS

In this research work, we have described a structured tagging method of Web service that covers and expresses the functional capability and the usage context of the service easily.

A Web service discovery system based on collaborative structured tagging is proposed to exploit the aforementioned tags to enhance the Web service discovery process. Tags issued from different users on a Web service are aggregated according the *bag-of-word* model. We also propose a query model that takes the exact form as a tag for users in order to search for Web services relevant to their need.

The similarity of a query and an annotated Web service is computed based on four basic similarities: usage context, operation, input and output similarity. Each one of them corresponds to a composite part of our proposed tag model. This kind of query and annotation is expressive enough and very easy to construct and use in the task of discovering or tagging the Web services. The proposed basic similarities reflect the degree of relatedness between a query and the functional capability of a Web service, in case that the service is annotated. Finally, the overall similarity degree between a query and an annotated Web service is calculated a weighted sum of the four basic similarities mentioned above.

We use four coefficients respectively for computing the overall similarity because they can allow users to tune the result sets of their discovery request and give them more flexibility to improve the quality of their discoveries by just according more importance to a specific part describing the functional capability of services, i. e. usage context, operation, input or output.

The discovered services are ranked according to their similarity values down from highest to lowest values. Users can, however, filter this result set with a defined or pre-defined *threshold* value to eliminate the non-interesting, irrelevant or ignorable discovery candidates. We offer another method of filtering based on *implicit user profile*. The latter is gradually built and evolved through the tagging activities of the user. Finally this profile gives more important to a user according to his/her tagging frequency, tag quantity and annotated Web service number. The *implicit user profile* takes the form a refining query. The filtering processing based on this profile is no other operation than computing the similarity between the user profile and the set of already discovered services. We apply the same discovery algorithm in both cases: overall similarity computing between initial discovery query and annotated Web services and filtering based on *user implicit profile* and discovered service candidates.

Then we propose some applicative scenarios in which our approach can be employed and recommend a strategic evaluation on our under-development prototype.

Despite the ongoing work on this discovery engine, we envisage some future extensions of the system and future directions of the research work already such as *(i)Treating free text (document) respecting minimally our tagging model as the tagging data (ii)Tag disambiguation (iii) Alternative*

discovery algorithm (iv) User profile and reputation (v)Result ranking and *(vi) Lowering ontology to tags*.

Based on our heuristic proposal, it seems strongly plausible that Web service semantic annotation become a much easier task for service suppliers or requestors, the service discovery quality can be enhanced based on this kind of annotation. Nonetheless, users are strongly encouraged to participate in the usage or testing of services before he/she tags them with our proposed tagging model. Participative data provided by users are crucial source for our approach to process the degree of semantic correspondence between a request and an annotated Web service and for building up user profile that is used in the result set filtering task.

It is finally very possible to combine our approach with the exploitation of other semantic annotations to propose a better Web service discovery system.

REFERENCES

Al-Masri, E., & Mahmoud, O. (2008). Discovering Web Services in Search Engines. *Internet Computing, IEEE, 12(3), 74-77.*

Averbakh, A., Krause, D., & Skoutas, D. (2009). Exploiting User Feedback to Improve Semantic Web Service Discovery. *In The Semantic Web - ISWC 2009 (pp. 33-48).*

Birukou, A., Blanzieri, E., D'Andrea, V., Giorgini, P., & Kokash, N. (2007). Improving Web Service Discovery with Usage Data. *Software, IEEE*, 24(6), 47-54.

Chen Wu, & Chang, E. (2007). Searching Services "on the Web": A Public Web Services Discovery Approach. In the proceedings of Third International IEEE *Conference on Signal-Image Technologies and Internet-Based System*, 2007. SITIS '07. (pp. 321-328).

Chukmol, U., Benharkat, A., & Amghar, Y. (2008). Enhancing Web Service Discovery by Using Collaborative Tagging System. In *4th International Conference on Next Generation Web Services Practices,* 2008. NWESP '08. (pp. 54-59).

D. Manning, C., Raghavan, P., & Schülze, H. (2008). Introduction to Information Retrieval (*Cambridge University Press.*).

Garofalakis, J., Panagis, Y., Sakkopoulos, E., & Tsakalidis, A. (2006). Contemporary Web service discovery mechanisms. *Journal of Web Engineering*, Volume 5(Issue 3), 265-290.

Golder, S., & Huberman, B. (2005). T*he Structure of Collaborative Tagging Systems*. Retrieved February 18, 2010, from http://arxiv.org/abs/cs.DL/0508082.

Hagemann, S., Letz, C., & Vossen, G. (2007). Web Service Discovery – Reality Check 2.0. Eds.: Becker, J. et al. Münster. Working paper, *European Research Center for Information Systems.*

Halpin, H., Robu, V., & Shepherd, H. (2007). The complex dynamics of collaborative tagging. In Proceedings of the *16th international conference on World Wide Web* (pp. 211-220). Banff, Alberta, Canada.

Kokash, N. (2006). A Comparison of Web Service Interface Similarity Measures. In Proceeding of the 2006 conference on STAIRS 2006: *Proceedings of the Third Starting AI Researchers' Symposium* (pp. 220-231).

Kopecky, J., & Vitvar, T. (2008). WSMO-Lite: Lowering the Semantic Web Services Barrier with Modular and Light-Weight Annotations. In *IEEE International Conference on Semantic Computing*, 2008 (pp. 238-244).

Kovacs, L., Micsik, A., & Pallinger, P. (2007). Handling User Preferences and Added Value in Discovery of Semantic Web Services. In I*EEE International Conference on Web Services*, 2007. ICWS 2007. (pp. 225-232).

Lamparter, S., Ankolekar, A., Studer, R., & Grimm, S. (2007). Preference-based selection of highly configurable web services. In *Proceedings of the 16th international conference on World Wide Web* (pp. 1013-1022). Banff, Alberta, Canada.

Leitner, P., Michlmayr, A., Rosenberg, F., & Dustdar, S. (2009). Selecting Web Services Based on Past User Experiences. In *Proceedings of the IEEE Asia-Pasific Services Computing Conference 2009*. Singapore.

Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., & Stumme, G. (2009). Evaluating similarity measures for emergent semantics of social tagging. In *Proceedings of the 18th international conference on World Wide Web* (pp. 641-650). Madrid, Spain.

Meyer, H., & Weske, M. (2006). Light-Weight Semantic Service Annotations Through Tagging. In *Service-Oriented Computing – ICSOC 2006* (pp. 465-470).

OpenID Foundation website . Retrieved February 22, 2010, from http://openid.net/.

OWL-S 1.2 Release. Retrieved February 18, 2010, from *http://www.ai.sri.com/daml/services/owl-s/1.2/.*

Robu, V., Halpin, H., & Shepherd, H. (2009). Emergence of consensus and shared vocabularies in collaborative tagging systems. *ACM Trans. Web, 3(4), 1-34.*

Semantic Annotations for WSDL and XML Schema. Retrieved February 18, 2010, from *http://www.w3.org/TR/sawsdl/.*

Shi, X. (2007). Semantic Web Services: An Unfulfilled Promise. IT Professional, IEEE, 9(4), 42-45.

Web Service Modeling Ontology. Retrieved February 18, 2010, from *http://www.wsmo.org/index.html.*

Web Service Semantics- WSDL-S. Retrieved February 18, 2010, from *http://www.w3.org/Submission/WSDL-S/.*