A Ranking Method for Social-Annotation-Based Service Discovery

Duo Qu, Xudong Liu, Hailong Sun, Zicheng Huang
School of Computer Science and Engineering
Beihang University
Beijing, China
{quduo, liuxd, sunhl, huangzc}@act.buaa.edu.cn

***Abstract*—With the rapid growth of Web services, service discovery becomes an important and difficult issue. Traditional UDDI-based and WSDL-based methods of service discovery have low precision, and semantic-based service discovery methods are usually inefficient and time-consuming. We observe that social annotations can optimize both precision and efficiency of service discovery. In this paper, we propose a social-annotation-based service discovery method by using a learning to rank method, and propose two algorithms, Query Annotation Relevance (QAR) and Service Annotation Ranking (SAR), to calculate the dynamic Query-dependent feature and the static Query-independent feature respectively. Our experiments show that our method is effective for improving service discovery performance.**

Keywords- Web service; service discovery; social annotation; tag

I. INTRODUCTION

Service discovery is of great importance in the Service Oriented Architecture (SOA). With the rapid growth of Web services, how to discover services from a large number of services precisely and efficiently becomes a challenging issue.

Traditional UDDI-based and WSDL-based[2] methods of service discovery are using keyword matching or other existing IR techniques between queries and services descriptions. But UDDI and WSDL files usually contain insufficient information, which makes traditional methods have low precision. In light of the above problem, semantic-based service discovery is widely studied[3-6]. These studies which mainly based on semantic Web and ontology language can get better precision. However, the cost of building and maintaining a semantic-based service discovery system is relatively high, and the process is inefficient and time-consuming due to manually ontology building and complicated reasoning.

Tagging is a popular social annotation methods coming with Web 2.0 in recent years. Web users can create social annotations for different resources such as web pages, music, video and Web services. Existing Web service repositories such as serviceXchange[1], seekda[2] and servicefinder[3] allow users to create annotations for Web services. These annotations provide more useful information than the traditional keywords which extracted from WSDL documents. On one hand, these annotations can describe Web services in terms of functional attributes. On the other hand, the tagging relations and the tagging frequency indicate the popularity and importance of the services and annotations.

In this paper, we propose a social-annotation-based service discovery method by using a learning to rank method. Learning to rank[17] is a type of supervised or semi-supervised machine learning problem in which the goal is to automatically construct a ranking model from training data. After we construct the ranking model, we can easily rank the results by multiple features. Two major features we consider are dynamic Query-dependent feature and static Query-independent feature. For dynamic Query-dependent feature, we need to know the relevance between a query and results. For static Query-independent feature, we want to get the popularity and importance of the results.

The major contributions of this paper include: 1) We propose a service annotation model for service discovery. 2) Based on this model, we proposed the Query Annotation Relevance (QAR) algorithm to calculate the relevance between query and services, and Service Annotation Ranking (SAR) algorithm to calculate the static rank of annotations and services and operations. 3) Furthermore, we propose a ranking method for annotation-based service discovery by using those two algorithms above. 4) We evaluate the proposed method on a real dataSet which contains 22967 services, 260155 operations, 5316 annotations and 45725 tagging relations, the experimental results show that our method makes service discovery better.

The rest of paper is organized as follows. Section II discusses the related work. Section III proposes service annotation model, Query Annotation Relevance (QAR) and Service Annotation Ranking (SAR) in detail, and introduces the annotation-based Web service discovery method. Section IV presents experimental results and evaluation. Finally, we conclude our work in section V.

II. RELATED WORK

A. Service Discovery

There has been many previous work on service discovery. Woogle[7], proposed by Dong Xin and Alon Halevy, is a system which can provide Web service similarity search. Woogle use TF/IDF[13] algorithm combined with the method of association rules to calculate the similarity of the service.

[1] http://www.servicexchange.cn
[2] http://seekda.com/
[3] http://www.service-finder.eu/

Vector space search engine[8] makes the keywords of a service a vector, and then compares the similarity degrees between two documents by calculating the cosine value of the two vectors. Merobase is a search engine[9] based on WSDL document, and provides name search and interface search. These service discovery methods listed above are limited to the deficiency of the description of WSDL document, extracted keywords is not well enough to describe the services and operations.

On account of the insufficient describe information of WSDL, many studies have been conducted on semantic Web service. Reference [11] has discussed how to crawl both WSDL and API services, and generating service description in the process of crawling, using ontology and RDF to store the semantic Web service description. URBE (the UDDI Registry By Example) is introduced in [12] based on Web service semantic markup language (SAWSDL). URBE uses WordNet to calculate distance between the services, and got a promotion in recall ratio and precision ratio. Reference [10] discussed the ranking problem of the semantic Web service discovery, they ranked by the IOPE (Inputs Outputs Preconditions Effects) based on OWL-S model.

Compared with keyword methods, those methods which based on semantic Web service have high precision, but the cost of those methods is relatively high, and it is difficult to create and maintain an ontology repository, meanwhile, they need to do much manually work for semantic Web service ontology building and reasoning, these may lead to low scalability and inefficiency.

B. *Social Annotations*

Web2.0 and social annotations (or Tags) can enrich the descriptions of services, and provide a good foundation for service discovery and make it more efficient. There has been some research show that social annotations can effectively improve the performance of the document ranking in information retrieval[21, 22, 23]. Meanwhile, Reference [14] also proposed a kind of ranking method for web pages named FolkRank based on the social annotations.

There are few studies on the relation between service discovery and social annotations, in [18] social annotations are just simply regarded as keywords of the services, they implement a simple keyword search function based on WordNet, and also support 'AND','OR','NOT' operations. Reference [19] calculates the relations between the annotations according to the statistics of tagging relations, and then proposed the extended annotations search algorithm QEBT&QPBT. These two studies above consider service as a single resource which is similar with web pages, images and video. But they ignored the characteristics of the service in service discovery process, which is one service may have many operations, and the annotations which tagged on the service may not be able to describe all operations. Reference [20] define and calculate the hierarchies of annotations, and then implement service discovery by matching service name and operation name, it mainly focuses on the matching of the operation and the automatic composition of service.

Different from the existing methods, we propose a ranking method for annotation-based service discovery by using Query Annotation Relevance (QAR) algorithm to calculate the relevance between query and services, and Service Annotation Ranking (SAR) algorithm to calculate the static rank of annotations and services and operations.

III. Service Discovery with Social Annotation

In this section, we will introduce the social annotation-based service discovery. Section III.A introduces service annotation model. Query Annotation Relevance (QAR) and Service Annotation Ranking (SAR) will be discussed in Section III.B and III.C. In Section III.D, we will describe the service discovery method based on QAR and SAR.

A. *Service Annotation Model*

A common service annotation model in previous works is shown as follows:

$U=\{u_1,u_2,\ldots,u_k\}$ is a set of users, where $u_i(1\leq i\leq k)$ denotes a user, and k is the total number of users.

$A=\{a_1,a_2,\ldots,a_m\}$ is a set of annotations (or Tags), where $a_i(1\leq i\leq m)$ denotes an annotation, and m is the total number of annotations.

$R=\{r_1,r_2,\ldots,r_n\}$ is a set of resources, where $r_i(1\leq i\leq n)$ denotes a resource which can be a URL or a picture or a video, and n is the total number of resources.

And the tagging relation $L \subseteq U \times A \times R$.

Previous works [18, 19] consider Web service as a kind of resource, and use the same method which is also used on Web pages or other resources to optimize service discovery. But they ignore the difference between Web services and other resources, one Web service may have many operations, and the annotations which tagged on the service may not be able to describe all operations. So, annotations tagged on operations can make service discovery more precise.

In this paper, we extend the model mentioned above. In our model, resources include services and operations, users can tag both services and operations. As shown in Fig. 1, the relationship between annotations and services is a many-to-many relationship, and the relationship between annotations and operations is a many-to-many relationship too, which means one annotation can be tagged on different services or operations, and one service or one operation can have different annotations; The relationship between services and operations is one-to-many relationship, which means one service can have several operations, and one operation belongs to only one service.

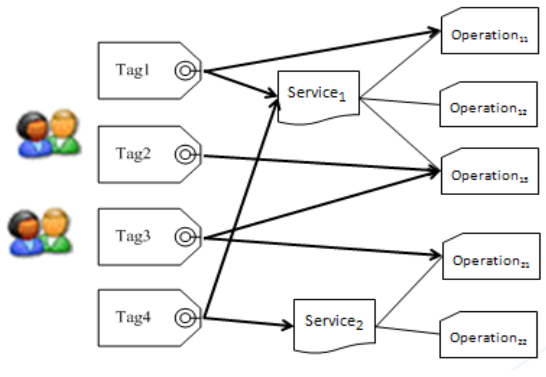

Figure 1. Illustration of service annotation model

Generally speaking, whether a resource is a relevant result to a user query depends not only on the resource itself and its annotations but other related resources as well. When calculating the relevance between a query and an operation, we should also consider the relevance between the query and the service which related to the operation. Section III.B will discuss Query Annotation Relevance (QAR) in detail. When calculating the rank of a resource, we should consider not only the importance and the popularity of the resource but other related resources as well. Section III.C will discuss Service Annotation Ranking (SAR) in detail.

B. Query Annotation Relevance (QAR)

1) Word Semantic Relevance

An annotation is a word with semantic meaning. A famous method for words relevance calculating is the dictionary-based approach, and usually the WordNet-based method is adopted. As mentioned in [24], the WordNet-based approach can be divided into three types: edge-based[10], node-based[11] and hybrid-based[12] approaches. In this paper, we choose the edge-based approach proposed by Leacock and Chodorow. The relevance between two words can be calculating by the following equation:

$$sim(w_1, w_2) = -\log \frac{min_{c_1 \in sen(w_1), c_2 \in sen(w_2)} len(c_1, c_2)}{2d_{max}} \quad (1)$$

where *sen(w)* denotes the set of possible senses of word *w*, d_{max} is the maximum depth of the taxonomy in WordNet. The $len(c_1,c_2)$ function is the calculation of shortest path's length between c_1 and c_2 in the taxonomy. Fig. 2 shows a segment of the hyponymy taxonomy in WordNet. Assuming c_1 is Car and c_2 is Bicycle, then $len(c_1,c_2)$=5.

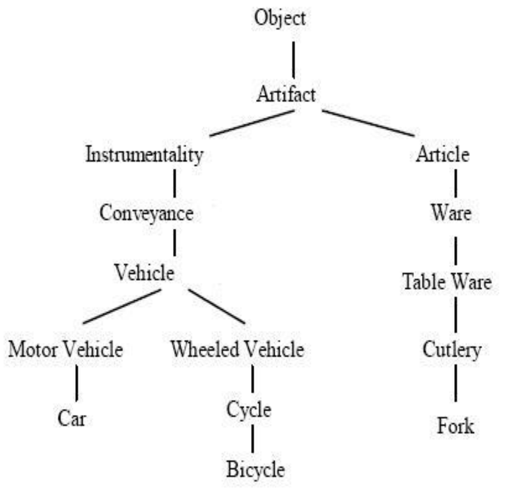

Figure 2. A segment of the hyponymy taxonomy in WordNet

2) Query Annotation Relevance (QAR)

Query Annotation Relevance aims to calculate the relevance between user's query and a service or an operation. In our work, we describe a service or an operation by a set of annotations, and the user query is a set of keywords. So, we calculate the relevance between the annotations set and the query keywords set as the QAR.

Calculating the relevance between two word sets, URBE[12] used a popular method called maximal weight matching method, as shown in Fig. 3, a graph G=(S_1, S_2, E), S_1={Tag_{1i}}, S_2={Tag_{2j}}, E=(<Tag_{1i},Tag_{2i}>, W_{ij}), edge weight W_{ij} is the word semantic relevance between Tag_{1i} and Tag_{2j}. The maximal weight matching method finds a set of edges, and has no two edges sharing the same vertex, and then calculates the sum of the weights. As shown is Fig. 3(a), the maximal weight matching of S_1 and S_2 is 0.9+0.7=1.6. But this method has some limitations, for those unmatched edges, their weights have no effect on the relevance of the two sets. For instance, we can see that the difference between Fig. 3(a) and Fig. 3(b) is the weight of the edge between Tag_{13} and Tag_{22}. Obviously, the relevance of S_1 and S_2 in Fig. 3(b) should be higher than the relevance of S_1 and S_2 in Fig. 3(a). But for the maximal weight matching method, they have the same relevance 0.9+0.7=1.6.

To solve this problem, we will consider the max weight edge of each vertex, and define the equation to calculate the relevance of two sets S_1 and S_2 as follow:

$$setSim(S_1, S_2) = \frac{\sum_{tag_{1i} \in S_1, tag_{2j} \in S_2} max(w_{ij})}{|S_1| + |S_2|} \quad (2)$$

According to (2), the relevance in Fig. 4(a) is (0.9+0.8+0.1+0.9+0.7)/(3+3)=0.567, and Fig. 4(b) is (0.9+0.8+0.6+0.9+0.7)/(3+3)=0.65. We notice that the relevance of the two sets in Fig. 4(b) is higher than the relevance of the two sets in Fig. 4(a), that means (2) can solve the problem that unmatched edges having no effect on the relevance of the two sets.

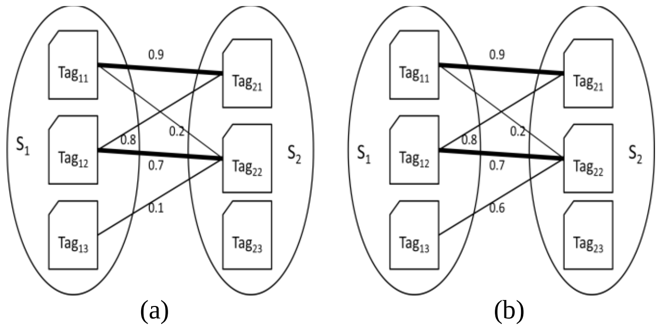

Figure 3. Maximal weight matching

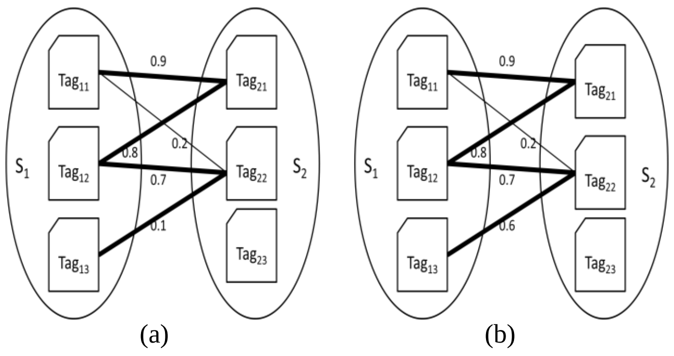

Figure 4. Query Annotation Relevance(QAR)

Since we have (2) for calculating the relevance of two sets, we can define the relevance between user query and a service as follow:

$$serviceSim(S_{query}, S_{service}) = setSim(S_{query}, S_{service}) \qquad (3)$$

where S_{query} is the set of user query keywords, $S_{service}$ is the set of annotations of the service.

Also, we define the relevance between user query and an operation as follow:

$$opSim(S_{query}, S_{op}) = w_1 \times setSim(S_{query}, S_{op}) + w_2 \times serviceSim(S_{query}, S_{service}) \qquad (4)$$

where S_{op} is the set of annotations of the operation, $S_{service}$ is the set of annotations of the service which includes the operation, w_1 and w_2 are factors.

3) *Calculating Query Results*

Since we have (3) and (4) for calculating the Query Annotation Relevance (QAR), we can simply calculate the QAR of all the services or operations and sort the results by their QAR. But for a large number of services and operations, calculating all the services and operations one by one is a time-consuming, inefficient way. So, we use the following method in Fig. 5 to calculate and sort the relevance of the results. Since users only care about the most relevant results, in this method, we do not calculate all the services and operations, we find the most relevant annotations first, and then find the services and operations which are tagged on these annotations, calculate their relevance.

Let *a* be the total number of annotations, *r* be the average number of resources of each annotation, and *n* is the input parameter in Fig. 5. We can calculate the time complexity by steps, the first step finding the *annotationSet* takes *O(a)* time, the second step finding the *serviceSet* and *opSet* takes *O(n)* time, and total number of *serviceSet* and *opSet* is *nr*. The final step is calculating and sorting the results, the time complexity mainly depends on the sorting algorithm, for quicksort algorithm, the time complexity of this step is *O(nr+nrlognr)*.

```
Query results calculating and sorting
Input: User's query, n

Steps:
    Foreach word in query do find top n annotations
    which have high relevance with the query word, and
    put these n annotations into a set named
    annotationSet

    Foreach annotation in annotationSet do find all the
    services which are tagged on this annotation, put
    these services into a set named serviceSet; and find
    all the operations which are tagged on this
    annotation, put these operations into a set named
    opSet

    Foreach service in serviceSet do calculate the
    relevance by equation(3)

    Sort all services in serviceSet by their relevance

    Foreach operation in opSet do calculate the
    relevance by equation(4)

    Sort all operations in opSet by their relevance

Output:
    Sorted services list and sorted operations list
```

Figure 5. Query results calculating and sorting

C. *Service Annotation Ranking (SAR)*

1) *Overview*

For static query-independent ranking, our basic notion is: if a service has more high quality annotations and more high quality operations, the service becomes high quality itself, and should be ranked higher; Similarly, if an operation is in a high quality service and has more high quality annotations, the operation should be ranked higher; If an annotation is tagged on more high quality services and operations, the annotation should be ranked higher.

The most famous existing ranking method is PageRank[1], its idea is using the links between pages to calculate the quality of pages, the rank of a page depends on the rank of other pages which have a link to this page. So, we take a similar method that the rank of a service is dependent on the number and the rank of the corresponding annotations and operations; the rank of an operation is dependent on the number and the rank of the corresponding annotations and service; the rank of an annotation is dependent on the number and the rank of the corresponding services and operations.

2) Service Annotation Ranking (SAR)

PageRank is a directed graph based method, but our model is an undirected graph based on Section 3.1.We have an undirected graph G=(V, E), where V is a vertex set including annotation set A, service set S and operations set O, $V = A \cup S \cup O$. E is an edge set including annotation-service relationship $X \subseteq A \times S$, annotation-operation relationship $Y \subseteq A \times O$, and service-operation relationship $Z \subseteq S \times O, E = X \cup Y \cup Z$.

Based on the undirected graph above, we propose our Service Annotation Ranking (SAR) method. Let M be the association matrix of the undirected graph, so M is a $|V|\times|V|$ symmetric matrix. Let R be a corresponding column vector, R_i is the rank of the corresponding vertex. Let N_i be the number of the edges of the *i-th* vertex, and N be the number of all vertices. Our SAR method is shown in Fig. 6.

We initialize the matrix and the column vector in Step1, the initial value of the matrix follow these rules: if the *i-th* vertex has N_i edges, then let the corresponding edge value be $1/N_i$ in the matrix, other values of the matrix should be set to 0. In Step2, we use α and β as the factor to accelerate convergence. Step3 outputs the final ranking results.

D. Service Discovery Method

With the growth of the Internet, there are different features which can affect ranking, learning to rank is an information retrieval and machine learning method which is adopted by many search engines. Learning to rank[17]is a type of supervised or semi-supervised machine learning problem in which the goal is to automatically construct a ranking model from training data. Training data consists of queries and documents matching them together with relevance degree of each match. It can be prepared manually by human assessors who check results for some queries and determine relevance of each result, or derived automatically by analyzing clickthrough logs.

In this paper we use a method called RankingSVM which proposed by T.Joachims in [15]. For convenience of learning to rank method, query-resources pairs are represented by numerical vectors, which are called feature vectors[17]. We consider two major features: query-dependent feature and query-independent feature. The QAR is query-dependent feature and the SAR is query-independent feature. After the RankingSVM method constructs a ranking model, we can have the final rank of the results. The process of service discovery is shown as Fig. 7, we have a tagging system for collecting the tagging relations, SAR runs background to calculate the query-independent rank, QAR receives a user query and calculate the relevance, and then RankingSVM puts these two features into a ranking model and then gets the final rank. We implement RankingSVM module by using SVM^{light}. SVM^{light} is an implementation of Support Vector Machine for the problem of pattern recognition and for the problem of learning a ranking function. Its goal is to learn a function from the features of the results and preference constraints, so that it orders a new set of objects as accurately as possible.

```
Service Annotation Ranking(SAR)
Step1:
        Initial:
        Foreach $R_i$ in $R^0$, Let $R_i=1/N$
        Foreach $(i,j) \in E$, Let $M_{ij}=1/N_i$, Otherwise, Let $M_{ij}=0$

Step2:
        Do{
                $R^{k+1} = \alpha R^k + \beta MR^k$
        }Until $|R^{k+1} - R^k|_1 < \varepsilon$

Step3:
        Output: $R^{k+1}$
```

Figure 6. Service Annotation Ranking(SAR)

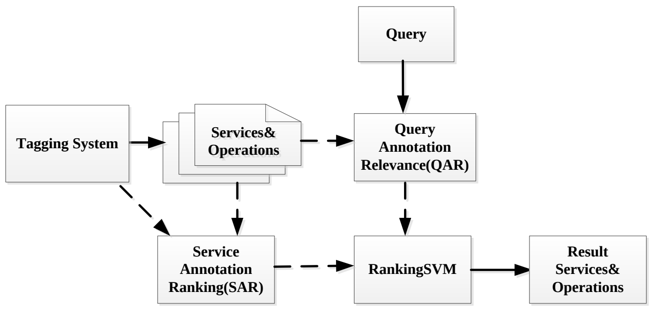
Query
Tagging System
Services& Operations
Query Annotation Relevance(QAR)
Service Annotation Ranking(SAR)
RankingSVM
Result Services& Operations

Figure 7. Process of service discovery

IV. EXPERIMENTAL RESULTS AND EVALUATION

A. DataSet

Our experimental data is from serviceXchange platform, serviceXchange has a web service crawler that can analyze and download web services from several major service repositories such as seekda[4], WebServiceList[5], webserviceX[6] and XMethods[7]. Our experiment is based on a dataset which contains 22967 services and corresponding WSDL files, and 838 annotations and 2859 tagging relations. Based on the WSDL files, we analyze the corresponding operations, 260155 operations in total.

1) Annotation Extraction

Due to lack of annotations, we analyze the WSDL files of the services and extract important keywords as new annotations. The process of extraction includes two steps. First, we do the text preprocess which includes removing symbol, splitting word, removing stop words and stemming. Second, we calculate the relevance of each keyword by using TF-IDF[13], and choose the best relevance keywords as new

[4] http://seekda.com/
[5] http://www.webservicelist.com/
[6] http://www.webservicex.net/WCF/default.aspx
[7] http://www.xmethods.net/ve2/index.po

annotations. Finally, we have 5316 annotations and 45725 tagging relations.

2) Annotation Normalization

Since annotations are created by different users in a free form, an uncontrolled annotation vocabulary is shared in the system, different people can use different terms for the same concept, so it is important to eliminate noise.

We observed the following types of divergence: the use of different word forms such as plurals, e.g. service, services; syntactic variance, e.g. fun, funny; synonym variance, e.g. graphics, image, photo.

To solve these problems, we analyze the data set, and use a WordNet-based stemmer to identify different words by their stems. When calculating the semantic relevance of two words, we will make the relevance be 1 if the words have the same stem.

B. Query Annotation Relevance (QAR) Evaluation

1) Case Study

Some of the QAR results of query{*weather*, *city*} is shown as the following Table I:

For query{*weather*, *city*}, service "*WeatherByCity*" has two matching annotations "*weather*" and "*city*", so the QAR of the service is 1. Service "*showTheWeather*" has one matching annotations "*weather*", its relevance is lower than service "*WeatherByCity*". Although service "*ProvinceCity*" has an annotation "*city*", another annotation "*province*" makes the relevance of the service lower than service "*showTheWeather*".

2) Time Performance

Table II shows the number of the query results, where *n* is the input of the method which is shown in Fig. 5. When *n* is 5, we can get a result list of 57 services and 168 operations. Fig. 8 shows the time performance of calculating QAR. We can see that there are a few results when *n* is small, and the time performance is good. When *n* becomes bigger, there are more results and the calculating time is growing too. So, when the number of service or operation is large, we can improve time performance by adjusting *n*.

TABLE I. QUERY RESULTS

Service	*Annotation*	*QAR*
WeatherByCity	weather,city	1
showTheWeather	weather	0.68291
ProvinceCity	city, province	0.67085
WeatherWS	Weather, region	0.60785

TABLE II. NUMBER OF QUERY RESULTS

n	*Service*	*operation*
5	57	168
10	64	235
20	136	399
30	214	624
50	730	1064

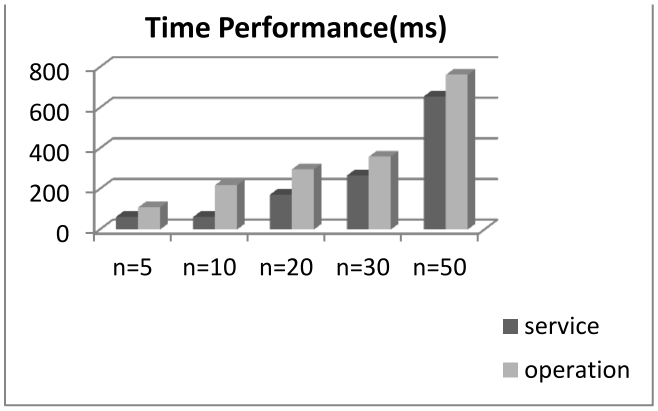

Figure 8. Time performance of QAR

C. Service Annotation Ranking (SAR) Evaluation

1) Case Study

Table III and Table IV show the top rank annotations and services.

Form these two tables, we can see that there are several factors have impact on the final rank. For annotation rank, tagging frequency is an important factor, but not decisive. Annotation "*hello*" which has 1840 times is lower than annotation "*registration*" which has 1263 times, and the rank of "*company*" is higher than "*download*" and "*blog*". For service rank, we can see that some high-ranked services have a large number of operations, but meanwhile, some services such as "*FollowTheGameService*" have many annotations but few operations. We can see that our SAR method takes both factors into consideration. We will do some work in future to optimize the ranking algorithm.

2) Time Performance

The time performance of SAR depends on the convergence times, so we choose different α and β to evaluate the results.

TABLE III. ANNOTATIONS RANK

Annotation	*Rank*	*Frequency*
registration	0.01296	1263
hello	0.01112	1840
company	0.00203	359
download	0.00169	482
blog	0.00153	400

TABLE IV. SERVICES RANK

Service	*Rank*	*Annotation*	*Operation*
interface	0.00146	1	582
directConnect	0.00069	2	546
dataService	0.00053	1	210
FollowTheGameService	0.00028	33	4
SportDisc	0.00026	2	108
LiveScoresService	0.00023	3	63

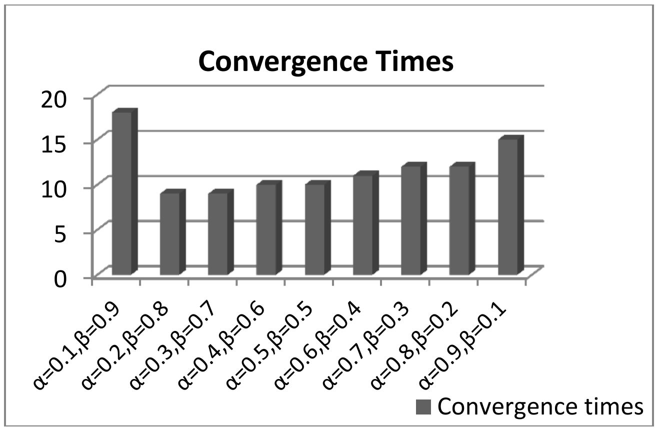

Figure 9. Convergence times of SAR

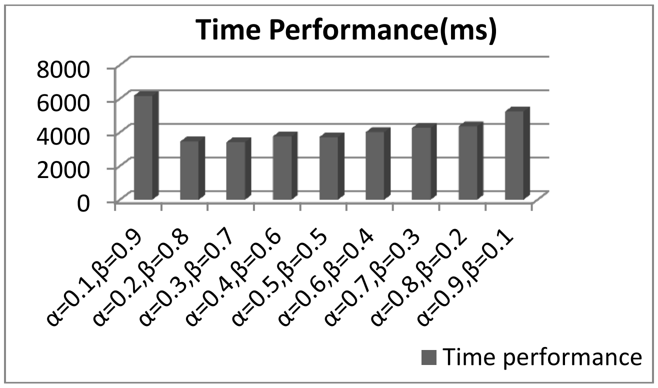

Figure 10. Time performance of SAR

Fig. 9 and Fig. 10 show that there is a positive correlation between the convergence times and time performance. When α is 0.1 and β is 0.9, the rank needs more iterations to converge. Generally speaking, the convergence times is no more than 20, and the time performance is acceptable.

D. Learning to Rank

In this section, we implement a TF-IDF keyword searching method to make comparisons, and we use a dataset manually collected by a group of students, they help us to collect 100 queries and the corresponding evaluations. We choose 90 records as our training data, and 10 records as testing data.

1) Mean Average Precision(MAP)

We evaluate learning to rank method and keyword method by a popular metric, *Mean Average Precision(MAP)*. *MAP*[25] emphasizes ranking relevant documents higher. It is the average of precisions computed at the point of each of the relevant documents in the ranked sequence.

The *Average Precision(AP)* for each query is defined as:

$$AP = \frac{1}{r}\sum_{i=1}^{n} P(i) \times rel(i) \qquad (5)$$

TABLE V. COMPARISON OF MAP

Method	*MAP*
Keyword	0.725
Learning to rank	0.852

where i is the position in the results list , n the number retrieved, r is the number of relevant results, *rel()* a binary function on the relevance of a given result position, and *P(i)*denotes the precision of the top i results.

Mean Average Precision(MAP) for a set of queries is the mean of the average precision scores for each query

$$MAP = \frac{1}{m}\sum_{i=1}^{m} AP_i \qquad (6)$$

where m is the number of queries.
Table V shows the *MAP* comparison between two methods.

2) Normalized Discounted Cumulative Gain (NDCG)

We also evaluate learning to rank method and keyword method by *Normalized Discounted Cumulative Gain (NDCG)*[16], another famous evaluation metric for learning to rank. We can calculate NDCG by the following equation:

$$NDCG@K = Z_k \sum_{i=1}^{k} \frac{2^{R(i)}-1}{log(1+i)} \qquad (7)$$

where *R(i)* is the relevance of the *i-th* result. Z_k is a factor to make the NDCG value of a perfect result ranking is 1. And $1/\,log(1+i)$ is a factor which can make those high-ranked results more important than the lower-ranked results. *K* is the results number. Popular evaluation metrics are NDCG@1, NDCG@3, NDCG@5, NDCG@10.

Fig. 11 shows the comparison between our learning to rank method and TF-IDF keyword searching method. Compared to keyword searching method, our learning to rank method has better NDCG. It shows that our method has satisfied results.

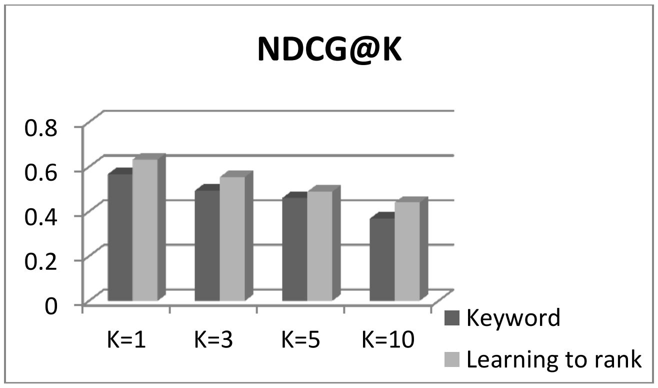

Figure 11. Comparison of NDCG@K

V. Conclusion and Future Work

In this paper, we observed that social annotations can help to describe Web services and make service discovery efficient. So we presented a service annotation model for service discovery. Based on this model, we proposed Query Annotation Relevance (QAR) algorithm to calculate the relevance between query and services, and Service Annotation Ranking (SAR) algorithm to calculate the static rank of annotations and services and operations. Furthermore, we discussed a ranking method for annotation-based service discovery by using those two algorithms above.

Experimental results show that both QAR and SAR algorithms have acceptable results and good time performance. And compared to traditional TF-IDF keyword searching method, our learning to rank method has an improvement for service discovery.

For future work, we plan to improve the word semantic relevance method, because the vocabulary of WordNet is limited. And we will record the users query log and clickthrough log as our training data to optimize our learning to rank method.

VI. Acknowledgment

This work was supported partly by National Natural Science Foundation of China under Grant No. 61103031, partly by the State Key Lab for Software Development Environment under Grant No. SKLSDE-2010-ZX-03, and partly by the Fundamental Research Funds for the Central Universities under Grant No. YWF-11-03-Q-040, YWF-11-05-013 and YWF-11-02-206.

VII. References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd. ThePageRank citation ranking: Bringing order to the web.Technical report, Stanford Digital Library TechnologiesProject, 1998.

[2] Curbera, F., et al., Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 2002. 6(2): 86-93.

[3] Paolucci M, Kawamura T, Payne TR, Sycara K. Semantic matching of Web services capabilities. In: Horrocks I, ed. Proc. of the 1st Int'l Semantic Web Conf. Chia: Springer-Verlag, 2002: 333−347.

[4] Bansal S, Vidal JM. Matchmaking of Web services based on the DAML-S service model. In: Rosenschein JS, ed. Proc. of the 2nd Int'l Joint Conf. on Autonomous Agents and Multiagent Systems. Melbourne: ACM Press, 2003: 926−927.

[5] Shi ZZ, Jiang YC, Zhang HJ, Dong MK. Agent service matchmaking based on description logic. Chinese Journal of Computers, 2004,27(5): 626-635 (in Chinese with English abstract).

[6] Ma YL, Jin BH, Feng YL. Dynamic discovery for semantic Web services based on evolving distributed ontologies. Chinese Journal of Computers, 2005,28(4): 603-615 (in Chinese with English abstract).

[7] Xin Dong, Alon Halevy, JayantMadhavan, EmaNemes, Jun Zhang, Similarity Search for Web Services, 30th VLDB, 2004

[8] Platzer C, Dustdar S. A vector space search engine for Web services[C]. Proceedings of IEEE European Conferenceon Web Services (ECOWS).

[9] Atkinson C, Bostan P, Hummel O, et al. A practical approachto Web service discovery and retrieval[C]. Proceedingsof International Conference on Web Services(ICWS'07), 2007: 241−248.

[10] UmeshBellur and Harin Vadodaria. Web Service Ranking Using Semantic Profile Information.ICWS, 2009

[11] Nathalie Steinmetz, HolgerLausen, and Manuel Brunner.Web Service Search on Large Scale.ICSOC2009.

[12] Pierluigi Plebani and Barbara Pernici. URBE: Web serviceRetrieval based on Similarity Evaluation. IEEE Transactions onKnowledge and Data Engineering, 16 Jan. 2009.

[13] K. CHURCH, W. GALE, Inverse Document Frequency (IDF): A Measure of Deviations from Poisson, Natural Language Processing Using Very Large Corpora, 1999 .

[14] Andreas Hotho, Robert Jaschke, Christoph Schmitz, Gerd Stumme . Information Retrieval in Folksonomies:Search and Ranking. The Semantic Web:Research and applications,2006,4011:411-426.

[15] T. Joachims. Optimizing search engines using clickthroughdata. In: Proc. of SIGKDD 2002, pp. 133-142, 2002.

[16] K Jarvelin and J. Kekalainen. IR evaluation methods forretrieving highly relevant documents. In Proc. of SIGIR 2000,2000.

[17] http://en.wikipedia.org/wiki/Learning_to_rank

[18] Uddam CHUKMOL, Aicha-Nabila BENHARKAT, Youssef AMGHAR.Enhancing Web Service Discovery by using Collaborative Tagging System.4th NWeSP,2008.

[19] Zhaoyun Ding, Deng Lei, Jia Yan, Zhou Bin, An Lun.A Web Service Discovery Method Based on Tag.2010 International Conference on Complex,Intelligent and Software Intensive Systems,2010.

[20] Eric Bouillet et al.A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition.SCC 2008,2008.

[21] S. Bao, X. Wu, B. Fei, G. Xue, Z. Su, and Y. Yu. Optimizing Web Search Using Social Annotations. InProc. of WWW 2007, pp.501-510, 2007

[22] D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring Social Annotations for InformationRetrieval. In Proc. of WWW2008, pp. 715-724, 2008

[23] S. Xu, S. Bao, Y. Cao, and Y. Yu.Using Social Annotations to Improve Language Model for InformationRetrieval. In Proc. of CIKM 2007, pp. 1003-1006, 2007

[24] Xianyang Qu, Hailong Sun, Xiang Li, Xudong Liu, Wei Lin . WSSM: A WordNet-Based Web Service Similarity Mining Mechanism

[25] http://en.wikipedia.org/wiki/Information_retrieval#Mean_average_pr ecision