Titan: a System for Effective Web Service Discovery

Jian Wu
Zhejiang University
Hangzhou, China
wujian2000@zju.edu.cn

Liang Chen
Zhejiang University
Hangzhou, China
cliang@zju.edu.cn

Yanan Xie
Zhejiang University
Hangzhou, China
xyn@zju.edu.cn

Zibin Zheng
The Chinese University of Hong Kong
Hong Kong, China
zbzheng@cse.cuhk.edu.hk

ABSTRACT

With the increase of web services and user demand's diversity, effective web service discovery is becoming a big challenge. Clustering web services would greatly boost the ability of web service search engine to retrieve relevant ones. In this paper, we propose a web service search engine *Titan*[1] which contains 15,969 web services crawled from the Internet. In *Titan*, two main technologies, i.e., web service clustering and tag recommendation, are employed to improve the effectiveness of web service discovery. Specifically, both WSDL (Web Service Description Language) documents and tags of web services are utilized for clustering, while tag recommendation is adopted to handle some inherent problems of tagging data, e.g., uneven tag distribution and noise tags.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; H.3.3 [**Information Search and Retrieval**]: Search process; H.3.4 [**Systems and Software**]: Information networks

General Terms

Design,Management,Performance

Keywords

Web service discovery, clustering, tag, WSDL

1. INTRODUCTION

Web service discovery can be achieved by two main approaches: UDDI (*Universal Description Discovery and Integration*) and web service search engines. Recently, the availability of web services in UDDI decreases rapidly as many web service providers decided to publish their web services through their own website instead of using public registries. Al-Masri *et al.* show that more than 53% of the UDDI business registry registered services are invalid, while 92% of web services cached by web service search engines are valid and active [1]. Compared with UDDI, using search engine to search and discover web services becomes more common and effective.

Searching for web services using web service search engines is typically limited to keyword matching on names, locations, businesses, and buildings defined in the web service description file [9]. If the query term does not contain at least one exact word such as the service name, the service is not returned. It is difficult for users to be aware of the concise and correct keywords to retrieve the satisfied services. The keyword-based search mode suffers from low recall, where results containing synonyms or concepts at a higher (or lower) level of abstraction describing the same service are not returned. For example, a service named "*Mobile Messaging Service*" may not be returned from the query term "*SMS*" submitted by the user, even these two keywords are obviously the same at the conceptual level.

To handle the drawbacks of web service search engines, web service clustering is proposed to help retrieve more related services. For example, given a web service ws_1 is one search result of query q, then the web services in the same cluster with ws_1 would be selected to extend the search result of q as these services maybe related to q. In previous works [5][8], only WSDL documents are utilized in the process of web service clustering, which limits the performance of web service clustering. In this paper, we propose to utilized not only WSDL documents which reflect service's functionality, but also tags which reflect user's knowledge in the process of web service clustering. Specifically, we extract 5 features from WSDL documents and generate the similarity between web services by combining feature-based similarities and tag-based similarity.

As tags are associated by users to web services, there are some inherent properties of tagging data that impact the reliability of tag-base similarity, e.g., uneven tag distribution and noise tags. In this paper, we propose a hybrid tag recommendation approach, which employs tag mining, tag co-occurrence, and semantic relevance technologies, to handle these problems.

We crawl 15,969 real web services from the Internet and build a demonstration of web service search engine called *Titan* based on these web services. Both web service clustering approach and hybrid tag recommendation approach are employed in *Titan* to facilitate the process of web service discovery.

[1]Titan Search Engine (http://ccnt.zju.edu.cn:8080)

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

The remainder of this paper is organized as follows. In Section 2, we review related work. The architecture of Titan is introduced in Section 3, while the detailed web service clustering process is introduced in Section 4. After an outline of the tag recommendation strategy in Section 5, we give an introduction of *Titan* user interface in Section 6.

2. RELATED WORK

With the development of service computing and cloud computing, web service discovery is becoming a hot research topic. A lot of work has been done to handle this problem. Xin Dong *et al.* propose to compute the similarity between web services employing the structures of web services (including name, text, operation descriptions, input/output description, etc) [4]. They also propose a search engine called Woogle which supports similarity search for web services. Nayak attempts to handle the service discovery problem by suggesting other related search terms to the current user based on what other users had used in similar queries by using clustering techniques [9]. Nayak proposes to cluster web services based on search sessions instead of individual queries. Yilei Zhang *et al.* provides three search styles to discover Web services, by considering both functional similarities to users' queries and non-functional QoS characteristics of Web services [11].

Recently, web service clustering is presented as a novel solution to the problem of service discovery. Liu *et al.* propose to extract 4 features, i.e., *content, context, host name, and service name*, from the WSDL document to cluster web services [8]. They take the process of clustering as the preprocessor to discovery, hoping to help in building a search engine to crawl and cluster non-semantic web services. Khalid *et al.* also propose to extract features from WSDL documents to cluster web services [5]. Different from Liu's work, Khalid extracts *content, types, messages, ports, and service name* from WSDL documents. In our previous work, we utilize both WSDL documents and Tagging data to cluster Web services for the purpose of Web service discovery [2].

Compared with above work, we introduce tag mining and semantic relevance technologies to improve the performance in *Titan* system. Specifically, we propose a hybrid tag recommendation approach to handle the problems of uneven tag distribution and tag noises which impact the reliability of tag-based similarity in the process of web service clustering.

3. ARCHITECTURE

Figure 1 shows the basic architecture of our proposed *Titan* system. The module of *Service Clustering* is employed to do web service clustering for web service discovery, while the module of *Tag Recommendation* is employed to smooth the tagging data utilized in the process of service clustering.

In the module of *Service Clustering*, we first extract features (i.e., content, type, message, port, and name) from WSDL documents and compute the corresponding feature-based similarity between web services. Before computing tag-based similarities, the module of *Tag Recommendation* has to be implemented, in which tag mining, tag co-occurrence, and semantic relevance are employed. After the tagging data is smoothed by processing *Tag Recommendation*, we combine feature-based similarities with tag-based similarity to cluster web services. In the realization of *Titan* system, we

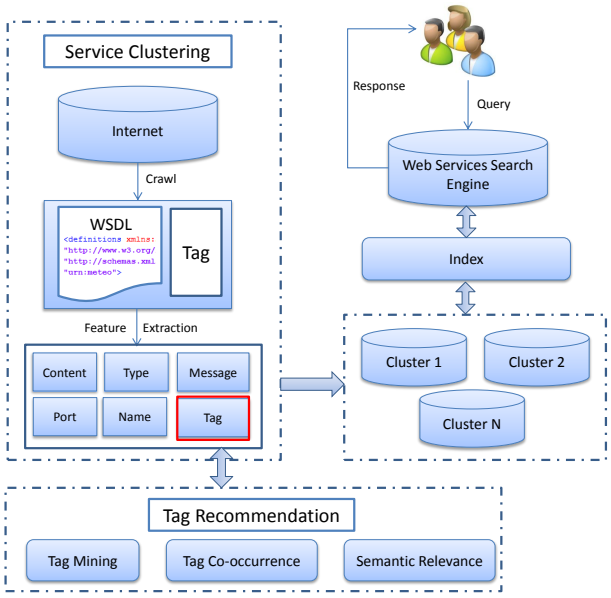

Figure 1: Architecture of *Titan* System

build the indexing of these clustered web services to improve the efficiency of web service discovery.

When a web service query is coming, we not only return web services which are mostly semantic related to the query, but also return some services in the same cluster for the purpose of retrieving more related services.

4. WEB SERVICE CLUSTERING

In *Titan* system, we extract five features (i.e., *Content, Type, Message, Port, and Service Name*) from web service's WSDL document, and use these five features and tags to cluster web services. In this section, we describe the detailed process of feature extraction, similarity computation, and clustering approach.

Content. WSDL document, which describes the function of web service, is actually a XML style document. Therefore, we can use some IR approaches to extract a vector of meaningful content words which can be used as a feature for similarity computation. The process of content vector extraction includes 4 steps: 1) Initialization, building an initial content vector by splitting WSDL document; 2) Suffix Stripping, stripping the suffix of all words that have the same stem, e.g., connect, connected, connecting, and connection, by using a Porter stemmer [10]; 3) Pruning, removing XML tags (e.g., *s:element* and *s:type*) and content words [7]; 4) Refining, removing words with very high occurrence frequency which are considered too general to discriminate between web services.

Given two web services ws_1 and ws_2, we can get the content vectors of these two web services Con_{ws_1} and Con_{ws_2}. Then the content-based similarity between ws_1 and ws_2 is defined as follows:

$$Sim_{con}(ws_1, ws_2) = \frac{\sum_{w_i \in Con_{ws_1}} \sum_{w_j \in Con_{ws_2}} sim(w_i, w_j)}{|Con_{ws_1}||Con_{ws_2}|}, \quad (1)$$

where $|Con_{ws_1}|$ means the cardinality of vector Con_{ws_1}, and $simw_i, w_j$ means semantic similarity between two words w_i

and w_j. Specifically, this semantic similarity between words is calculated by employing NGD (Normalized Google Distance) [3].

Type. In a WSDL document, each input and output parameter contains a name attribute and a type attribute. Sometimes, parameters may be organized in a hierarchy by using complex type. Due to different naming conventions, the name of parameter is not always a useful feature, whereas the type attribute which can partially reflect the service function is a good candidate feature. Type-based similarity between two web services is defined as follows:

$$Sim_{type}(ws_1, ws_2) = \frac{2 \times Match(Type_{ws_1}, Type_{ws_2})}{|Type_{ws_1}| + |Type_{ws_2}|}, \quad (2)$$

where $Type_{ws_1}$ means the set of types defined in ws_1's WSDL document, $Match(Type_{ws_1}, Type_{ws_2})$ means the number of matched types between ws_1 and ws_2.

Message. Message is used to deliver parameters between different operations. One message contains one or more parameters, and one parameter is associated with one type as we discussed above. Message definition is typically considered as an abstract definition of the message content, as the name and type of the parameter contained in the message are presented in the message definition. Similar to Eq.2, we match the messages' structures to compute the message-level similarity $Sim_{mes}(ws_1, ws_2)$ between web services.

Port. The portType element combines multiple message elements to form a complete one-way or round-trip operation. As the portType consists of some messages, we can get the match result of portType according to the match result of messages. Similar to the computation of type-level and message-level similarity, we also use Eq.2 to compute the port-level similarity $Sim_{port}(ws_1, ws_2)$.

Service Name. As the service name (*sname*) can partially reflect the service function, it is an important feature in WSDL document. Before computing the *sname*-level similarity, we first implement a word segmentation process to service name. After the process of word segmentation, $s_1's$ name $SName_{s_1}$ can be presented as a set of words. And then we can use Eq.1 to compute the *sname*-level similarity between web services.

Tag. The tagging data of web services describes the function of web services or provide additional contextual and semantical information. In this paper, we propose to improve the performance of traditional WSDL-based web service clustering by utilizing the tagging data. Given a web service s_i contains three tags t_1, t_2, t_3, we name the tag set of s_i as $T_i = \{t_1, t_2, t_3\}$. According to the Jaccard coefficient [6] method, we can calculate the tag-level similarity between two web services s_i and s_j as follows:

$$Sim_{tag}(s_i, s_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}, \quad (3)$$

where $|T_i \cap T_j|$ means the number of tags that are both annotated to s_i and s_j, and $|T_i \cup T_j|$ means the number of unique tags in set T_i and T_j, i.e., $|T_i \cup T_j| = |T_i| + |T_j| - |T_i \cap T_j|$.

To cluster web services, we have to combine the five feature-based similarities and tag-based similarity together to be a global similarity between services. As WSDL document is provided by service provider and tags reflect user's knowledge, we first combine five feature-based similarities to be a WSDL-based similarity $Sim_{wsdl}(ws_1, ws_2)$ and then generate the global similarity by integrating $Sim_{wsdl}(ws_1, ws_2)$ and $Sim_{tag}(ws_1, ws_2)$. The generation of $Sim_{wsdl}(ws_1, ws_2)$ can employ basic weighted mean approach or some approaches else [5]. And the global similarity $Sim_{global}(ws_1, ws_2)$ is computed as follows:

$$\begin{aligned} Sim_{global}(ws_1, ws_2) = \lambda Sim_{wsdl}(ws_1, ws_2) + \\ (1 - \lambda) Sim_{tag}(ws_1, ws_2), \end{aligned} \quad (4)$$

where λ is used to balance the weights of WSDL documents and tags. After the computation of global similarities between web services, density-based clustering approach is employed to do web service clustering.

5. TAG RECOMMENDATION

Tags of web services reflect users' knowledge which are important to web service clustering. However, some inherent properties of web service tagging data, e.g., uneven tag distribution and noise tags, impacts the reliability of tag-level similarity. To handle the problem, we propose a three-fold tag recommendation approach.

Tag Mining. As some web services have no tags and traditional tag recommendation can not handle this kind of *cold start* problem, we propose to mine tags from web service's textual features. The assumption of tag mining is that the more important this term in the textual features, the more probable for this term to be attached as a tag. In this paper, we use TF-IDF approach, which is widely adopted in the domain of Information Retrieval and Nature Language Processing. Specifically, textual features used for tag mining contain three parts: 1) service name; 2) service description given by service provider; and 3) content words extracted from WSDL document.

Tag Co-occurrence. Tag co-occurrence is a common used tag recommendation method for web services with few tags. In our preliminary work, we propose a simple tag co-occurrence based approach by employing Jaccard coefficient method [6]. Association rule is also employed to do tag co-occurrence based recommendation in recent works. In our Titan system, we prefer to extract association rules to do tag recommendation.

Semantic Relevance. Tags recommended only based on tag co-occurrence sometimes maybe irrelevant to the target web service due to some strong association rules. Thus, the relevance between candidate tag and target web service should be considered in the process of tag recommendation. In *Titan* system, NGD distance is employed to evaluate the semantic relevance between tags and services. Further, the introduction of semantic relevance also filters the noise tags (e.g., misspellings or unrelated terms) or reduces their importance.

6. USER INTERFACE

Our Titan system is online variable, users can use Titan search engine to discover web services by visiting http://ccnt.zju.edu.cn:8080. Figure 2 shows the search result page while user uses *weather* as query term. From Fig.2, we can find that each search result entity contains four parts: 1) web service name; 2) service description; 3) tags given by users; and 4) service provider. The clustering result of retrieved web services is shown in the division named *Filter by Clustering* in the right side of search result page. Specifically, we use an unique color to represent one cluster. While users click one

color block in the *Filter by Clustering* division, web services in the corresponding cluster will be removed from the search result. Service provider is also employed as a metric to filter web services in the search result. In the division of *Filter by Provider*, there is a list of service providers. Once user clicks the symbol of one service provider, web services provided by this provider will be removed from search result.

Figure 2: Search Result Page

Figure 3: Compare Result Page

Users can also compare the similarity between two web services in the search result of *Titan*. As Fig.3 shows, we compare the web service *Global Weather* (the bottom one) with another 6 web services listed in the search result of *weather*. It should be noted that the up *Global Weather* is the same as the bottom one, but provided by different service providers. The similarity between these two web services is not 1 because the tags associated to these two services are different. As discussed above, the global similarity is composed by 5 feature based similarities and tag based similarity. Specifically, different color in the bar of global similarity represents the corresponding feature based similarity or tag based similarity.

7. ACKNOWLEDGMENTS

This research is was partially supported by the National Technology Support Program under grant of 2011BAH15B05, the National Natural Science Foundation of China under grant of 61173176, Science and Technology Program of Zhejiang Province under grant of 2008C03007, National High-Tech Research and Development Plan of China under Grant No.2009AA110302, National Key Science and Technology Research Program of China (2009ZX01043-003-003).

8. REFERENCES

[1] E. Al-Masri and Q. H. Mahmoud. Investigating web services on the world wide web. *International World Wide Web Conference*, pages 795–804, 2008.

[2] L. Chen, L. Hu, Z. Zheng, J. Wu, Y. Li, and S. Deng. Wtcluster: Utilizing tags for web services clustering. *International Conference on Service Oriented Computing*, pages 204–218, 2011.

[3] Cilibrasi, R. L, Vitnyi, and P. M. B. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. *International Conference on Very Large Data Bases*, pages 372–383, 2004.

[5] K. Elgazzar, A. E. Hassan, and P. Martin. Clustering wsdl documents to bootstrap the discovery of web services. *International Conference on Web Services*, pages 147–154, 2009.

[6] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[7] C. K. and G. W. Inverse document frequency (idf): a measure of deviations from poisson. *Proceedings of the ACL 3rd workshop on Very Large Corpora*, pages 121–130, 1995.

[8] W. Liu and W. Wong. Web service clustering using text mining techniques. *International Journal of Agent-Oriented Software Engineering*, 3(1):6–26, 2009.

[9] Nayak and Richi. Data mining in web service discovery and monitoring. *International Journal of Web Services Research*, 5(1):62–80, 2008.

[10] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] Y. Zhang, Z. Zheng, and M. R. Lyu. Wsexpress: A qos-aware search engine for web services. *International Conference on Web Services*, pages 91–98, 2010.