



Facilitating the Service Discovery for the Cluster of Web Services using Hybrid WSTRec

Nandini N, Divya K V

Assistant Professor, Department of ISE
New Horizon College of Engineering
Bangalore, India

Abstract- *Clustering Web services would greatly increase the ability of Web service search engine to retrieve relevant services. The performance of traditional Web service description language (WSDL)-based Web service clustering is not satisfied, due to the singleness of data source. Recently, Web service search engines such as Seekda allow users to manually annotate Web services using tags, which describe functions of Web services or provide additional contextual and semantical information. In this paper, we group the Web services by utilizing both WSDL documents and tags. To handle the clustering performance limitation caused by uneven tag distribution and noisy tags, we propose a hybrid Web service tag recommendation strategy, named WSTRec, which employs tag co-occurrence, tag mining, and semantic relevance measurement for tag recommendation. Finally the proposed approach produces gain in both precision and recall compared to traditional WSDL based service clustering.*

Keywords - *Web service • Clustering • Tag recommendation • Service discovery • tag co-occurrence*

I. INTRODUCTION

A service-oriented computing (SOC) paradigm and its realization through standardized Web service technologies provide a promising solution for the seamless integration of singlefunction applications to create new large-grained and value-added services. Web service discovery can be achieved by two main approaches: universal description discovery and integration (UDDI) 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 Web site instead of using public registries. Al-Masri et al. [1] show that more than 53% of registered services in UDDI business registries are invalid, while 92% of Web services cached by Web service search engines are valid and active. Compared with UDDI, using search engine to discover Web services becomes common and effective.

In the field of service computing, Web service search is typically limited to keyword matching on names, locations, businesses, and buildings defined in the Web service description file [2]. 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 targeted services satisfactorily. 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.

To handle the drawbacks of the keyword-based Web service search engines, some approaches are proposed to extend the search result. Lim et al. [3] propose to make use of ontology to return an expanded set of results including subclass, superclass, and sibling classes of the concept entered by the user, while Elgazzar et al. [4] and Liu et al. [5] propose to handle the drawbacks of traditional search engine by clustering Web services based on WSDL documents. In these previous works, if Web services with similar functionality are placed into the same cluster, then more relevant Web services could be included in the search result. In fact, many experiments have demonstrated that good Web service clustering boosts the performance of Web service search. In this paper, we attempt to improve the performance of Web service clustering for the purpose of more accurate Web service discovery.

The contribution of this paper can be summarized as follows:

- We propose a Web service clustering approach, which improves the performance of Web service clustering by utilizing both WSDL documents and Web service tags.
- We propose a hybrid tag recommendation strategy to attack the service clustering performance limitation caused by uneven tag distribution and noisy tags.

II. RELATED WORK

With the development of service computing, searching Web services according to users' requirements, i.e., Web service discovery, is becoming a hot research topic. Web services can be classified into two main kinds, i.e. semantic and non-semantic, based on the description language. Approaches for discovering semantic and non-semantic Web services

are different. The semantic-based approaches adopt the formalized description languages such as OWL-S[4] and WSMO[5] for services and develop the reasoning-based similarity algorithms to retrieve the satisfied Web services [12,13]. High level match-making approaches are usually adopted in the discovery of semantic Web services. Specifically, BoualemBenatallah et al. [14] propose to tackle the problem of service discovery in the context of description logics. Aabhas V. Paliwal et al. [1] projected semantic based service discovery. The web service discovery relies on semantic categorization of web services and semantic enhancement of service request. The semantic categorization of web services is achieved by ontology framework as offline in UDDI. Semantic enhancement of web services is achieved by parameter based service refinement and semantic similarity based matching. The matching of increased service request with retrieved service description is achieved by Latent Semantic Indexing (LSI). Ranking of semantic relationships, hyper clique pattern discovery, additionally used for the invention. Solely single web service is taken into account for matching service request. Therefore web service composition is not satisfied.

JebersonRetna Raj et al. [2] projected web service discovery based on computation of semantic similarity distance and QOS normalization. The author introduces functional and nonfunctional analysis for computing the similarity activity. The WSDL program extracts the Meta information contents from the WSDL file and it will be preprocessed. Presently once preprocessing, acceptable terms are going to be known. Semantic distance between terms is going to be calculated by Normalized Google Distance. Web service similarity may be determined by using bipartite graph algorithm. The score worth is going to be computed based on aggregation of purposeful and non purposeful activity values. For a user request, an inventory of candidate services are going to be provided to the client based on degree of matching with the search query from higher to lower order. Similarity measurement algorithm is employed for sheming the similarity between two terms. The QOS attribute like time interval, throughput, dependableness and availableness may be normalized. The similarity distance that may be obtained from Google that provides correct weight to the several terms. Solely trained and novice user will discover the services based on keyword and operation based mostly queries. Guisy Di Lorenzo et Al. [3] introduces towards semantic driven generation of executable web services compositions. A life cycle is projected for the automated composition of web services and also the description of data relies on the usage of domain ontologies. Verifiable and valid executable method may be created and that is achieved by exploiting formal definitions of composition rules and of BPEL4WS constructs. Operational semantics is that the formal basis of all the life cycle phases. The development method could begin from a composition goal that describes a particular customer's request or additionally from a composition goal that needs building an web service ready to satisfy a category of requests. The life cycle approach consists of automated synthesis of two graph model like the operations flow model and service work flow model. In operational flow model, choose set of (available) service operations that are semantically compactable. And in service workflow model confirm a composition of such operations that semantically matches preconditions and effects of the requester composite service. Approach addresses the generation of each ad-hoc and re-usable compositions. Check and supply the Input/output matching of the chosen operations; the applying of graph transformation techniques so as to specific the composition by suggests that of workflow patterns. Since life cycle model is employed, the standard of services and also the security needs are not considered.

Puwel Wang et al. [4] introduce building toward capability specifications of web service based on environment ontology. Automated web service discovery needs web service capability specifications of web service supported on environment ontology. The most ideas of environment ontology are the environment entities during a explicit application domain and their interactions. For every environment entity, there is a tree like hierarchic state machine modeling the consequences that are to be achieved by the web services on this environment entity. Algorithms made for domain environment ontology and match making between the web service capabilities shows however a web service discovery. supported. The algorithm for constructing environment ontology from general domain ontology has additionally been delineated that the environment ontology is made. Associate effect-based capability profile that creates the web service specification a lot of correct while not exposing the web service's realization details and also the structured effects of the sharable environment entities modeled by FCHMs that build the web service specification a lot of comprehensible with one another and build the web service capability communicatory. The algorithm has additionally been developed to transfer the effect-based profile into a capability specification mechanically. This framework is not ready to develop an effective service discovery support system.

III. ARCHITECTURE OF WSTRec

The figure shows the architecture of our proposed Web service clustering approach. It consists of three modules: (1) data preprocessing, (2) WSTRec (i.e., Web service tag recommendation), and (3) Web services clustering. In the first module, WSDL documents and tags of Web services are crawled from the Internet. Similar to the work in [4], we extract five important features from WSDL documents, i.e., Content, Type, Message, Port, and Service Name. Thus, we can obtain tagging data and five kinds of feature data for Web service clustering after data pre processing. These data are sent to the module of Web service clustering, in which feature-level similarities and tag-level similarities are integrated into a global similarity to cluster Web services. Since the data pre processing, WSTRec, and Web service clustering modules are executed off-line, the efficiency is not a big concern, whereas the accuracy is more important

When employing tags to help Web service clustering, there are two main problems which limit the effectiveness of tag data in Web service clustering: (1) uneven distribution and (2) noise. To address these problems, we introduce WSTRec strategy, in which tag co-occurrence, tag mining, and relevant metrics are adopted. Specially, WSTRec is implemented to do some process on the tagging data. Thus, the module of WSTRec can also be treated as data preprocessing.

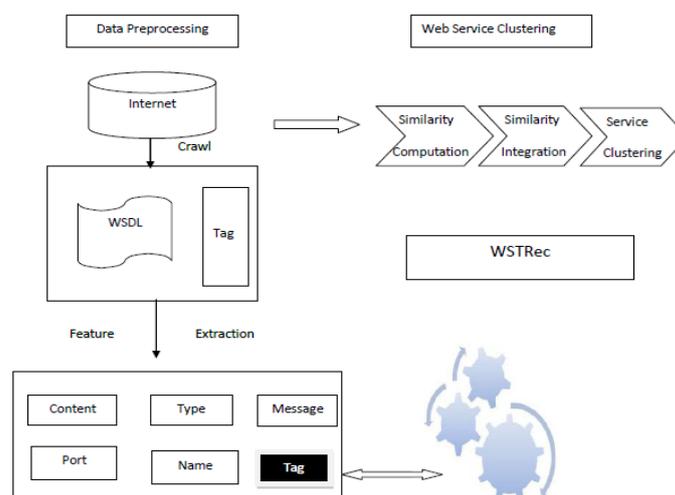


Fig. 1 architecture of the proposed Web service clustering approach.

IV. WEB SERVICE CLUSTERING

Web service clustering is the process of feature extraction and the corresponding feature-level similarity computation, including tagging data and five features extracted from a WSDL document then similarity integration and service clustering is described.

A. Feature Extraction and Similarity Computation

Five features (i.e., Content, Type, Message, Port, and Service Name) are extracted from a Web service's WSDL document. These five features and tags are employed to cluster Web services. In the following, we describe the detailed process of feature extraction and the corresponding similarity computation.

1) *Content*: A WSDL document, which describes the function of Web service, is an XML style document. Therefore, we can use IR approaches to extract a vector of meaningful content words which can be used as a feature for similarity computation. Our approach for building the content vector consists of four steps:

1. Building an original vector. In this step, we split the WSDL content according to the white space to produce the original content vector.
2. Suffix Stripping. Words with a common stem will usually have the same meaning, for example, connect, connected, connecting, connection, and connections all have the same stem connect [5]. For the purpose of convenient statistics, we strip the suffix of all these words that have the same stem by using a Porter stemmer [32]. After the step of suffix stripping, a new content vector is produced, in which words such as connected and connecting are replaced with the stem connect.
3. Pruning. In this step, we propose to remove two kinds of words from the content vector. first kind is XML tag. For example, the words s:element, s:complexType, and wsdl: operation are XML tags which are not meaningful for the comparison of content vectors. As the XML tags used in a WSDL document are predefined, it is easy to remove them from the content vector. Content words are typically nouns, verbs, or adjectives
4. Refining. Words with very high occurrence frequency are likely to be too general to discriminate between Web services. After the step of pruning, we implement a step of refining, in which words with too general meanings are removed. Clustering based approaches were adopted to handle this problem in some related work [4,5]. In this paper, we choose a simple approach by computing the frequencies of words in all WSDL documents and setting a threshold to decide whether a word has to be removed.

2) *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 types. Due to different naming conventions, the name of a parameter is not always a useful feature, whereas the type attribute which can partially reflect the service function is a good candidate feature.

3) *Message*: Message is used to deliver parameters between different operations. One message contains one or more parameters, and each parameter is associated with a type as we discussed above. Message definition is typically considered as an abstract definition of the message content which defines the name and type of the parameter contained in the message.

4) *Port*: The portType element combines multiple message elements to form a complete one-way or round-trip operation, which contains some operations (due to space limitation, we only list one operation in this portType). As the portType consists of some messages, we can get the match result of portType according to the match result of messages.

5) *Service name*: As the service name (sname) can at least partially reflect the service function, it is also an important feature in WSDL document. Before computing the sname-level similarity, we first implement a word segmentation process to service name. For example, the service name SendCustomForm can be separated into three words: Send, Custom, and Form. A simple version of word segmentation is to split the service name according to the capital letters. However, its performance is not satisfied due to different naming conventions. In this paper, we first use

the capital letters to split the service name and then manually adjust the final result. After the process of word segmentation, SNames1, the name of service s1, can be presented as a set of words.

6) *Tag*: The tagging data of Web services describe their function 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. Similar to above five features extracted from the WSDL document, tagging data are also treated as a feature in the process of similarity computation.

B. Similarity integration and service clustering

In the above subsection, we calculate feature-level similarity and tag-level similarity among Web services, i.e., Simcon, Simtype, SimmesImport, Simsname, and Simtag. In this section, we integrate them into a global similarity for the purpose of Web service clustering. Before similarity integration, it should be noted that the WSDL document is created by service providers and tagging data reflect users' knowledge. Thus, these two kinds of similarities should be treated separately. In this paper, we first merge the similarities of features extracted from WSDL documents to derive a WSDL-level similarity and then integrate WSDL-level similarity and tag-level similarity into a global one.

V. WSTRec

A. Tag Mining

In this case, we have to mine some tags from the textual features of the services first and then use a tag recommendation strategy to improve the quality of the tagging data.

B. Tag Co-occurrence

Here tag co-occurrence is calculated using Jaccard coefficient

C. Semantic Relevance

Relevance is calculated using candidate tag

VI. CONCLUSION

In this paper, we propose to utilize tagging data to improve the performance of traditional WSDL document-based Web service clustering for the purpose of more accurate Web service discovery. In our proposed clustering approach, Web services are clustered according to the global similarities among services, which are calculated based on tagging data and five features extracted from the WSDL document.

Furthermore, we propose a hybrid Web service tag recommendation approach WSTRec to improve the performance limited by the uneven tag distribution and noisy tags. Tag co-occurrence, tag mining, and semantic relevance are adopted in WSTRec. Specifically, tag mining technique is employed to mine some initial tags for Web services with no tags. Furthermore, we will conduct more research in utilizing social network information to facilitate Web service mining.

ACKNOWLEDGEMENT

Any work would not be complete without the mention of the people whose guidance and encouragement lead to the development of the work. We consider ourselves privileged to express gratitude and respect towards all those who guided throughout my work. We extend our sincere thanks to the HOD, ISE Dept and the management for their co-operation for providing a very good infrastructure and all the kindness forwarded to us in carrying out this paper work in college.

REFERENCES

- [1] Al-Masri E, Mahmoud QH (2008) Investigating web services on the world wide web. In: Proceedings of international World Wide Web conference, pp 795–804
- [2] Nayak, Richi (2008) Data mining in web service discovery and monitoring. *IntJWebServ Res* 5(1):62–80
- [3] Lim SY, Song MH, Lee SJ (2004) The construction of domain ontology and its application to document retrieval. *Lect Notes ComputSci* 3261:117–127
- [4] Elgazzar K, Hassan AE, Martin P (2009) Clustering wsdL documents to bootstrap the discovery of web services. In: Proceedings of international conference on Web services, pp 147–154
- [5] LiuW, WongW (2009) Web service clustering using text mining techniques. *Int J Agent-Oriented SoftwEng* 3(1):6–26
- [6] Lipczak M, Hu Y, Kollet Y, Milios E (2009) Tag sources for recommendation in collaborative tagging systems. *ECML PKDD DiscovChall* 497:157–172
- [7] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of international conference on very large data bases, pp 487–499
- [8] Wu Z, Deng S, Li Y, Wu J (2009) Computing compatibility in dynamic service composition. *Int J KnowlInfSyst* 19(1):107–129
- [9] Hu C, ZhuY, Huai J, LiuY, NiLM (2007) S-club: an overlay-based efficient service discovery mechanism in crown grid. *Int J KnowlInfSyst* 12(1):55–75

- [10] Liang QA, Chung J-Y, Miller S (2007) Modeling semantics in composite web service requests by utility elicitation. *Int J KnowlInfSyst* 13(3):367–394
- [11] Wu J, Chen L, Xie Y, Zheng Z (2012) Titan: a system for effective web service discovery. In: *Proceedings of international conference companion on World Wide Web*, pp 441–444
- [12] Agarwal S, Studer R (2006) Automatic matchmaking of web services. In: *Proceedings of international conference on Web services*, pp 45–54
- [13] Klusch M, Fries B, Sycara K (2006) Automated semantic web service discovery with owls-mx. In: *Proceedings of international conference on autonomous agents and multiagent systems*, pp 915–922
- [14] Benatallah B, Hacid M, Leger A, Rey C, Toumani F (2005) On automating web services discovery. *Int J Very Large Data Bases* 14(1):84–96
- [15] Zhang Y, Zheng Z, Lyu MR (2010) Wsexpress: a qos-aware search engine for web services. In: *Proceedings of international conference on Web services*, pp 91–98
- [16] Dong X, Halevy A, Madhavan J, Nemes E, Zhang J (2004) Similarity search for web services. In: *Proceedings of international conference on very large data bases*, pp 372–383
- [17] Hu S, Muthusamy V, Li G, Jacobsen HA (2008) Distributed automatic service composition in large-scale systems. In: *Proceedings of distributed event-based systems conference*, pp 233–244
- [18] Liu F, Shi Y, Yu J, Wang T, Wu J (2010) Measuring similarity of web services based on wsdl. In: *Proceedings of international conference on Web services*, pp 155–162