Research on Service Dependency Mining Technology Based on Tag Extension and Bipartite Graph

Liangjun Zhang¹,a Xiuguo Zhang²,b * Zhiying Cao³,c

¹Dalian Maritime University
Dalian, China
²Dalian Maritime University
Dalian, China
³Dalian Maritime University
Dalian, China

a973060149@qq.com bzhangxg@dlmu.edu.cn czysophy@dlmu.edu.cn

ABSTRACT: Due to the rapid development of service portfolio model on network, service computing has become a hot topic in recent years. It is important for both service management and service recommendation to clarify the dependencies between services. Tag, as a widely used network information resource in the current service field, can intuitively reflect the function, purpose and other attributes of the service, and it has become a common resource in the field of service discovery and combination. Tag sparsity, however, will reduce the accuracy of service relationship analysis. Therefore, this paper proposes a service dependency mining method based on tag extension and bipartite graph. By extending the existing tags and abstracting the combination of service and the relationship between the extension tags into a bipartite graph, dependency propagation mechanism in bipartite graph is applied to improve the comprehensiveness and accuracy of the service dependency mining. The programmable Web APIs and Mashups are used to show the validity of the service dependency mining method.

CCS CONCEPTS
• Networks ➝ Network services • Web services ➝ RESTful web services • Association rules • Collaborative and social computing systems and tools ➝ Social tagging systems

1. INTRODUCTION

Service-oriented computing (SOC) is a new distributed computing model [1]. In the age of Web 2.0, there are more and more APIs on the web being described, published, searchable, and invoked. And value-added services are provided to users as a network resource invoked in the form of multiple mashup combinations.

In this situation, it is particularly important to mine the dependencies between services quickly and accurately. In view of high multiplex used rate of some services, the method of analyzing service dependency from the historical perspective of service combination will often ignore some infrequently used services. And the service tag, as a widely used network information resource which can visualize service function and use, can improve the service mining coverage. Considering the possibility of inaccurate relationship mining caused by sparse service tags, this paper expands the original service tags to improve the accuracy of service dependency mining. As a common method of service
recommendation, bipartite graph can spread tag dependencies to other tags with dependency propagation mechanism [11, 12] and further explore potential dependencies among some tags.

Therefore, on the basis of extending service tags, bipartite graph is used in this paper to mine the dependencies between service tags, and then the dependencies between services can be obtained according to the mapping relationship between tags and services.

2. RELATED WORKS

2.1 Service Dependency Mining Technology Based on Software System Execution Information

According to the data on which the service dependency discovery technology is based, the service dependency discovery technology is divided into four categories: the data based on monitoring data, the data based on network traffic, the data based on system log and the data based on request tracking [3].

Yin Jianwei et al. [4] collected the number of TCP/UDP collections of each component in the distributed software system and eight monitoring indexes reflecting the resource usage, and found the service dependency by analyzing their similarities. Kitajima S et al. [5] discovered the correlation between logs through statistical methods, and inferred the dependency corresponding relationship between logs and services.

Such methods collect large amounts of online data and cover a limited number of services, making it easy to ignore services that are not frequently invoked.

2.2 Service Dependency Mining Technology Based on Service Abstract Information

Such methods are often based on service description documents, such as BPEL and WSDL. Feng Renjun et al. [6] determined the dependencies between member services by analyzing the WS-BPEL process. Chen Shizhan et al. [7] abstracted Web Service into a conceptual model of three levels, namely Service, Operation and Parameter by WSDL, and mining the dependencies between services by matching input parameters and output parameters among services. In addition, a composition analysis method that follows Web semantics is also a common method [14], which applies system reasoning and inference method to ontology concept to discover composition [2].

Semantic ontology can be used to accurately mine the dependencies between services. But the complexity of ontology construction makes semantic-based mining technology meet the bottleneck in practical application. Although the idea of the algorithm based on the matching of service parameters is simple, it needs the pairwise comparison between services. The efficiency of the relationship mining is reduced by the repetitive reasoning.

3. SERVICE DEPENDENCY MINING

In this paper, tag extension, tag weighting and tag dependency analysis based on bipartite graph are studied. The main process framework of service dependency mining in this paper is shown in figure 1.
3.1 Service Tag Extension Based on TF-IDF

The tag can reflect the function, use and other attributes of the tagged resource. In the Web services domain, Web APIs and Mashups are also tagged accordingly [8]. Considering the sparsity and diversity of service tags will affect the accuracy of association relation mining between tags. In this paper, frequency inverse document frequency (TF-IDF), a technical term based on keywords, is used to extract keywords from service description documents and expand service labels. The following relevant formulas of TF-IDF algorithm are defined according to reference [17].

Definition 1: TF (Term Frequency). It is the word Frequency, which refers to the Frequency of a keyword in the current document, and its calculation formula is as follows.

\[ TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,k}} \]

Where TF\(_{i,j}\) is the frequency of keyword j in document i. \(n_{i,j}\) is the number of times of the keyword j appears in document i. \(\sum n_{i,k}\) is the number of words that appear in document i.

Definition 2: IDF (Inverse Document Frequency). It is the inverse of document Frequency DF (the Frequency of a word in the whole library dictionary). The calculation formula is as follows.

\[ IDF_i = \log \frac{|D|}{|\{ j : t_i \in d_j \}| + 1} \]

Where \(IDF_i\) is the inverse document frequency of the word i. |D| is the total numbers of files in the corpus. |\{ j : t_i \in d_j \}| is the total numbers of documents with the word i, +1 is just to keep the denominator from getting 0.

Definition 3: TF - IDF

Considering the situation that when a word appears more frequently in a document, but its IDF value is lower. We still cannot ignore the possibility that it is a keyword. Therefore, in order to judge whether a word is a keyword more accurately, the value of TF and IDF should be taken into account at the same time. Thus, the following formula is obtained.

\[ TF-IDF_{i,j} = TF_{i,j} * IDF_i \]

Where the greater the TF-IDF value is, the greater the probability is that this word will be a keyword.

Before the TF-IDF algorithm is applied to extract keywords from the Web API description document, the API description document data needs to be preprocessed. The specific processing methods are as follows:

1) Remove stop words, punctuation marks and other special characters.
2) Participle processing.
Due to the different description habits of different developers, synonyms will be used to tag the same service. In addition, the same word may have different forms of expression. Therefore, the extended tags need to be further processed to obtain high-quality data and reduce experimental errors:

1) Morphology reduction.
2) Synonym unification.

This section uses TF-IDF to calculate the words in the API description documents to get the TF-IDF value of each word, and sorts the words from high to low according to the value size. Finally, according to the threshold for the number of keywords, we select the first few keywords with the highest TF-IDF value as the new tags to extend the Web API tags.

3.2 Acquisition of Mashup-Tag Association Relationships Based on Tag Co-occurrence
Co-occurrence refers to the phenomenon that features of the same or different items appear together. Based on this phenomenon, further studies on entity clustering, association mining, entity recommendation and other aspects can be carried out [15, 16].

In this study, the Mashup-Tag association relationships can be obtained with the tag co-occurrence distribution in the APIs and the API co-occurrence distribution in the mashups. Tag co-occurrence relationship is shown in Figure 2.

![Figure 2. Tag Co-occurrence Relationship Instance](image)

This paper draws on the method in reference [8] to obtains the Mashup-Tag association relationship through the "Mashup-API" relationship and the "API-Tag" relationship.

In this study, the "API-Tag" matrix $Q_{mn}$ is constructed, in which, API is the row vector and tag is used as the column vector. If a certain API has a tag $t_j$, the value is 1; otherwise, it is 0. Similarly, Mashup-API matrix $R_{zm}$ is constructed, in which mashup is used as row vector and API is used as column vector, and then the Mashup-Tag correlation matrix is obtained according to formula (4).

$$M_{mn} = R_{zm} Q_{mn} = (m_1, m_2, \cdots, m_p)^T$$ (4)

Where the row vector of $M_{mn}$ represents the mashups, the column vector represents the tags, and the element value $m_{ij}$ represents the number of times the tag $t_j$ appears in the mashup composite $m_i$, which reflects the weight of the tag $t_j$ in the mashup composite service tag set.

3.3 Service Tag Dependency Mining Based on Bipartite Graph

3.3.1 Construction of Weighted Bipartite Graph

Definition 4: Bipartite Graph. It is a special model in graph theory. The vertices in the graph can be divided into two disjoint subsets, and the edges connect two nodes from different classes to represent the association between nodes.

Based on the Mashup-Tag incidence matrix obtained in the previous section, the mashup set and tag set are considered as two node sets of a bipartite graph. If a mashup has an association relationship with a tag, an edge is established between them, and the value of the matrix element represents the weight of the edge. The bipartite graph mentioned above is shown in Figure 3.
3.3.2 Tag Dependency Propagation Based on Bipartite Graph

**Definition 5:** The adjacency matrix of a bipartite graph. It is used to represent the information in the bipartite graph. The elements of the adjacency matrix are defined as follows.

\[
a_{i,j} = \begin{cases} 
1 & (e_{i,j} \in E); \\
0 & (\text{other}), 
\end{cases}
\] (5)

Where \(m\) and \(n\) respectively represent the number of two types of nodes in the bipartite graph.

**Definition 6:** Weight Matrix. It refers to the importance of the connection between nodes at both ends of an edge. The weights are defined as follows.

\[
W_{ij} = x(0 \leq x \leq 1)
\] (6)

The value of the weight value \(x\) in the formula (6) can be obtained according to the method described in 3.2.

Dependencies are propagated between tags based on a bipartite graph. The propagation process is expressed by formula (7) and (8).

\[
mt[i][j] = \frac{a[i][j] \times w[i][j]}{\sum_{j=1}^{n} w[i][j]}
\] (7)

\[
t[i][j] = \sum_{i=1}^{m} \left( m[i][j] \times \frac{w[i][j]}{\sum_{j=1}^{n} w[i][j]} \right)
\] (8)

Where formula (7) is the dependency between tags and mashups after the first resource flow allocation; Formula (8) is the dependency between tags after the second resource flow allocation, where the element value is the degree of the dependency between tags. Finally, we use the Levenshtein algorithm mentioned in the comparison experiment in reference [1] to complete the correspondence between tags and services, where the matching degree in formula (9) has a certain impact on the quality of service dependency mining.

\[
\text{match degree}(t, s) = \frac{N_{t_s}}{N_{s_t}}
\] (9)

Where \(N_{t_s}\) represents the number of services that the tag matches to, and \(N_{s_t}\) represents the total number of services that have a similar relationship with the tag.

### 3.4 Algorithm Description

The pseudocode of this algorithm is described as follows.

**Algorithm 1: Service Tag Extension Algorithm Based on TF-IDF**

**Input:** service description document list collection \(SD = \{s_{d1}, s_{d2}, \ldots, s_{dn}\}\)

**Output:** the expanded set of tags for the service \(ST = \{s_{t1}, s_{t2}, \ldots, s_{tm}\}\)

1: **for** each service description document in \(SD\)
2: remove stop words and special characters in \(s_{di}\), \(s_{di}$$\in$$SD$$
3: segmentation in \(s_{di}\)
4: extract keywords with TF-IDF in $s_{di}$
5:  Keywords for morphological reduction, synonym normalization
6:  Store the extension tags in $ST=\{s_{t1}, s_{t2}, \ldots, s_{tm}\}$
7: end for
8: return $ST$

Algorithm 2: Mashup-Tag Weighted Association Relation Acquisition Algorithm

Input: Mashup-API collection list $L_{m}$ and API-Tag collection list $L_{at}$
Output: Mashup-Tag weighted relational matrix $M_{zn}$
1: Initialize Mashup-API matrix $R_{zm}$ and API-Tag matrix $Q_{mn}$
2: get $M_{zn}$ by formula (4)
3: return $M_{zn}$

Algorithm 3: Service Tag Dependency Mining Algorithm Based on Bipartite Graph

Input: Mashup-Tag weighted relational matrix $M_{mn}$
Output: Dependencies between services
1: $M_{zn}$ is initialized to weighted bipartite graph $MT_{G}$
2: Initialize the adjacency matrix $A_{mn}$ and the weight matrix $W_{mn}$; define Mashup-Tag dependency matrix $MT_{mn}$ and Tag-Tag dependency matrix $TT_{nn}$
3: for each mashup in $W_{mn}$
4: for each tag node about the mashup
5: get $mt[i][j]$ by formula (8), then Assign it to $MT_{mn}$
6: Keywords for morphological reduction, synonym normalization
7: Store the extension tags in $ST=\{s_{t1}, s_{t2}, \ldots, s_{tm}\}$
8: end for
9: end for
10: for each tag node in $W_{mn}$
11: for each mashup node about the tag
12: get $tt[i][j]$ by formula (9), then Assign it to $TT_{mn}$
13: end for
14: end for
15: get the dependencies between services with $TT_{mm}$ and the Mapping between services and tags
16: return Dependencies between services

4. EXPERIMENT AND ANALYSIS

4.1 The Data Set
This dataset consists of 1042 mashups, 1007 APIs and 1574 tags, all crawled from Programmable Web in October, 2018. The Web API description data includes service name, tag, description document, and the mashup description data includes the name, tag and included API.
4.2 The Experimental Contrast

In order to evaluate the performance of Web API dependency mining proposed in this paper, this experiment was evaluated according to the commonly used evaluation indexes in data mining: Precision Rate and Recall Rate of their comprehensive evaluation. In this paper, the matching degree between tags and services will have some influence on the mining of service dependencies. Therefore, the comparative experiment is mainly compared with different experiments from two perspectives:

1) At the perspective of tag dependency: the service dependency mining algorithm based on extended tags and weighted bipartite graph (ETBG method) proposed in this paper was compared with the mining algorithm (TAR method) proposed in reference [1], which used association rule mining technology of social annotation to predict mashup patterns.

Ten groups of tags were randomly selected, with 10 tags for each group. The total numbers of binomial set dependencies between the mined tags were the recall ratio of the two methods. The precision was measured by judging whether the binomial set dependencies existed between mined tags or not. Since both methods will dig out the potential dependencies of tags to varying degrees, we adopt manual judgment for such data. If the combination is reasonable, it is right. The comparison results are shown in Figure 4 and Figure 5.

![Figure 4. Recall Rate Comparisons For The First Experiment](image1.png)

![Figure 5. Precision Rate Comparisons For The Second Experiment](image2.png)

It can be seen from the above experiments that, although the mining method proposed in this paper is not as good as the method in reference [1] in individual test groups, the method in this paper is obviously better than TAR method in terms of precision rate and recall rate on the whole.

2) At the perspective of service dependency: considering the influence of service matching degree on evaluation indexes, this paper, based on ETBG method, obtains the service relationship mining indexes with three different matching degrees and compares them with the service dependency analysis algorithm (LFH) based on portfolio history in [10].

Ten groups of services were randomly selected, with 10 services in each group. The total numbers of binomial dependencies between the mined services was used as a recall for both methods. Precision was measured by determining whether the correct binomial set dependencies existed in the mined services. For the mined potential dependencies, the rationality was still judged manually. The comparison results are shown in Figure 6 and Figure 7.
As can be seen from the above experiments, the recall rate and precision rate of the method described in the literature [10] are similar to that of this experiment with low matching degree. When the approximate matching degree is improved, the evaluation index is greatly improved compared with the LFH method.

To sum up, the method proposed in this paper has obvious advantages in terms of recall rate and precision rate of service dependencies, and it can effectively mine the dependencies between services.

5. CONCLUSION
In order to mine service dependencies in the service ecosystem, this paper proposes a service dependency mining method based on tag extension and bipartite graph. Firstly, TF-IDF method is used to extend tags, avoiding the influence of tag sparsity on mining results. Then the dependency propagation idea in the bipartite graph is used to mine the dependencies between tags. This method improves the precision rate and recall rate of service dependency mining to a certain extent, so as to alleviate the cold start problem in the service recommendation system and avoid the service recommendation limitations caused by the long tail effect of network service.

ACKNOWLEDGMENTS
This work is supported by the National Natural Science Foundation of China (Grant No. 51479021, 51679025) and the Fundamental Liaoning Provincial Natural Science Foundation of China(Grant No.20170520196)

REFERENCES
[1] Papazoglou MP. Service-oriented computing:Concepts, characteristics and directions. Information System Journal, 2003,[2]:3-12.
[2] Goarany K, Kulczycki G, Blake M B. Mining social tags to predict mashup patterns//Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents. Toronto, Canada, 2010:71-78.
[3] Yong Y. Introduction to service dependent discovery. https://mp.weixin.qq.com/s/JO73Mn0ATPwe3L_LieK5w, 2018.(in Chinese)
[4] Yin J, Zhao X, Tang Y, et al. CloudScout: a non-intrusive approach to service dependency discovery[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(5): 1271-1284.
[5] Kitajima S, Matsuoka N. Inferring Calling Relationship Base on External Observation for Microservice Architecture[C]/International Conference on Service-Oriented Computing. Springer, Cham, 2017: 229-237.

[6] Renjun Feng, Wei Song, Xiaoming Man. Web service dependency analysis based on ws-bpel process [J]. Computer engineering and application, 2014, 50(02):25-30+74. (in Chinese)

[7] Zhanshi Chen, Zhiyong Feng, Hui Wang. Service relationship and its application in service-oriented computing [J]. Acta computica sinica, 2010, 33(11):2068-2083. (in Chinese)

[8] Liu Jian-Xun, Shi Min, Zhou Dong, Tang Ming-Dong, Zhang Ting-Ting. Topic Model based Tag Recommendation Method for Mashups. Chinese Journal of Computers, 2017, 40(2):520-534.

[9] Xiang Chen, Xiuzhen Qiu. Collaborative recommendation research based on tag similarity [J]. Chinese journal of science and technology, 2013, 8(10):974-980. (in Chinese)

[10] Weifeng Pan, Bing Li, Bo Shao, et al. Research on automatic service classification and recommendation methods based on software network[J]. Acta computica sinica, 2011, 34(12):2355-2369.

[11] Zhou T, Ren J, Medo M, et al. Bipartite network projection and personal recommendation[J]. Physical Review: E 2007, 76(4):1-7.

[12] Shuiguang Deng, Jianwei Yin, Ying Li, et al. Semantic Web services discovery method based on binary graph matching. Acta sinica, 2008, 31(8):1364-1375.

[13] A. Fernandez, C. Hayes, N. Loutas, V. Peristeras, A. Polleres, and K. Tarabanis, “Closing the Service Discovery Gap by Collaborative Tagging and Clustering Techniques,” Proceedings of the 7th International Semantic Web Conference. ISWC’08, 2008, pp. 115-128.

[14] J. Li, S. Chen, Y. Li and Q. Zhang, “Semantic Web Service Automatic Composition Based on Service Parameter Relationship Graph,” 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, 2011, pp. 1773-1778.

[15] H. Li, J. Liu, B. Cao, M. Tang, X. Liu and B. Li, "Integrating Tag, Topic, Co-Occurrence, and Popularity to Recommend Web APIs for Mashup Creation," 2017 IEEE International Conference on Services Computing (SCC), Honolulu, HI, 2017, pp. 84-91.

[16] S. Seo, J. Kim and L. Choi, "Semantic hashtag relation classification using co-occurrence word information," 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, 2017, pp. 860-862.

[17] B. A. Kuncoro and B. H. Iswanto, "TF-IDF method in ranking keywords of Instagram users' image captions," 2015 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, 2015, pp. 1-5.