Enriching Web Services Tags to Improve Data-Driven Web Services Composition

Nahid Dara[1,2] and Sima Emadi[2,*]

[1]*Diabetes Research Center, Shahid Sadoughi University of Medical Sciences, Yazd, Iran*
[2]*Department of Computer Engineering, Yazd Branch, Islamic Azad University, Yazd, Iran*
E-mail: daranahid@gmail.com; emadi@iauyazd.ac.ir
[*]*Corresponding Author*

Received 04 December 2020; Accepted 09 December 2020;
Publication 09 March 2021

Abstract

Due to the large number of existing services and complexity of manual composition, automatic service composition is provided to enable automatic search of the service compositions for the given queries. Many solutions for automatic service composition have been developed, including integer programming, graph planning, artificial intelligence, and so on in this paper, a heuristic method is proposed to improve the data-driven composition of web services by enriching tags based on tags semantic. To do so, firstly, useful information on web services is collected from various sources and is turned into collections of tags. In the next step, using the hierarchical clustering algorithm, the tags are clustered based on semantic similarity. Thereafter, for services which do not have enough tags, enrichment of the tag is carried out and finally, using an algorithm, composition solutions based on QoS parameters are extracted, which can formulate user targets or even provide potential compositions. Moreover, a series of tests were conducted on the web services, which validate the efficiency of the proposed approach. The experimental results confirm the effectiveness of the

Journal of Web Engineering, Vol. 20_2, 327–358.
doi: 10.13052/jwe1540-9589.2025

proposed service composition method and high quality of tag enriching strategies.

Keywords: Service composition, clustering, enrichment tag, data-driven, web service, WSDL.

1 Introduction

Web services are a new generation of web applications that are self-contained, self-describing, and modular, and can be published, located, and invoked across the web [1]. The composition of web services and web applications depends on the establishment of coordination among a number of existing web services to provide a richer service composition also to achieve some of the user's requirements which cannot be met with a single service [2]. With the introduction of a new requirement, the service composition is considered as an essential requirement for advancement of users' objectives to meet demands and also provide benefits from several service capabilities. The most important challenges to web services composition include the large number of services, the need to frequently update the services, and creation of services by various manufacturers with different models and goals.

On the other hand, since situational applications developers may be non-professional programmers or even end-users, the challenge is that they tend to represent their intended goals simply and quickly so as to find suitable services that can respond to their requests. They usually do not understand, or simply neglect, technical details such as syntactic rules, semantics, and message mediation. Therefore, the process of designing and developing the situational application requires not only abstraction of individual services, but also a much broader view on evolving sets of services that can potentially be combined with this service. For this reason, the data-driven composition technique should be used for situational applications. Previous studies also have clearly demonstrated that a dynamic method which can be helpful in solving this problem is the data-driven composition through which users can quickly search their services and achieve their goals by entering multiple tags or keywords [3]. In situational service composition users use tags to represent their desired goals and find relevant services. Throughout the development process, the user selects or inputs some tags to select the services. Selected services either use these tags as inputs, or generate these tags as outputs. Since the input and output of services are associated with tagged data, Services can be connected to create data flow and Developers can search their

goals by means of tags, and compose selected services based on data driven approaches.

Nowadays, tags are widely used as a non-hierarchical keyword or an assigned item to information in popular web 2.0s, including Flickr and Del.icio.us. Tags make searching easier and, to a considerable extent, enrich the metadata to describe web resources [3]. They are assigned to web services either automatically or manually. Tags are semantic and syntactic information in the form of important verbs, nouns and phrases that they extract from WSDL documents and related descriptions [4]. But the problem with real-world scenarios is that most web services do not have enough meaning tags and don't provide sufficient enough information to reveal the relationships among services.

A review of existing literature shows that the current research on web service tagging and service composition using tags, has certain limitations. For instance, uneven distribution of tags of web services or noisy and misspelled tags created by end users can decrease the effectiveness of the method. In fact, some web services do not have enough tags with semantic and syntactic information to select the candidate service in the composition process and therefore may not be well discovered and selected in the service composition process [5]. So, tag enrichment becomes an important issue. In the enrichment process, tags of similar services are assigned to a service that does not have a tag or does not have enough tag information. Another problem is that in most of the existing tag recommendation approaches [4, 6], tag semantics is not perfectly considered. Moreover, most studies on web service composition have focused on the functionality of web services, QoS of composed service and search space reduction; as a result, semantic and the syntactic information defined in the tags have not been investigated extensively [7, 8]. Also, the number of studies on tag-based service composition is very limited while tag-based search can improve search performance of a single web service and consequently improves the quality of web service composition. While considering the semantic similarity between words (for example, similarity between car and automobile) helps to offer more and more suitable candidate services. The use of tag semantics in web service composition has a great impact on accuracy and speed.

Therefore, in this research, an effective method for dynamic composition of web services using enrichment of extracted tags based on semantic similarity between them is presented. In this method, using semantic similarity between tags, tags with insufficient information are enriched to help in better selection of services. Then in the service composition process, a forward

algorithm is used to compose the services that are closer to the user's request in terms of QoS parameters.

The key contributions in this paper are:

1. A novel method has been proposed to enrich tags for web services with insufficient tags that can improve discovery of services.
2. Proposed method has been searched services using tags and without the need for technical information about the services.
3. In addition to clustering web services based on the similarity of WSDL files and the co-occurrence of tags, the semantic similarity of tags is also considered.
4. The number of proposed services has been increased by tag enrichment.
5. An efficient algorithm based on forward search algorithm is presented to dynamically combining web services with enriched tags.
6. In service composition, geographical distance QoS parameter has been used to provide the least network delay in the use of web services.

This paper is organized as follows. The next section briefly discusses related works and research background. Section 3, presents an example to better understand the research. Section 4 introduces the proposed method and algorithms, and finally in Section 5, the proposed method is evaluated and the results of the analyses are discussed. The paper ends with summary and conclusion.

2 Related Works

Given that this study is a data-driven composition of web services and that the service composition is supposed to be done using tagging, literature reviews have been categorized in two areas: (1) service discovering and service composition, and (2) web service clustering, tagging, and tag extraction.

2.1 Studies Related to Service Discovering and Service Composition

Based on the previous studies, service composition methods can be categorized as follows: workflow-based methods, artificial intelligence planning methods, graph theory methods, Petri Net-based approach, goal-oriented compositions, data-driven composition, QOS-Aware (Quality of service-Aware), etc. Some of these categories have been investigated in this paper.

Ma et al. proposed a novel service composition method called TAD (Transformation-Annotation-Discovery) which can discover and compose

real-world RESTful services. At first, TAD converts OAS documents into graph structure and then to simplify the service retrieval, semantic annotation is performed on each graph node using Latent Dirichlet Allocation and WordNet. Next, according to the user's requirements, the service combination is performed by two modules; service discovery chain and pipeline-based composition which finds services that semantically fit the user's requirements. Their results show that the proposed method has a good performance according to precision metric [9]. Renzis et al. proposed a solution for Service Selection by applying Case-based Reasoning, in which the similarity between a pair of cases is measured through three similarity functions. Their main goal was to take the knowledge obtained from successive service selections as a set of cases in the form of problem-solution pairs. They defined certain case representation capturing information in Web Services functional descriptions and provided three implementations for the similarity function, concerning structural and semantic aspects from functional service descriptions. Their results showed that the proposed approach achieves a high precision and recall with all the three alternative implementations of the similarity function [10]. Liu et al. provided a tag-based data-driven composition for web applications, which is a dynamic approach where in service composition is done as soon as the tag is searched. They used the tags extracted from search engines and WSDL files to explain service capabilities in form of a tag. A hierarchical clustering algorithm was used for clustering the tags and a heuristic graph-based planning algorithm with polynomial time complexity was proposed to achieve services composition. The service composition was performed using two search algorithms; depth-first and backward search. Their main focus was on capabilities and functions of services; hence, they did not take into account quality parameters [3]. Shuiguang Deng et al. addressed quality-based and high-scale services composition. The composition algorithm is based on combination of depth-first and back track search methods, both of which can run in parallel. Their method can successfully create K web services with the highest quality, accuracy, and performance. To evaluate the efficiency, accuracy, response time, and scalability, different factors have been considered including the number of services, the number of concepts, depth of the solution, and the number of presented solutions [11]. Guobing Zou et al. provided an automatic method for service composition using a state-of-the-art AI planner. They considered web services composition as a web service composition planning problem. Unlike the traditional methods of service composition which parse and traverse a web service repository several times, their proposed method

converts a web service repository into a planning domain in PDDL just once. It is then re-generated when the web service repository is changed. This reduces the response time and improves scalability of the service composition problem [2]. Song et al. presented a goal-oriented approach for composing adaptive services with programming. They created a prototype that supports dynamic service composition in a smart working environment. The system is capable of composing services in the smart office domain to help users attain their goals. In this research, users submit their requests in non-technical terms. Service composition programming was based on a hierarchical task network and service discovery process was automatic [7]. Tan Weiten et al. used a data-driven method for service composition. Based on the relationship between business domain data and service domain data, they created additional data mediations. Based on these data and relationships, a Petri-net based approach was developed to compose services; this improved communication between the service domain and the business domain, and delivered hybrid services [8]. Liu et al. provided a method for combining data-driven mashup based on tag-based semantic rules. Their most important task was to extract meanings of common tags and link them with inputs and outputs of the program. They used a planning technique to extract potential compositions. The disadvantage of their method was that they supported only personal mashups, and could not support the complex composition logic in enterprise systems. In addition, inappropriate distribution of web site tags limited the composition quality [12]. Eric Bouilette et al. presented a new method for combining follow-based information processing systems. Software design and component development were facilitated through tag-based descriptions. They used top-down and bottom-up composition designs [13]. Dandan Wang et al. presented a genetic algorithm-based approach for composing web services in a distributed cloud environment. Experimental results showed that the use of the genetic algorithm in cloud computing minimized time and cost of web browsing [14]. Fanjang and Syu used a genetic algorithm for semantic-based automatic service composition. They considered functionality and quality of the services simultaneously [15]. Gao et al., proposed a novel model, "Extended Dependency-Compensated Service Co-occurrence LDA" (EDC-SeCo-LDA) to identify the evolutionary characteristics of service ecosystem at the topic level. Since, topic dependencies reveals the latent trend of service composition patterns in the service ecosystem, So, they designed an algorithm to calculate the topic dependencies based on topic-service distribution based on leveraging the information of published time of mashups [16].

Vairetti et al. considering that the behaviour of composed services is highly complex or ignored in the composition process, proposed a method that extracts complex composed service behaviour semantics. They are exploit in their proposed composition method service, signature and semantic annotations along with rules to identify simple and complex control-flow patterns between services [17].

A graph of the services that are connected through such patterns is then created to represented the behavioural semantics of a composed service.

Studies have shown that many of web service composition approaches focused only on the functionality of services and quality parameter. In addition, just a few of them were data driven and tag-based. Also, they focus on reducing search space and composition time and don't consider the information contained within the service. In fact, they assume the services have the information they need.

2.2 Studies on Clustering and Tagging Web Services

Shi et al. proposed a new topic-sensitive method for mashup tag recommendation based on Factorization Machines. They extracted different types of relationships such as composition relationships between services and the annotation relationships between mashups and tags as features. Factorization Machines can be used to predict suitable tags for mashups. The researchers demonstrated that the performance of their proposed method was very good [18]. Kapitsaki G. presented an approach for finding web service replacements, which can be helpful in software engineering process of web service-based systems with choosing appropriate services both on development and on execution time. In his proposed technique, WSDL features like port type, operation and message, user description and user tags sources were used for service tagging [19]. Shi et al. proposed a probabilistic topic model that automatically recommends tag for mashup. The model concurrently combines the composition relationships between mashups and APIs as well as the annotation relationships between APIs and tags. They calculated semantic similarity between mashups and Web APIs; accordingly, when the mashup and the APIs are most similar, tags of selected APIs are suggested to a mashup. Also, they provided a tag filtering algorithm that suggests the most relevant tags. To calculate the similarity of tags, they used both the relativity and popularity of tags. Their results showed that their proposed topic model-based method outperforms frequency-based methods [20]. Lin et al. provided a clustering method for web services

based on carrot search and K-mean clustering to group similar services and generate tags. They used text mining techniques to extract candidate tags from the WSDL documentations and employed the naive bayes machine learning technique for selecting related tags from among all candidate tags. In their study, information was extracted from WSDL, including services, ports, input type, output type, operations, binding, and messages, and WSDLs were indexed for faster retrieval. This method used co-occurrence of tags to enrich the tags of web services [6]. Fang et al. used two strategies for automatic tagging of web services, namely tag enriching and tag extraction. In the former, web services were clustered using WSDL documentation; using tags of other services in this cluster, the tags of a web service were enriched. They used five features in the WSDL, including Operation, Message, Type, Documentation, and Service. They used Normalized Google Distance (NGD) to calculate the similarity tags [4]. Zein Azmeh et al. offered an automated tagging service for web services. Using text mining, they extracted candidate tags from WSDL files and, then, selected related tags from candidate tags using a machine learning algorithm. They used only the relevant synonyms in Word Net to enrich and increase the tags. They did not use hand-crafted service tags [21]. Falleri et al. presented a method which automatically extracted tags from descriptions of web services. Similar to the study by Azmeh et al., text mining techniques were used to extract candidate tags, and machine learning techniques to select related tags from among these candidate tags [22]. Chen et al. used WSDL and tags for clustering, and employed two strategies to suggest the tag. They used Normalized Google Distance to determine the similarity between the two web services and the K-mean algorithm to cluster the web services. The study noted that only the use of extraction tags from WSDL could limit the discovery of services. In this research, the accuracy of service discovery is more important than performance and runtime [23]. K. Zheng et al. clustered web services users based on their interests. This information was analyzed and then compared to the features of web services to find the proper service. After service discovery, the quality parameters were evaluated to determine whether the user's expectations were satisfied or not [24]. Zhu et al. presented a similarity model to measure the similarity between web services in order for clustering. Their proposed model used a preprocessing approach that takes into account programming style and names rules of the services, and focuses more on the similarity of the web services functionality [25]. Pop et al. presented a semantic clustering method using the ant ontology concept. To measure the similarity between services, a matching method was presented, together

with a set of metrics. The metrics measured the degree of matching and the similarity between the two services based on their ontology descriptions. This clustering method was based on the functional similarity of web services. They did not use tags in their method [26]. Gholamzadeh and Taghiyareh used a data mining technique and fuzzy semantic clustering algorithm. By reducing search space, they improved service discovery process, accuracy, and speed [27]. Nandi proposed a hybrid method called WSTREC to suggest a tag. They used tag mining, co-occurrence of tags, and semantic relevance measurements for tag recommendation. The Jaccard coefficient was applied to determine co-occurrence. In their study, the similarity between web services was based on data tagging and five extracted features from the WSDL document. Their proposed method improved recall and precision [28]. Jian Wu et al. used both WSDL and tags to cluster Web services. In order to solve the problem of uneven tag distribution and noisy tags, they proposed a hybrid web service tag recommendation strategy called WSTREC, which used a tag mining technique for tag extraction, co-occurrence of tags for proposing candidate tags, and a semantic relevance measurement technique for ranking candidate tags and removing noisy tags. They extracted five features from WSDL documents, namely content, type, message, port, and service name. Since data pre-processing and web service clustering were executed off-line, efficiency was not very important, and their concentration was on accuracy. Results showed improved accuracy and recall in their method [5]. Shi et al. proposed a hybrid approach consists of two continuous processes: APIs selection and tags ranking. in first level, the most important APIs of a new mashup are selected based on a probabilistic topic model and a weighted PageRank algorithm. Then the proposed model to extract topic information, simultaneously incorporates the composition relationships between mashups and APIs as well as the annotation relationships between APIs and tags. Then, tags of chosen important APIs are recommended to this mashup [29].

Tagging is also used in software development. According to many researches [3, 13, 30, 31] MARIO system is one of the best works using tag-based descriptions as the basis for components' annotations; it enables the users to achieve their goals with regular search. MARIO system facilitates composition of components to create programs that meet end-user's goals [3, 32].

Studies have shown that most of the researches on web service clustering only used extracted tags from WSDL files and did not use users' handmade tags and also clustering web services was more based on the similarity of

WSDL files and co-occurrence of tags and semantic similarity of tags are not considered so it can reduce quality of clustering.

3 Motivation Example

Initially, a scenario [3] was presented which illustrated the motivation and purpose of the study (Figure 1). In this scenario, it was assumed that a Chinese tourist is planning to travel to Milan and use several related services. At first, he plans to reserve a hotel and find some Italian restaurants. Just like using a search engine, he can use a series of tags as query keywords, which include "Milan, Hilton hotel, Italian food, and map". These tags are interpreted as requests and are linked to the related services. For instance, the tags "Hilton hotel, Milan" were used as inputs by a hotel search service such as the Hotel Club or Google Map. There may be several hotels associated with the tag "Hilton hotel" (for instance, Milan Hilton Hotel, Hilton and Hilton Garden Inn, Milan Malpensa Hotel, etc.) and many restaurants associated with the tag "Italian Food". The tourist himself can check the returned results and choose one. For instance, he can choose "Milan, Hilton hotel" and then choose the LaLocanda Del Gatto Rosso restaurant from Yelp as an Italian food label. At the moment, the results of the three selected services – Google Map, Hotel Club, and Yelp – have been displayed to the tourist. These results included the name, address, and telephone number of the hotel and restaurant. The tourist may want to search for more services related to these two services. He is likely to select an address, phone, and zip code to search for new fun opportunities. For instance, he might add the desired address of the hotel and restaurant and the "driving" tag to get a guide to drive between the hotel and the restaurant. Based on these purposes, local Yahoo search service can be introduced because it can generate a path between two points on the map. Then, the Hotel Club, Yelp, and Yahoo local search will be merged due to the dependency of their tags. Similarly, he can use the VOIP service and, by choosing a phone number, book a hotel and restaurant through a phone call; he can also use the Google Translate service to translate the menu. Additionally, when the tourist browses the menu, he can add other tags such as "Veal Shank Meat". Next, Flickr Photo Search can be combined with Yelp to show images of the food [3].

In this way, the tourist can change his program by adding or changing the tags and gain more from these combined opportunities. He can choose one of the flows, such as Hotel Club, Yelp, Yahoo! Local, Google Map, and Google

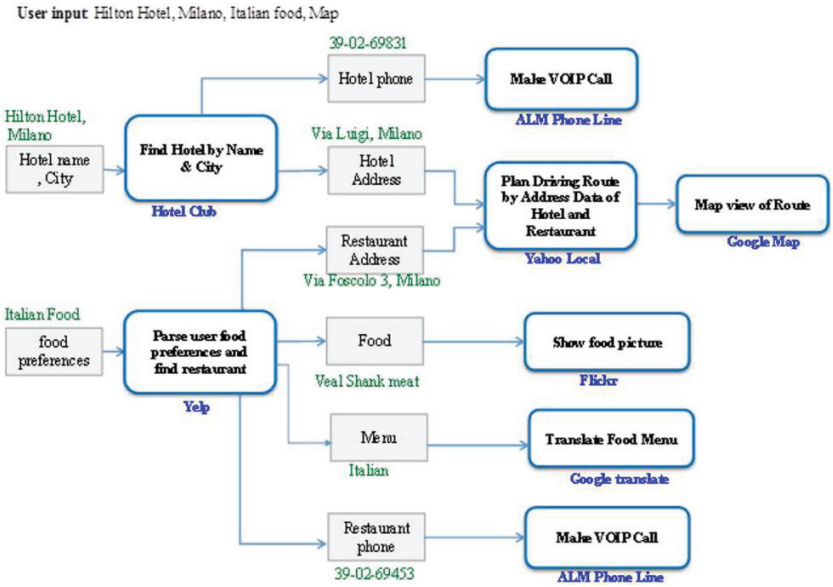

Figure 1 Motivational example [3].

Translate, and implement this flow on a combined search engine. All stages of the combining process are shown in Figure 1.

4 The Proposed Method

Tagging is widely used in popular Web 2.0 sites to simplify browsing and enrich the meta-data for web service describing. Tag-based search improves the search performance of a single web service and, consequently, promotes the quality of web service composition. The problem is that most of the currently available web services do not have enough meaningful tags and in most of the existing tag recommendation approaches [4, 6], semantics of tags is not perfectly considered. Accordingly, we have composed two strategies; tag enriching considering the semantics of words, and tag-based web service composition. A novel approach was, then, proposed based on these strategies.

The proposed method in this paper consists of two main phases; data processing and service composition. The data processing phase is independent of the user's query, including data collection, tag extraction and refinement, semantic clustering, and tag enrichment. This phase is performed offline (Figure 2). The service composition phase involves choosing appropriate

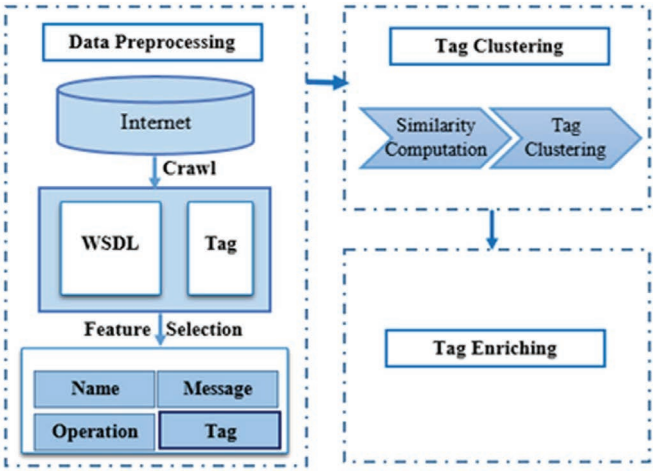

Figure 2 Data processing stage.

services, combining them, and applying a geographical distance quality parameter, which is dependent on the user's query, and is implemented online for each query. In this article, geographical distance is considered for combining services to select the nearest service to the user for the combination. Of course, there is no limit on the selection of QoS parameters, and it is easy to consider any number of quality parameters. This approach is a dynamic one, and service composition is implemented at runtime and after user request.

4.1 Data Processing

Many current common web services often come with text specifications which describe their information, such as functionalities, interfaces, input and output, and even quality attributes. This section describes how to collect and extract tags from text files, create a hierarchical structure of tags, enrich the tags, and so on.

4.1.1 Data collection

Since the purpose of this project is to automatically combine web services based on tags, it is necessary to first extract sufficient and meaningful tags for each web service. In this research, tags were extracted from service textual descriptions such as WSDL files, and user-generated annotations, namely hand-made tags, if present [19, 21]. WSDL files and handmade tags used in this study were extracted by [23].

4.1.2 Processing WSDL files, extracting and refining tags

In this stage, the WSDL files were processed to extract useful features of Web services. WSDL documents, are actually a XML style documents, which describe web services functionalities, and tags can be extracted from elements such as service name, message, operation, port, type, content, and binding. Useful tags usually reside in service name and/or service interfaces. Different elements have been used for extracting tags in various studies [3, 4, 6, 12, 19, 23]. Among the most common elements, one can refer to the name of the service, message, and operations. The name of service describes general information, and service interfaces (including operations and input/output messages) describe service usage. In this research, WSDL files of all web services were processed and tags were extracted from elements such as service name, message, and operations, and each tag was kept in memory as a record. In addition to the tags that were extracted from the WSDL file, user generated tags were also used and added to the tags (Figure 3). Since, Chen et al. [23], have extracted tags from user queries, feedbacks, and comments as handmade tags, and the tags they extracted were readily used in the present study as user tags, in this research, their extracted tags were used.

The surveys conducted found that more than 90% of the WSDL documents used capital letters, numbers, or "" to separate tokens in the names of the services [4]. Therefore, the following rules were used to separate tokens from tags [3]:

- Capital letters, numbers, and "%" and "_" are considered as the beginning of a new word.
- The first place is considered as the beginning of a new word.
- Continual, single capitals, and the numbers connected to them are merged into a token.

For example, based on these rules, the service name "AmazonSimpleDB" is split into three tags; Amazon, Simple, and DB [3].

Since many extracted tags are common words and do not convey significant information, the set of stop words such as uppercase words and common words that are not important in search (for example important, able, similar, so on) were removed from the list of tags of each web service, and set of web service tags was updated. Finally, Porter stemming algorithm was employed for keyword stemming in order to bring all terms in the same format and, then, the tags of all services were stored in a list and duplicated words were deleted to obtain a list of meaningful words without repetition to be used in clustering. All stages of processing phase are shown in Figure 3. As the

```
Processing WSDL Files, Extracting and Refining Tags
Input: Set ws_set of web service descriptions
Output: List of meaningful terms for each web service in the set
for each WSi ∈ ws_set
    begin
        // collection of available information
        WSi _wsdl:={service name ∪ operation ∪ message    documentation terms from WSDL}
        if user tag exists then
            WSi _usrDescr := extract terms from user description text
        // data preprocessing
        WSi _data:={WSi _wsdl∪ WSi _userDescr }
        for each term Ti in WSi _data
            begin
                tokenize(Ti )
                if Ti ∈ stop words then remove Ti
                stem(Ti )
                delete duplicate (Ti)
            end
    end
```

Figure 3 The pseudo-code of processing WSDL files, extracting and refining tags.

presented algorithm in this figure shows, the input of this algorithm is a set of descriptions of web services and its output is a list of terms meaningful for each web service. Initially, for each web service WS_i, the available information about it, such as the service name, operations, and messages, are extracted and stored in a list related to WS_i. Then, if there are user-generated tags WS_i, these tags are extracted from user descriptions and added to the list. In the pre-processing step, any term in the list is removed from it if it is a stop word. The final list is a collection of meaningful terms about web services that are used in the enrichment stage of other web services tags as well as in the composition process.

4.1.3 Hierarchical clustering of extracted tags

The ambiguity of tags (i.e. the same tag to express different meaning or different tags with the same meaning) will have significant side effects on performance and efficiency and lead users to lose valuable information and gain redundant information. Having a large number of tags also cause problems with browsing systems capabilities and the software domain. Therefore, it is argued that the tags describing services should be defined by controlled words.

Tag clustering can be implemented in a variety of ways. Since in data-driven composition of web services and tag enriching, semantics of tags is of great significance, Word Net was used in the current study to calculate semantic similarity of tags. Moreover, an average-linkage hierarchical clustering algorithm was also used in this study. We used this algorithm because hierarchical clustering methods can be used on all types of data and also produce high quality clusters. Compared to partitional clustering algorithms such as K-means, this method has higher runtime, but it does not need any predefined parameters such as number of clusters, so it is suitable for clustering real-world data [33]. For the purpose of clustering, it was necessary to calculate the distance between words. There are different algorithms for calculating the distance and similarity between words. For example, the Levenshtein distance considers the written form of words [34] and the Normalized Google Distance is calculated based on co-occurrence of the tags [35]. However, none of these two methods can fully calculate semantic similarity of words. In this research, the semantic distance between words was calculated through the Word Net semantic similarity using the WS4J library in Java and was saved in an M*N array. In Word Net, synonyms are linked to each other using semantic-conceptual relationships and lexical relationships. All the stages of clustering phase are shown in Figure 4.

As the figure shows, the input of the algorithm is the set of extractive tags and the matrix of the semantic distance between the tags, and its output is the hierarchical clusters of the tags. As described, the semantic similarity matrix between the tags is obtained using the WS4J library in Java. Initially, each tag is considered a cluster. That is, with n tags, there are n clusters in the first step. In the second step, the two clusters with the shortest distance are selected and merged, thus the number of clusters are reduced by one. The distance between the new cluster and the other clusters is calculated according to Equation (1):

$$\mathrm{D(a,b)} = \sum \mathrm{(x,y)/N(a) * N(b)} \tag{1}$$

Where a and b are two clusters and x and y are members of the data set in clusters a and b, respectively. To calculate the average distance, sum of distance of cluster members a and b is divided by the product of the number of members of cluster a and cluster b.

The above steps are repeated to finally obtain a cluster.

4.1.4 Tag enrichment/tag recommendation

Since many web services do not have enough tags and, therefore, may not be well discovered and selected in the service composition process, it is

```
Hierarchical Agglomerative Clustering of tags
Input:
    A set of tags T= {t1, t2, …, tn}
    A distance function dist (c1, c2)  // average-linkage method
Output:
    Clusters C= {c1, c2, …,cm}
begin
    for i=1 to n
        ci= {ti}
    end
    C= {c1, c2, …,cn}

    L=n+1

    While C.size>1 do

        (cmin1, cmin2)=minimum dist(ci, cj) for all ci,cj in C

        Remove cmin1 and cmin2 from C

        Add (cmin1, cmin2) to C

        L=l+1

    end while

end
```

Figure 4 The pseudo-code of hierarchical clustering algorithm.

necessary for these services to do tag recommendation and enrichment, which can be conducted in four phases (Figure 5):

1. Candidate tag selection: the tags placed in the same cluster as service tags have a high semantic similarity and can be used as candidate tags for enrichment. All the brothers of such tags are selected as the candidate tag.
2. Candidate tag ranking: co-occurrence metric was used to rank tags [23]. The co-occurrence can be calculated by different methods. In this study, the Jaccard coefficient method [36] was used as it is widely used to compare similarities between words and, it can be given the proximity of the two data sets efficiently without the use of data redundancy [37].

$$\mathrm{Co}(t_i, t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|} \tag{2}$$

   – $|t_i \cap t_j|$ represents the number of web services with both t_i and t_j tags.
   – $|t_i \cup t_j|$ means the number of web services in which the t_i or t_j tag is used.
3. k-top tag selection: in this phase, k-top tags with the highest co-occurrence score with the web service tags are selected.

WS_k^T: Tags related to the WS_k web service

WS_k^{CT}: Candidate tags related with the WS_k web service (for enrichment)

Brothers (ti): Tags placed in a same cluster with tag ti.

$WS_k^{CT} = \Theta$
For all WS_k(k=1 to n)
    For all ti $\in WS_k^T$
        $WS_k^{CT} \leftarrow WS_k^{CT} \cup$ brothers(ti)
    End
    For all ti $\in WS_k^T$
        For all ti $\in WS_k^{CT}$
            $Co(ti, tj) = \frac{|ti \cap tj|}{|ti \cup tj|}$
        End
    End
    If $Co(ti, tj) \in$ k-top // select k-top co-occurrence
        $WS_k^T \leftarrow WS_k^T \cup tj$
end

Figure 5 The pseudo-code of tag enrichment algorithm.

4. Adding the selected tags: in this phase, the selected tags are added to the list of web service tags and the list gets updated.

4.1.5 Determining input and output tags

Ultimately, each web service contained a sufficient number of meaningful tags after the enrichment phase. Since input and output tags of the web services were required in the service composition phase, a number of tags for each web service were randomly selected and considered as input tags. Other tags were output tags. Then, a function maps input/output parameter in the service description to a set of tags. In this study, similar to [3], the nouns of web service messages are assigned to tags. So, a service from a data centric perspective is included requests and responses that they are represented in form of tags.

4.2 Tag-Based Service Composition

This phase, which is executed online and dynamically, depends on the user's query and begins after receiving the desired tags from the user. In the first step, the purpose was to find the most relevant services according to the query. In the next step, these web services were arranged based on their geographical distance. The first step was performed by a forward search algorithm, and then the services were ordered by geographical distance and were suggested

to the user. Different quality parameters can be used, including performance, reliability, cost, and time; however, geographical distance was used in this study as quality of composed services.

4.2.1 Definition of composition planning problem

Based on tag-based semantic rules, if web service ws_1 generates tag t_1 as its own output and the web service ws_2 can use t_1 as input, it is assumed that ws_1 and ws_2 can be composed because a data flow can be generated between them. From this perspective, the tag-based service composition problem is introduced as a result of creating a data flow between sequences of tags.

In this research, a web service ws was describe as a four-tuple $<T_i, T_o, X, Y>$, where $T_i \subseteq T$, $T_o \subseteq T$. T_i and T_o denote a set of input and output tags, respectively and X and Y represent the latitude and longitude of the web service, respectively. T also shows the collection of tags extracted from all web services. q indicates the set of user query tags.

The two ws_k and ws_{k+1} web services can be composed when the ws_{k+1} service input tags are present in the ws_k service output.

In brief, the tag-based composition problem finds a finite sequence of ws_1, ws_2, ws_3... services where:

- $\forall\, ws_k$,$T_{ik} + 1 \subseteq T_{oK}$
- $T_f \subseteq (q \cup T_{1o} \cup T_{2o} \cup \ldots \cup T_{no})$
- The total cost function is minimized, which means only those services are selected which have a minimum geographical distance from each other according to Equation (3).

$$\text{Distance}(ws_1, ws_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3}$$

Where x_1 and y_1 indicate geographical latitude and longitude of web service1 (ws_1), respectively and x_2, y_2 denote the geographical latitude and longitude of web service 2 (ws_2), respectively.

4.2.2 Forward search algorithm

This algorithm discovers the services related to users' query tags and composes them using the following algorithm:

Input:
Set of input and output tags, and geographical latitude and longitude of web services
User input tags ($q \subseteq T$).

Output:

- A set of seed web services; web services that match the user's purpose (high priority services), W_s.
- A set of web services that are combined together and meet the user's purpose (W_f).
- A set of output tags (T_f).

In the first step and after receiving the user's desired tags, the purpose is to find the most relevant services (services that contain all user tags and user QoS) in relation with the received query. Since the user is expected to enter a small number of tags at the beginning, all web services are searched to find web services that contain all user tags; if found, they will be located in the list of seed web services and their output tags will be added to the list of composition tags. If there are no web services in the seed list, services that include at least one of the input tags will be added to the seed list. So far, seed web services have been found and, now, based on the input and output tags of each of the web services, the composition operation of web services must take place.

Based on the forward search algorithm, all web services compare their input tags with composition tags set (T_f). If all the input tags of the web service are included in the list of the output tags of composed services or seed services, the web service will be added to the composed service list.

With this algorithm, all the possible composition solutions are generated and a list of services which meet the user's goal are proposed (Figure 6).

Input: q, training data (<T^i,T^o,X, Y>)
Output: w_{seed}, w_f, t_f
Initialization;
T_f=Θ , w_f= Θ , w_{seed}= Θ;
//finding seed services
For k= 1 to n
If q⊆ T_k^ithen{ W_{seed}= w_{seed} ∪ ws_k , T_f= T_f ∪ T_k^o}
End
If ws= Θ then
{
For all wsk that T_k^i contains q
{ W_{seed}= w_{seed} ∪ ws_k , T_f= T_f ∪ T_k^o}
}
//composit services
For all wsk that T_k^i ⊆ T_f
{
W_f = wf ∪ wsk;
T_f= T_f ∪ T_k^o
}
//applying quality parameter (geographical distance)
For all W_{seed} // W_{seed}={$W_{seed\,1}$, $W_{seed\,2}$.. $W_{seed\,i}$)
For all W_f // W_f={Wf_1, $W_{f\,2}$... W_{fj})
ServiceDistance (Wseed i, Wf j)
//Distance(ws1, ws2) = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
SortedServices =InDescendingDistanceOrder(serviceDistance)

Figure 6 Forward search algorithm.

These composed web services can potentially provide more features to users. In the last step, based on the quality parameter defined for the problem (the geographical latitude and longitude of each web service, in this case), the distance between the selected services is calculated according to Equation (3). Composed services are ranked according to this. In other words, the user will receive a list of seed services, along with its nearest services, matching his queries.

5 Evaluation of the Proposed Method

Tag semantics plays an important role in the proposed composition method, and textual description of web services was needed to extract tags. The data set in this experiment consist of 2000 real web services which were available online and free of charge [23]. The services were in different groups including travel, news, weather, maps, geography, food, photo, messaging, business, music, translate. Of these web services, which Only 310 services contained handmade tags. At first, the splitting technique was used to extract tags from text descriptions; this gave a set of 27784 tags, from which the unnecessary and meaningless tags, such as prepositions and repeated words, were filtered and deleted (Figure 7). Eventually, a collection with 20801 different tags was created. Finally, according to the proposed method, these tags are clustered and enriched and used in the service composition process. Examples of program outputs are shown in Figures 8 and 9. In order to evaluate the proposed method, the quality of the suggestions and performance of the program was examined in spite of the increase in the number of web services.

5.1 Evaluation of Precision and Recall

Generally, performance of service composition is depended upon the precision of discovering services as well as the precision of tag recommendation algorithm. Hence, it has been tried to evaluate the precision of the proposed enriching tag method. To evaluate the performance of tag recommendation method, the two following metrics were used:

Recall@K: the Recall of top-k tags recommended. It is defined as:

$$\mathrm{Recall@K} = \frac{\mathrm{successtag}}{\mathrm{successtag} + \mathrm{missedtag}} \tag{4}$$

Service no. 112 - Service Name: Dist_List02QueryService
Extracted Tags: [list, result, logout, query, service, login]

Service no. 113 - Service Name: cop.teamCapacityLinkableQueryService
Extracted Tags: [result, copteam, logout, linkable, query, service, capacity, login, auth]

Service no. 114 - Service Name: TeamIncidentsQueryService
Extracted Tags: [incidents, result, logout, query, service, login, team, auth]

Service no. 115 - Service Name: cop staffingLinkableQueryService
Extracted Tags: [result, logout, linkable, query, service, login, auth, copstaffing]

Service no. 116 - Service Name: custr4_goallookNSQLQueryService
Extracted Tags: [result, logout, custr4, query, service, login, auth, goallook]

Service no. 117 - Service Name: RiskExposurebyCategoryQueryService
Extracted Tags: [risk, result, category, logout, query, service, login, exposure,]

Figure 7 Example of extracted tags before enrichment.

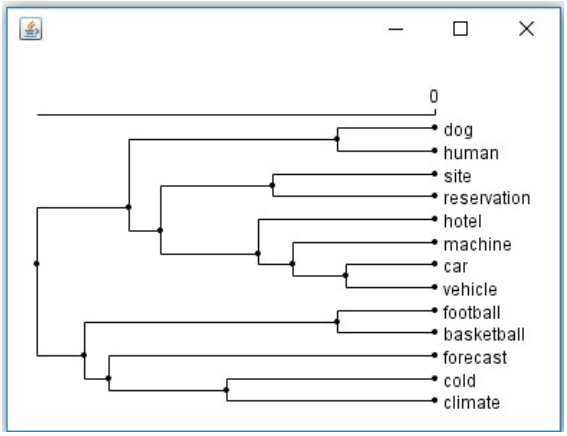

Figure 8 An example of hierarchical clustering of tags, using WordNet semantic similarity.

Precision@K: the Perecision of top-k tags recommended. It is defined as:

$$\text{Precision@K} = \frac{\text{successtag}}{\text{successtag} + \text{mispltag}} \tag{5}$$

Where successtag refers to the number of tags recommended successfully, missed tag is the number of tags that should be recommended but are not

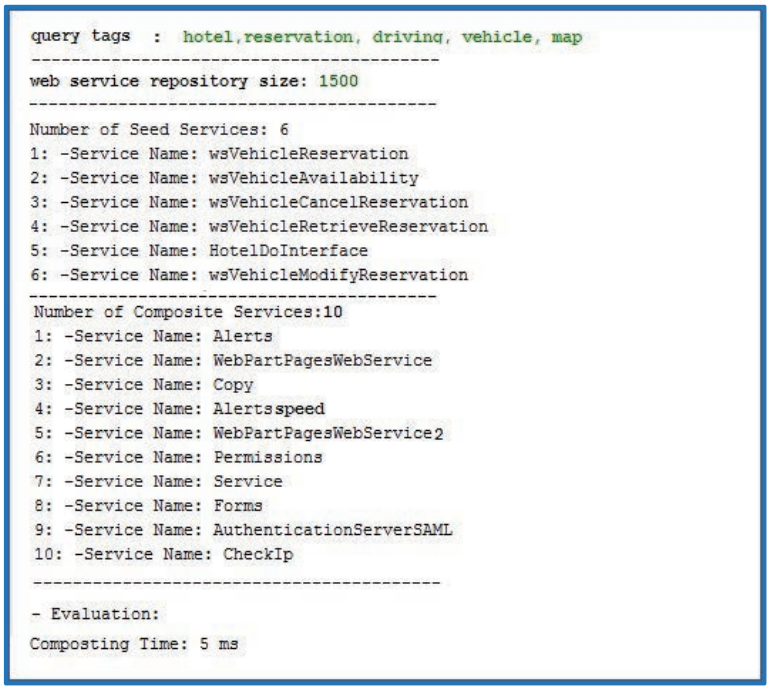

Figure 9 An example of a combined service based on enriched tags.

among the recommended tags, and mispltag is the number of tags that are incorrectly recommended. The results of the evaluation were compared with Term Frequency-Inverse Document Frequency (TF/IDF) technique. TF/IDF recommends tags for web services from the tag set of web services whose textual documents are similar. In TF/IDF calculating the similarity between documents is based on term frequency and inverse document frequency. Since TF/IDF method considers only lexical matching between documents rather than semantic similarity of tags, it may miss some important information. As shown in Figures 10 and 11, the proposed semantic-based method outperforms frequency-based method (TF/IDF). The results are completely similar and consistent to the previous studies [3, 18, 20].

5.2 Evaluation of the Quality of the Proposed Services

To evaluate the quality of the proposed services, two criteria were used: (1) how users ranked the proposed services against their queries; and (2)

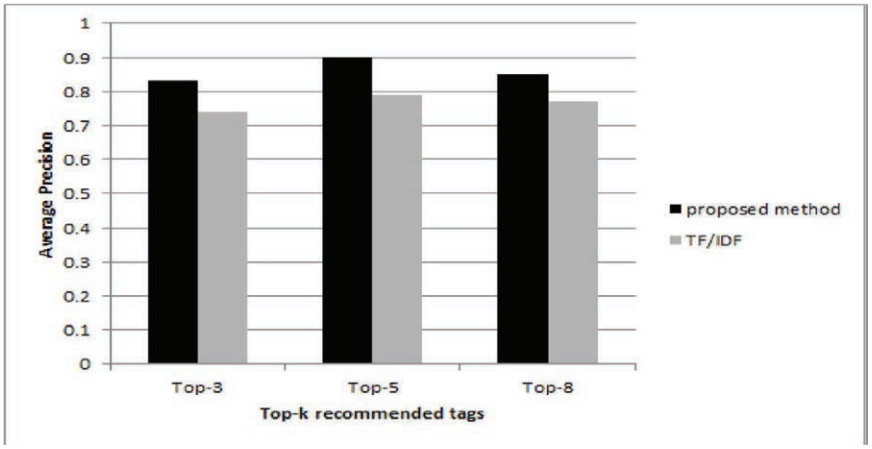

Figure 10 Precision of tag recommendation algorithm.

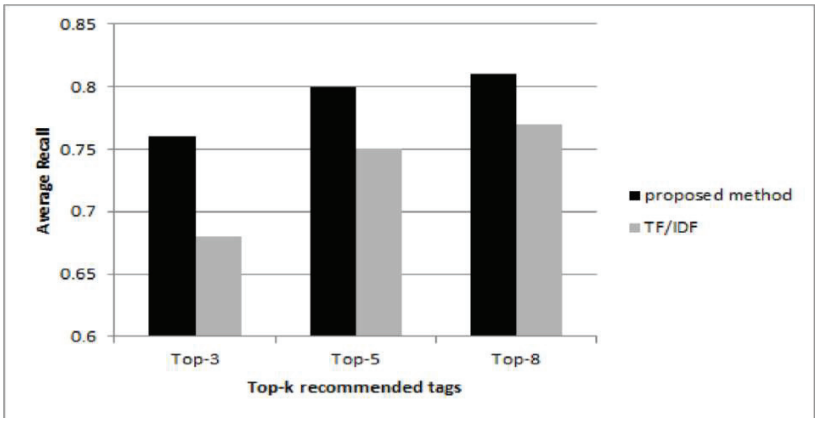

Figure 11 Recall of tag recommendation algorithm.

how many potentially new, relevant web services can be planned. Fifteen junior students majored in computer science were asked to participate in our user study. They had experiences in web service development and skills of tagging.

At first, users selected their desired services; the services were considered as reference, and their tags were entered as a query in the search engine of this study. Then from among five top recommended services, they reported how many of their goals were satisfied or how many desired outputs were

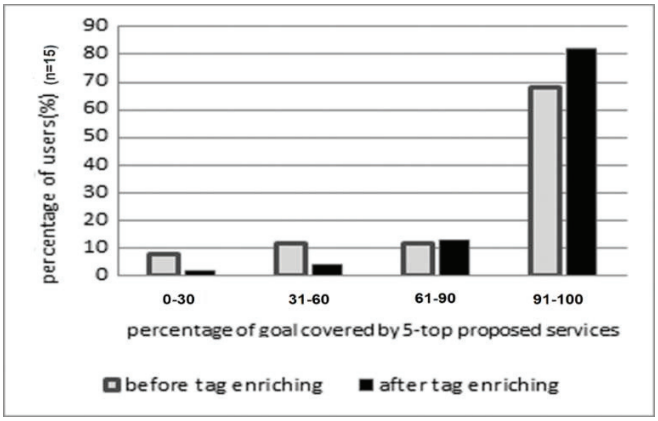

Figure 12 Percentage of user goal satisfaction with suggested services.

found among the services offered. Having run different queries, it was found that most of the user's desired web services were placed among the five top services offered to users. Feedback emerging from this is shown in Figure 12. It can be seen that before enriching the tags, only about 68% of the users were able to find more than 90% of their output in the first five proposed services, while after enriching, almost 81% of the users achieved over 90% of their target goals. This chart shows that the percentage of users' satisfaction with the proposed services increases after enrichment of the tags. In comparison with previous study [3] in which about 78% of users achieve 90% of their desired goals, the proposed algorithm in the current study showed superior performance after enriching.

Another benefit of this research is the discovery of potential compositions. As shown in Table 1, the proposed composition algorithm can achieve the desired goals. In other words, this method can generate more relevant composition solutions and increases the opportunities for compositions. Also, other services beyond the initial goals of the users can be added to the outputs as these solutions are calculated based on potential semantic relationships. Another fact is that in 90% of cases, the proposed technique (after enrichment) can offer more web services (Figure 13).

5.3 Evaluation of the Composition Planning Function

In this study, at first, we conducted multiple queries against different sizes of the service repository from 200 to 2000 services. Each query contained

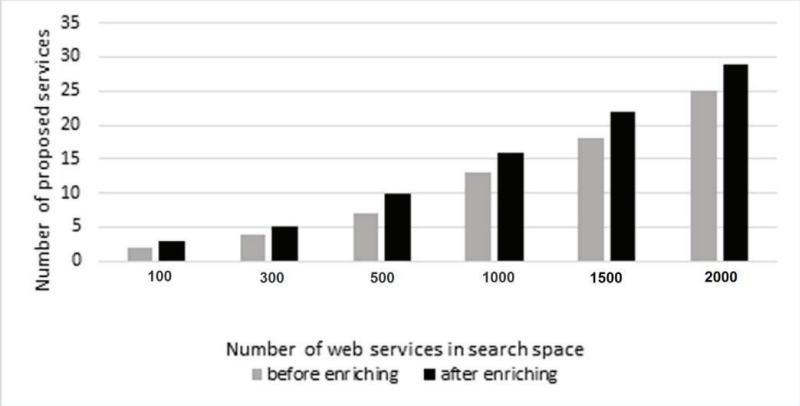

Figure 13 The association of number of proposed services with number of repository services.

at least 3 tags. Then, composition algorithm time, memory consumption, and number of suggested services were evaluated for processing requested queries. Some of these queries are shown in Table 1.

This study examined how the number of repository services affected response time, memory consumption, and number of suggested services (Figures 13–15).

5.3.1 Composition algorithm run time evaluation

Conducting various queries on the service repository in different sizes, it was found that the runtime increases almost linearly when the number of services increases in search space. It should be noted that the mentioned time solely contains the composition algorithm execution time and does not include the clustering algorithm time. This is because the clustering algorithm is executed only when the web services repository is changed and whenever a user requests a query, just the composition algorithm is run. The results of the runtime algorithm are shown in Figure 14.

Considering that the clustering algorithm used in this research is a hierarchical clustering algorithm, with increasing the number of services in search space, and consequently increasing the number of tags, run time of this clustering algorithm increases, but since the data pre-processing and clustering process are executed only when web services repository is updated, the efficiency and run time of hierarchical clustering algorithm are not a big concern, whereas the accuracy is of great significance.

Table 1 Program performance with sample composition requests

N	Number of Services in Repository	Required Tags	Before Tag Enriching			After Tag Enriching		
			Run Time (Milliseconds)	Number of Core Services	Number of Suggested Services	Run Time (Milliseconds)	Number of Core Services	Number of Suggested Services
1	500	Japan, hotel, reservation, map	7	3	5	6	3	7
2	500	Japan, hotel, reservation, map, driving, vehicle, weather, forecast	5	4	9	5	6	10
3	1000	Japan, restaurant, address, photo	19	7	13	16	8	16
4	1000	Japan, hotel, reservation, map, driving, vehicle, weather, forecast	15	8	15	13	8	19
5	2000	restaurant, food, picture, search	34	6	25	26	7	31

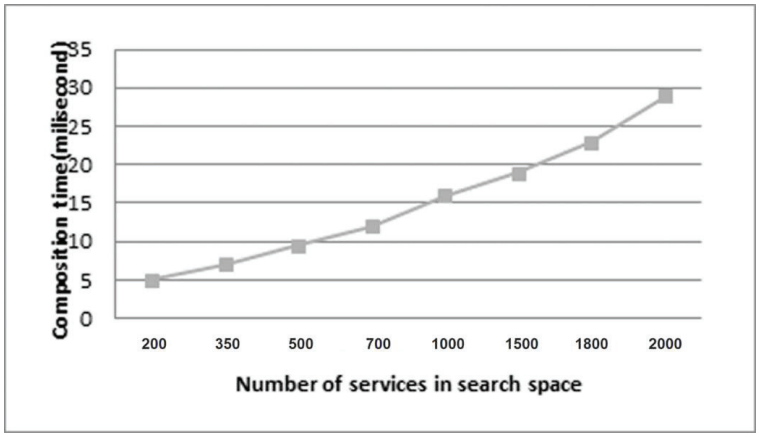

Figure 14 Composition algorithm runtime.

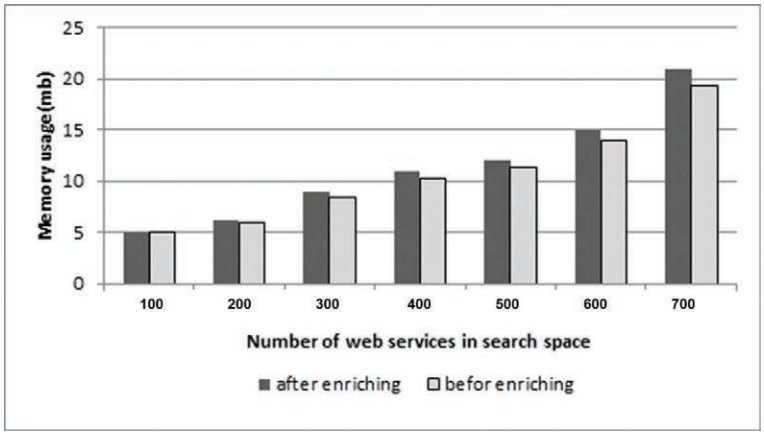

Figure 15 Memory consumption with repository size.

5.3.2 Evaluation of memory consumption by the number of services in search space

Multiple queries were made on the basis of different sizes of the service repository. It was found that in both approaches (before and after enrichment), the memory consumption increases almost linearly with increasing the number of services in search space. In the proposed method, due to the tag enrichment and increase in the number of tags, memory consumption increased negligibly as compared to the previous studies (without enrichment). The results are shown in Figure 15.

6 Conclusions and Future Works

As tagging is an important way of managing web service resources, a novel method was proposed in this paper for enriching tags so as to improve the data-driven composition of web services. Since semantic clustering of tags and tags enrichment have significant effects on the quality of composed services, the proposed method was able to improve the quality of composed services. Experimental results showed that semantic-based tag recommendation method outperforms the frequency-based method (TF/IDF), also that qualities of the offered services and users' satisfaction improved as a result of semantic clustering of tags and tag enrichment.

One of our future goals is to improve the clustering algorithm runtime and scalability, also to consider more quality parameters such as response time and availability. In addition, in this paper, the input and output tags were randomly selected from extraction tags. We intend to extract the input and output tags rather definitely in the future. It is also better to compare the proposed method with other methods that have done the composition process based on tags but have not considered tag enrichment.

References

[1] Q.Z. Sheng, et al., 'Web services composition: A decade's overview'. Information Sciences, 280:218–238, 2014.

[2] G. Zou, et al., 'Dynamic composition of Web services using efficient planners in large-scale service repository'. Knowledge-Based Systems, 62:98–112, 2014.

[3] X. Liu, et al., 'Data-driven composition for service-oriented situational web applications'. IEEE Transactions on Services Computing, 8(1):2–16, 2015.

[4] L. Fang, et al., 'Towards automatic tagging for web services'. in Web Services (ICWS), 2012 IEEE 19th International Conference on. IEEE, 2012.

[5] J. Wu, et al., 'Clustering web services to facilitate service discovery'. Knowledge and information systems, 38(1):207–229, 2014.

[6] M. Lin and D.W. Cheung. 'Automatic tagging web services using machine learning techniques'. in Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on. IEEE, 2014.

[7] S. Song and S.-W. Lee, 'A goal-driven approach for adaptive service composition using planning'. Mathematical and Computer Modelling, 58(1):261–273, 2013.
[8] W. Tan, et al., 'Data-driven service composition in enterprise SOA solutions: A Petri net approach'. IEEE Transactions on Automation Science and Engineering, 7(3):686–694, 2010.
[9] S.-P. Ma, et al., 'Real-world RESTful service composition: a transformation-annotation-discovery approach'. in 2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2017.
[10] A. De Renzis, et al., 'Case-based reasoning for web service discovery and selection'. Electronic Notes in Theoretical Computer Science, 321:89–112, 2016.
[11] S. Deng, et al., 'Top-k Automatic Service Composition: A Parallel Method for Large-Scale Service Sets'. IEEE Transactions on Automation Science and Engineering, 11(3):891–905, 2014.
[12] X. Liu, et al., 'Composing data-driven service mashups with tag-based semantic annotations'. in Web Services (ICWS), 2011 IEEE International Conference on. IEEE, 2011.
[13] E. Bouillet, et al., 'A tag-based approach for the design and composition of information processing applications'. in ACM Sigplan Notices. ACM, 2008.
[14] D. Wang, Y. Yang, and Z. Mi, 'A genetic-based approach to web service composition in geo-distributed cloud environment'. Computers & Electrical Engineering, 43:129–141, 2015.
[15] Y.-Y. Fanjiang and Y. Syu, 'Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach'. Information and Software Technology, 56(3):352–373, 2014.
[16] Z. Gao, et al., 'Discovery and Analysis About the Evolution of Service Composition Patterns'. Journal of Web Engineering, 18(7):579–626, 2019.
[17] C. Vairetti, R. Alarcon and J. Bellido, 'A semantic approach for dynamically determining complex composed service behaviour'. Journal of Web Engineering, 15(3&4):310–338, 2016.
[18] M. Shi, et al., 'A topic-sensitive method for mashup tag recommendation utilizing multi-relational service data'. IEEE Transactions on Services Computing, 2018.

[19] G.M. Kapitsaki, 'Creating and utilizing section-level Web service tags in service replaceability'. Service Oriented Computing and Applications, 11(3):285–299, 2017.
[20] M. Shi, et al., 'A probabilistic topic model for mashup tag recommendation'. in 2016 IEEE International Conference on Web Services (ICWS). IEEE, 2016.
[21] Z. Azmeh, et al., 'Automatic Web Service Tagging Using Machine Learning and WordNet Synsets'. in WEBIST (Selected Papers). Springer, 2010.
[22] J.-R. Falleri, et al., 'Automatic tag identification in web service descriptions'. in WEBIST'10: The International Conference on Web Information Systems and Technology. 2010.
[23] L. Chen, et al., 'Wtcluster: Utilizing tags for web services clustering'. in International Conference on Service-Oriented Computing. Springer, 2011.
[24] K. Zheng, et al., 'User clustering-based web service discovery'. in Internet Computing for Science and Engineering (ICICSE), 2012 Sixth International Conference on. IEEE, 2012.
[25] Z. Zhu, et al., 'WS-SCAN: A effective approach for web services clustering'. in Computer Application and System Modeling (ICCASM), 2010 International Conference on. IEEE, 2010.
[26] C.B. Pop, et al., 'Semantic Web Service Clustering for Efficient Discovery Using an Ant-Based Method'. in IDC. Springer, 2010.
[27] N. Gholamzadeh and F. Taghiyareh. 'Ontology-based fuzzy web services clustering'. in Telecommunications (IST), 2010 5th International Symposium on. IEEE, 2010.
[28] D.K.V. Nandini N, 'Facilitating the Service Discovery for the Cluster of Web Services using Hybrid WSTRec'. International Journal of Advanced Research in Computer Science and Software Engineering, 5(2):5, 2015.
[29] M. Shi, J. Liu D. Zhou, 'A hybrid approach for automatic mashup tag recommendation'. J Web Eng, 16(7&8):676–692, 2017.
[30] E. Bouillet, et al., 'A faceted requirements-driven approach to service design and composition'. in Web Services, 2008. ICWS'08. IEEE International Conference on. IEEE, 2008.
[31] A. Ranganathan, A. Riabov, and O. Udrea. 'Mashup-based information retrieval for domain experts'. in Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009.

[32] A.V. Riabov, et al., 'Wishful search: interactive composition of data mashups'. in Proceedings of the 17th international conference on World Wide Web. ACM, 2008.
[33] A. Bouguettaya, et al., 'Efficient agglomerative hierarchical clustering'. Expert Systems with Applications, 42(5):2785–2797, 2015.
[34] Z. Su, et al., 'Plagiarism detection using the Levenshtein distance and Smith-Waterman algorithm'. in Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on. IEEE, 2008.
[35] R.L. Cilibrasi and P.M. Vitanyi, 'The google similarity distance'. IEEE Transactions on knowledge and data engineering, 19(3), 2007.
[36] A.K. Jain and R.C. Dubes, Algorithms for clustering data. Prentice-Hall, Inc., 1988.
[37] S. Niwattanakul, et al., 'Using of Jaccard coefficient for keywords similarity'. in Proceedings of the international multiconference of engineers and computer scientists. 2013.

Biographies

Sima Emadi is an Assistant Professor and Director of Computer postgraduate at Computer Engineering Department, Islamic Azad University, Yazd Branch. He received the B.Eng. degree from Islamic Azad University, Iran, in 1995 and the M.S. degree from Islamic Azad University, Iran, in 1997, both in Computer Software engineering. In 2008 she completed the Ph.D. program at Islamic Azad University, Science and Research Branch, Iran. Her current research interests include services computing Software, Web service Composition, Service Driven Architecture, Software Testing and Design Pattern.

Nahid dara is a master of science in software engineering. She received the B.S degree from Yazd university in 2006 and M.S degree from Islamic Azad University of Yazd in 2016. She is a software engineer in Yazd Diabetes research Center, Shahid Sadoughi University of medical sciences from 2007. Her research interests include software engineering and data mining and database.