# Different Approaches for Web Service Discovery

Ajinkya Kolhe
ask2538@g.rit.edu

Jinesh Dhruv
jad6566@g.rit.edu

Nisha Bhanushali
nnb7791@g.rit.edu

## ABSTRACT

Web service discovery plays an important role for selecting best web service to match the user's request. This process involves querying UDDI registry to obtain the WSDL of the service which helps the client to interface with the web service. The query consists of user-specific information like the desired service, preferred price and a maximum number of returned results, etc. The matching service information published on the registries are returned to the requester. At the end of this discovery process, the requester can exactly locate the service and its description document. There have been different approaches proposed to make the web service discovery more efficient and easier than the traditional approach. Some of the approaches we have discussed are domain-ontology and linked social network with multiple feature attribute based discovery in big data analytics domain. Also, web services can be discovered using social network principles, semantic approaches and using mining and multilevel indexing. Our goal for this project is to study these approaches and implement one of them. The problem of finding the relevant web service to user's choice is resolved by incorporating Web services into social networks. To achieve this goal, we make use of user's query and his/her FoaF- based ego centric social profile and build a user's model in the social context [5]. A user can find the most relevant web-services to his query in his and close friends past invocations list of services. The performance evaluation of this system is done by comparing the traditional approaches of the discovery to the implemented approach using the terms of evaluation like precision, recall and F-measure.

## 1. INTRODUCTION

The evolution of web 2.0 which was marked by a transition of the web from purely static pages to dynamic pages and the increase in speed of the Internet opened a new horizon in the world of technology. This lead to the development of various aspects of the web like the introduction of new protocols, frameworks, and development of APIs, etc. Also with time, we observed an exponential growth in the use of cross-platform products and technologies which facilitated and acted as a catalyst in the growth of web services. Owing to the large-scale increase in the use of web services and a large number of choices available among the web services, it becomes crucial that we find an efficient and systematic approach to find the best possible methods and algorithms to find a web service that perfectly caters to consumers need. It is also equally important that such searching techniques are inclusive of the new web services that come into being every day as they may have a potential to become the next big thing in the market seeking quality web services.

Till date, most of the web services made use of UDDI and similar registries which were listed in a Global registry and written primarily in WSDL format among others. As the world of web is opening, the need for new methodologies to search and store web services has emerged, like in the case of mobile platforms with their dynamic nature. Similarly, the use of Big Data related data mining methods has opened the way for convenient searching of multi-indexing models and in-depth searching for more accurate results. The topic that we decided to cover is Web Service Discovery and Selection which is a result of all the above-mentioned contributing factors. The selection of different papers for the survey on the topic mentioned above was done with a purview of addressing the need of variability in types of searches as may be encountered in Web Service Selection. As such, the web services can be searched by making use of different Social Networks, Data mining techniques like Bloom Filters and Multilevel indexing, etc.

## 2. RELATED WORK

We have read 10 papers related to Web services discovery and selection to get a better understanding of this topic. This section summarizes all the papers.

### 2.1 Web Services related to Big Data Analytics Domain

Big data analytics is the process of analyzing a large amount of data to find out hidden patterns and uncover the knowledge from that data[7]. This process is a combination of various challenges like data cleaning, preparation, modeling, maintaining the quality of the data and analyzing the data. There is an increasing demand of the BDA in all the organizations and industries in today's world. Hence, there is a bright scope for the automation service composition in the BDA field and the paper has managed to take advantage of this demand.

The functional behavior of the ontology is represented by the CRISPDM i.e., Cross Industry Standard Process for Data Mining. The authors of this paper have already completed the phase one of bringing the automation service composition in the BDA field to reality. This paper is the next stage where they have focused on the discovery and composition of the services for the BDA procedure. The discovery process is **domain-ontology based** to ensure that the discovery approach is aware of the domain context and

to identify from the registry the services which are precise and fulfill effective service discovery for the ASC in BDA domain. Domain ontology is used to find the fine-grained services and manage them by manipulating their behavioral signature-level description. The BDA services are dependent on the prior service, due to which the stages of BDA are highly interdependent on each other. The **linked social service network (LSSN)** with multiple features attribute based services is used to provide the users, the facility to combine the services in an efficient manner. The relationship between the different services is obtained using hybrid term similarity between the web services.

The two approaches used for the discovery stage of the ASC process for BDA automation were domain-ontology based service discovery and LSSN with multiple feature similarity based service discovery. In the domain ontology-based service discovery, they build a domain ontology and prepare the service registry. The next step is to cluster the web services by calculating the **HTS (hybrid term similarity)** between the services. An abstract work-flow according to the functional properties of the user request, is created by the work-flow generator at the planning stage of the ASC framework. The services to be used for the particular task are found in two steps:

Step 1 is calculating the similarity based on HTS.

Step 2 finds the most suitable cluster group for a task by calculating the semantic similarity between the tasks and all the cluster centers.

This similarity is found out in three steps i.e., **Semantic feature extraction, Ontology learning, and feature similarity calculation**. In semantic feature extraction, the task features from the BDA domain ontology and service features from WSDL document are compared. The features which are compared are inputs, output, and task/service name. In the next step, dynamic ontologies are created by identifying the complex terms from the input and output features. Ranging this term by calculating the TF-IDF of those terms. The similarity between the features is obtained by calculating the HTS. The performance of the system is improved by distinguishing the fine-grained services. The performance measures taken into consideration are improving accuracy and reducing the noise. The results obtained are sorted and the cluster of services which have the highest-rank is selected for the particular task.

Another approach used for service discovery is LSSN with multiple feature similarity-based service discovery. This approach tries to bring near to optimum solution and highest precision which are important to satisfy the work-flow based on the ASC technology. The first step towards implementing the approach model is to build the BDA domain ontology which remains the same as the above procedure. In the next step, however, instead of building a domain ontology registry they have built an LSSN. The quality of the social links between the services is then calculated based on the QSL values. The link between the service and the target task is calculated taking into consideration the functionality, sociability, preferential connectivity and the Quality of Service between the services. The services which have worked with the task earlier or are very popular compared to other web services for that task have higher importance compared with the ones which are lesser used. The network of the services which is made based on this QSL is then available for use to the user. The edges towards the web services change according to the selections made by the user. So the two factors which majorly affect the selection of the web service is the semantic similarity between the services and the links that a web service has.

The next step is clustering the web services. The process of clustering the web service is similar to the one discussed in the domain ontology based discovery, the only difference here is instead of using hybrid term similarity, they have used QSL as the measure of calculation. The discovery of the web service is done finding out the cluster groups similar to the domain ontology method followed by finding sociable similarity value of tasks vs cluster group members. The sociable similarity is the combination of the sociability and semantic similarity into one value. The services are ranked in the increasing order of their sociable similarity value and the service which has the highest value is selected for a particular task.

## 2.2 Web Services using Social Network Principle

### *2.2.1* Framework for Weaving the Principles of Social Network

The discovery of web services has been a problem with the increasing number of Web services that offer similar utilities to users and other peers. To solve this problem, a framework for weaving social networks with web services discovery was suggested in [4]. The functionality of Web services can be similar or different, based on which the framework identifies three types of social interaction between web services that can exist, which are the substitution, competition, and collaboration. The social network is a graph which consists of nodes and edges. The nodes are the Web services and edges are the social interactions between the web services. Both the Substitution and Competition interaction consists of Web services of similar functionality and are used based on user's non-functional parameters (QoS). The substitution also follows semantic policy to replace a Web service with similar functionality in case of failure. The Collaboration interaction consists of Web services of different functionality and is used based on users request that requires a combination of Web services. It also follows semantic and policy matching to ensure that user's request is satisfied. Now, below are the steps involved in weaving social networks into Web services.

1. **Social network component identification:** In this step, we identify the purpose and properties of social interactions. The purpose of Substitution interaction is high availability for failure handling, Competition interaction is composition establishment and Collaboration interaction is composition enrichment.

2. **Web services matching analysis:** This step will provide different matching algorithms to calculate the degree of similarity and degree of complementarity between two Web services. The profiles of Web services are categorized into Preconditions (P), Inputs(I), Outputs(O), Effects(E) and QoS. The matching score is computed based on the above categories and based on the matching results, we can determine what type of social interaction exists among the two Web services.

3. **Social networks management:** This step will help us in building the social network from scratch or by

extending the existing one. We divide the degree of similarity into weak, average and strong cluster and degree of complementarity into yes and no cluster, each based on a threshold value. Each cluster might consist of multiple Web services based on the similarity value. All the nodes in the clusters are connected to a root node by edges.

4. **Edge weight initial evaluation:** Initially, each edge will be assigned a default weight equal to its degree of similarity/degree of complementarity for the new social network. The length of the edges is different in same and different clusters since the degree of similarity are different for each node in the social network. So, the length of the edge is calculated using edge weights multiplied by a constant factor. The constant factor helps to scale the edge length from low to high values as initial edge weights are very small.

5. **Social networks navigation:** It helps in navigating the nodes via edges present in the social network. Further, it calculates the cost of each node when this Web service(node) was selected. Now, a selection function decides which is the best mode for the user's request. The node that was selected is the most appropriate substitute, collaborative, or competitor peer.

6. **Edge weight ongoing evaluation:** Every time a Web service is discovered in these social networks, the weight of those edges is updated based on the type of social interaction (i.e. cost is increased). This ensures that any Web service present at any level can be promoted to a higher level (weak to average) or demoted to the lower level (strong to average). The step 5 and step 6 are cyclic. Both steps iterate until the Web service is discovered.

7. **Social networks ongoing management:** This is the final step of the proposed framework which reviews the shape of the social network. It ensures that edge's weight during node removal does not drop below a threshold, the Web services do exist, etc.

The paper also provides an end-to-end implementation technique based on above steps along with an example for better understanding. Thus, the paper presented a framework that weaves social networks to Web services discovery. It captured the social interaction types that existed in the Web services. This framework added extra functionality in Web services discovery.

This paper makes of Web Service social network component and navigation for quick Web Service discovery. The dynamic assignment of weights on social edges for searching relevant web services makes it an effective means of searching. The paper falls short in a way that the technique is not tested and compared with other registry based methods. Also this techniques only works if it correctly captures the Web Service relationships like substitution, competition and collaboration which is quite hard to achieve.

### 2.2.2 Discovery using user's social profile

Out of the number of research papers now published that deal with web service discovery with the use of their Social Profile or Social Networks, one such is [5] which makes use of FOAF (Friend of a Friend) based social profile and user's query in order to obtain the result. The use of the intimate social circle of a person is considered which is a novel approach considering it was not possible in the earlier days when social networks were in their inception stage. The paper initially covers the drawbacks that we encounter due to the centralized nature of the UDDI service and emphasizes the need for a distributed discovery solution. The primary flow of the paper distinguishes between the common social circle and the creation of an ego-centric social profile of a user which is used to restrict the number of options available. Static data such as name, age, etc. and dynamic data like interests, social activities, etc. are considered to find a suitable set of web services to be used.

The paper makes use of **XML and RDF** format primarily with the selection of social profiles done by the extension of a methodology known as **FOAF**. A similarity degree(SD) based on Jaccard's similarity coefficient is predetermined which finds the user's social circle and determines the ranking preference of web services for the given user. The coefficient can be calculated as:

$$SD = \frac{Ncfriend(u, v)}{Tfriend(u, v)}.$$

where Ncfriend are the mutual friends of the two users and Yfriend are the total number of friends. Also, a threshold index alpha is predetermined which decides the web services to be listed above a given user input value. The service discovery makes use of something known as an **epidemic protocol**. Finally, the web service related information is drawn from the WSDL documents. The architecture described by the paper is known as **Social Context based Web Service Discovery System(SC-WSDS)** which is based on **Service Oriented Architecture(SOA)**. Also, a new Algorithm which is the backbone of the given Architecture is introduced called as **User's Relationship and Past Invocation Discovery(URPI-Disc)**. Due to the lack of Social Networks that provide a ground for web service search, two independent datasets are used which are connected to a social network and which are used and invoked by the users. The paper thus provides a unique model which uses Social Network uses with FOAF methodology and considers importantly the ego-centric network of a user. However, we can also see a scope of improvement in the approach used for finding the ego-centric network of the user. The paper primarily is based on the reliability factor considering the suggestions based on close circle of friends of a person and can thus be having a good approach for search of web services.There is no feedback system as mentioned by the authors because of which it is hard to understand the performance that the paper achieves.

### 2.2.3 Web Service as a Global Social Network

The paper that we next took under consideration which undertakes the need for a unified social network of web services is constructing a Global Social Service Network for Better Quality of Web Service Discovery. According to the [2], the isolation of web services and also the lack of a global linking of these services with a lack of social relationships among services with the common connection are the reason for the slow uptake in the use of web services as compared to the growth of web technologies in general.

To overcome this, the authors decided to provide a Linked Social Services network as a Service. Commonly used ser-

vices today like WSDL, Web APIs and OWL-S largely treat web services as isolated islands with no inter-connection and no knowledge of the related set of services or the ones which can be commonly grouped with other such services. The authors address this by considering the issues of functionality and sociability. Though functionality is widely covered by services today, sociability is the issue addressed in this paper. QoS as a functionality covers performance characteristics like reliability, execution cost, security, etc. The aspect of sociability which was dealt earlier in Pedrinaci et. al. made use of URIs, HTTP URIs for searching data, standards like RDF, SPARQL for looking up URIs and including side links with these URIs. These were known as linked data principles but were found inadequate. To overcome this issue, linked social service-specific principles were introduced as a recipe for interlinking of services which is covered in the paper.

The Global Social Service Network is a social network with a structure of a directed graph

G = <V, E>

where V is each node being a linked social service and E is the directed social link for these nodes. The Quality of these social links is decided by factors like Dependency Satisfaction Rate, QoS Preference, Sociability Preference and Preferential Service Connectivity as parameters. The proposed solution is known as Link-As- You-Go which delves deep into the approach which can be adapted for establishing these links. It allows users to browse in one service and navigate further into social links with related services to explore service-to- service such that the users can explore global social service network deeply in the open web paradigm. By observing the shortest path between two given services in such a network, we can decide on the efficiency of such search methods by counting the smallest number of social links that a user has to navigate before settling for a service.

The paper also describes the utility of such a network in establishing a Global network which will facilitate a more dynamic search of services and provide more options. At the same time, emphasizes the need for factoring in social influence and feedback system later, though presently it is primarily based on the trust model. The idea of the paper and implementation is novel and very much pertinent to the issue of web service discovery. The lack of such social networks is a big impediment to the development of web and should be addressed in the near future.

This paper leaves the work of reducing the work of performing heavy tasks of data analysis. It has the advantage of improved discovery performance by linking services based on sociability.However, it does not perform the task of selecting the web service and neither does complete the automation service composition of the BDA process.

Although the three papers that we have covered so far [2] [5] [4] follow different approaches and methodologies, they are similar in use of Social networks in affecting the selection of web services. The initial selection of web services does not remain the same and evolves according to the input of the user primarily based on their social links. All the three papers use the social network but the purpose and the method of using social network is unique for each of them. The paper [2] uses nodes to represent the web-services and edges to define the related web-services while in paper [5] nodes represent both the web-services as well as the name of the users and edges are between the users and the web services used by those users. In paper [4], nodes represent the linked services and edges represent the factors like dependencies, satisfactory rate, etc. The paper [2] majorly focuses on the substitution, composition and other tasks on the web-services while the paper [5] resolves the issue of finding the related web services to the one web service the user has browsed in.

## 2.3 Semantic Approaches

### *2.3.1* Discovery by integrating multiple conceptual relationships

A semantic similarity measure integrating multiple conceptual relationships for web service discovery is a paper written primarily on probabilistic and semantic approach as encountered in English language for discovery of web services.

In the paper[1],it makes use of ontology like the is-a and has-a relationships and autonomy differences represented by **vector of terms**. The main idea behind paper is improvement in document classification or clustering using expert and intelligent systems. The paper mentions the lack of depth that is encountered in UDDI registry due to syntactic keyword matchmaking. A semantic-based approach instead of the more commonly used syntactic based method, known as **Semantic Web Service** description framework type such as **Ontology Web Language** for Services and Web Service Modeling Ontology is used. The advantage of such a technique is the contextual information extraction. For example, in the case of financial knowledge, semantic Web services Support Business Intelligence, executive information systems and uses financial concepts like cash flow, sale, profit, etc. The proposed semantic based approach is called as similarity measure integrating multiple conceptual relationships (SIMCR).

Conventional searching methods use metadata to match service requests, whereas this paper made use of **bipartite graphs** for comparison and techniques like **TF-IDF** (the number of times a term occurs in each document) and Vector Space Models. On a broader scale, the foundation is to estimate semantic similarity using concepts with means of similarity measures more than the metadata. Among others, these include **similarity measures between nouns and verbs and weighing of edges using has-a, is-a and autonomy relationships between words**. The paper also makes use of natural languages processing techniques like **Part of Speech tagging** and other methods finding the stem of each word like -ly, -ing, etc before making use of the above-mentioned bipartite graph for comparison. The primary metric for determining the efficiency of the algorithm is the use of Precision, Recall and F-measure which helps to evaluate the performance of web service discovery methods. The F-measure further integrates Precision and Recall in a single equation using composite harmonic mean. The formula for the same can be stated as below:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

A parameter epsilon represents the similarity of services above a particular value. The paper concludes with comparative study between SIMCR and other semantic based searching techniques and results are displayed for a given

dataset. Primarily addressing the need to implement such algorithm for scalable service discovery methods, the paper concludes with a satisfactory set of results.

Most of these methods are again related to the algorithms in Big Data about which we have mentioned in the first paper that we have reviewed.

### *2.3.2* Web service discovery using Bloom filter

This paper [10] replaces the traditional method of syntax and semantic-based method to find the most appropriate web service. The traditional model of the web service discovery is approached as a network routing problem and deterministic annealing is applied to facilitate service classification in the network construction. The reason for the **change of the approach** is because the **syntax and semantics-based service** matchmaking do not ensure **scalability**. Scalability is an important factor in today's world because of the increasing publication of the reusable data services on the Internet. To ensure the scalability in handling these ever-growing web-service publications different methodologies are required. Hence, the authors have decided to deal with this issue beforehand so that it does not become a very big issue in the near future and so they have applied network routing mechanism to facilitate service discovery so as to ensure scalability and performance. They have used bloom-filter which is a **space-efficient hash-based probabilistic data structure**.

Bloom filters are generated for all the services and the system they have designed is analogous to its working using the network of machines. Here, the role of machines is played by the services and the local network is analogous to the cluster of services. They are generated based on the information carried by the services. The service discovery is made possible by creating **virtual routers based on service clustering** and assigning addresses to those virtual routers. This is how the whole service discovery process is converted into network routing problem. The deterministic annealing method is applied to save bits which are used to build the bloom-filter addresses. This is done by leveraging the structure information of the OWL-S which helps to design the service network according to the service classification. The directory tree structure is used along with multiscale bloom filter. The hierarchical bloom-filter used for this approach is of 4-tuple which is a set of hierarchical keywords. In this system, the services are organized as a tree structure. There are a different number of address bits at each layer of the tree. The number of bits on each layer is decided based on the bits that are sufficient enough to differentiate all the nodes. In case the cluster contains many leaf nodes, semantic matchmaking approach is used instead of the Bloom filter. The services present in the network are assigned a bloom filter address. The incoming request is also converted into the address. The search for the web service starts from the root node and each layer of the network is checked to find the perfect match for the service. The search ends either at the leaf level or until a perfect match for the address is found in the tree. On, reaching the leaf node each node in the cluster is checked to find the most relevant web service which can be recommended to the user.

One of the drawbacks in the approach could be that as more number of web services are added the number of leaf nodes will increase which will affect the search performance of the system. To overcome this challenge, they have designed the system which predicts scaling functions and fits the predictions into connectivity distribution which describes the growth of the service network. There is a threshold value and if the size of the network is greater than that threshold value a new network is created with more clusters and fewer leaf nodes. This is how the scalability issue in the web service discovery is handled by the bloom filter approach.

Contrary to the previous paper, this paper too does not utilise semantic approach in analysing the web services. The previous paper [1] had cited the reason for the same as overcoming lack of depth for UDDI registries, while this paper utilises Bloom filters for semantic analysis to increase scalability. The data structure used for previous paper is bipartite graphs while the current paper makes use of trees for storing the services. Both the papers use semantic approach for matchmaking at certain point while implementing the algorithms.

## 2.4 Discovery in Mobile Framework

In paper[8], we reviewed was A Novel Web Service Directory Framework for Mobile Environments which accounts for the discovery of web services in an optimal way in the dynamic environment of mobile devices. The primary problems of mobile devices are minimal infrastructure, battery and network constraints, a limited computational power of devices and dynamic service registry management and use of conventional web service tools like SOAP (Simple Object Access Protocol), WSDL (Web Service Discovery Language) and UDDI (Uniform Discovery, Description, and Integration).

The centralized management of web services need frequent updating and volatility of such devices makes the use of legacy services inefficient. The proposed solution makes use of Service-oriented architecture (SOA) and deploying service registry over such mobile devices. The paper primarily substitutes UDDI with XMPP (Extensible Messaging and Presence Protocol) and is stated as appropriate for cooperative and personal web services hosted on mobile devices in local networks. As it is also hosted on personal handheld devices, it is important that it is lightweight, loosely-coupled and extensible. RESTful services are more suitable due to their flexibility. The implementation is done using XMPP roster management where a service keeps listening for all query requests and every time a service becomes unavailable, the registry agent gets notified and XMPP engine then updates the roster. The unique identification of devices is done by the Jabber Identifier (JID) which follows the format like:

localID@domainID/resourceID

where domainID is fixed for a local network and localID depicts the type of web service. Local service discovery is the main emphasis of the paper which helps address the dynamic nature of mobile environments and devices. Two types of service discoveries viz. Direct Discovery and Category based Discovery are performed using IQstanza of XMPP. The application of the paper is useful in volatile environments like war front activities, disaster scenarios and where minimal infrastructure is required.

This paper is almost unique in a way that it is the only one that deals with the mobile devices which is crucial in today's world. Though the implementation provided can have different approaches, it highlights the fundamental need for use of mobile devices in service discovery.

The mobile network framework as highlighted is useful for

remote areas or setting up a service network during some event. The downside for it however is setting up a web service roster on each device will eat up memory and space in handheld devices. Also the service registries though implemented on a local device still do not have option for customization according to the user which is fundamentally why such an implementation cannot be considered efficient.

## 2.5 Discovery using Mining and Multilevel index

### *2.5.1* Multilevel index for large scale repository

This paper proposes multilevel index model to reduce the time required to search and discovery the web services from the large-scale repositories. There are four functions defined in this paper which help to maintain services in the repositories.

In paper [9], they have replaced the traditional approach of considering the services in the composition of the services as a part of that composition instead of the individual system. Therefore, it is difficult to perform tasks like addition, deletion, updating of the services. Hence, with the increase in the number of services there is an increasing demand of the web services to be independent so that it can be useful and help to perform tasks like discovery, maintenance and management. The authors have helped to resolve this issue by adding the various operations and considering them as an individual system. They have included four levels of indexing which are **addition, deletion, replacement and retrieval**. There are four levels of indexing which are used for this purpose of which two are based on the input and the output parameters while two are based on the key to remove the redundancy from the inverted index.

The algorithms proposed in this paper are based on the **equivalence relations and quotient sets**. Service retrieval operation outputs the services based on the set of parameters provided as input by the user. The output is basically the subset of web services instead of the whole set of services which reduces the execution time of the search and the discovery process. The service addition, deletion and replacement are the maintenance operations which are use to maintain the web services in the repository. The theoretical foundation of the proposed multilevel index is equivalence based indexes which means clustering the similar web services into a virtual web service, thus reducing the search space of the web services. The another basis for the proposed multilevel index is redundancy removal index where they have removed the input-similar classes which are irrelevant to the user's request. The multilevel index modular has to determine the keys of the input-similar classes. This is also another basis for the proposed multilevel index. The various algorithms for the different operations like retrieval, addition, deletion and replacement is presented in the paper. Overall all this algorithms help to reduce the execution time for processing the large-scale service repositories using theoretical analysis and theory of equivalence relations and quotient sets.

Multilevel index model developed is more efficient than existing structures like sequential and inverted index structures and saves the time required for service discovery and composition leading to performance improvement. On the other hand, various factors such as number of input parameters for each service, number of services containing the same parameter as input, probability of each service not taken into consideration in details. The adjustment of addition operation to handle the above different factors maintaining the high efficiency not taken into consideration in depth.

### *2.5.2* Mining Interface Semantics for discovery

In paper[3], web services help in the reuse and sharing of the resources. Over the period, large number of Web services(WSs) have emerged and are stored in the WS library. Due to which accurately searching a WSs from the WS library has been a problem. The traditional way of finding WS over search engine is not efficient as WS are buried deep inside large number of web pages. Lot of WSs that are highly correlated gets missed as they are expressed in different structure which search engine cannot identify efficiently. This paper provides a unique way for Web services discovery(WSD) by mining the underlying semantic structures of interface parameters. Web Service Description Language (WSDL) document consist of information about the WSs types of parameters, operations, etc. that can help in searching WSs. This information is majorly not present in the natural language but rather in a coarse-grained description. So, this semantic information is not caught by normal search engines during WSs discovery. The proposed model catches parameter types, operation interfaces and other information from WSDL document. The model includes a 2-step process.

1. **Keyword Semantic Body (KSB) Mining:** This step extracts semantic information used for matching WSs from the WSDL documents. The set of keywords consisting of multiple words are created based on extracted names of operations performed by the WSs, parameter types, interaction messages, etc. which have similar semantic composition. This keywords sets provides a meaningful abbreviation's and words that can help in identifying a WSs. This keyword set also consists of meaningless terms which is to removed. Also, synonyms and word abbreviations of semantic terms can also exist. The pre-processing phase ensures that such meaningless terms are removed and provides an algorithm for mining keyword semantic body set (KSBS). The KSB is composed of set of terms that express different semantics. Based on the confidence factor, synonym terms get merged first then abbreviation sets that express the same semantic gets merged if it satisfies a predefined threshold value.

2. **Keyword Semantic Body (KSB) Indexing:** This steps introduces indexing between the semantic terms and WS operations. Due to indexing, searching of a WSs will be more efficient compared to the traditional way searching techniques without indexing. The structure of an index from semantic terms to WS operations (TermToOp) is <Term, OpList>. Term stands for the semantic term and is the key of the structure for unique identification. OpList is an index value that consist of a list of structure <OpID, LocationMark>. The OpID identifies the WS operation to which the term belongs and LocationMArki is a designation number. It provides algorithms for semantic body operation index set establishing and TermToOp set establishing in detail.

The above 2-steps establishes semantic mining process. Later the paper provides an WSs operation discovery algorithm for

integrating with the semantic mining process. This model was tested on a dataset of 10,084 description files. The cleaning of the dataset was done to remove duplicates WSs. Two test were performed for this proposed model. The first test was time cost comparison and the second test was comparison of top-k precision ratios. The results of both the test were impressive. This model used index based approach to perform time efficient search operations on WSs registry. It also provided a method to search composite WSs based on user's request.

The paper helps in the excavation of undiscoverable buried web services which might be less correlated with others which helps in identifying some unique services which may be useful. The information present in a coarse-grained description which is semantic information not caught by normal search engines during WSs discovery is searched by the means suggested in the paper. As a unique paper it also provided a method to search composite WSs based on user's request which makes it a really good implementation. The mining performed in the description model however is based on the efficiency of the algorithm used and an inefficient algorithm can give misleading results.

### 2.5.3 Diversify services and discovery using probabilistic approach

The paper [6] proposed a ranking algorithm for diversifying Web services (WSs) discovery and composition. This will minimize the redundancy in the WSs search results. The ranking algorithm ranks the WSs based on its relevancy, service density and service diversity. The WSs composition has always have been a problem and this paper provides a solution to this problem. It selects dissimilar WSs from number of top-N ranked WSs based on its relevancy, service density and service diversity. The paper provides 3-steps to achieve its goal.

1. **Re-ranking default discovered WSs search results:** This step uses probabilistic WS matching and ranking technique. It first discover implicit topics related to the user query. A topic is a word or a concept present in the WSDL document that provides information about WS and which can be expressed in probabilistic distributions. A probabilistic distribution equation is used to compute similarity of user query with all the topics. Top-K implicit topics are selected. Now, WSs related to the Top-K topics can be easily discovered. This discovered WSs are re-ranked based on the probability of relevancy, density and diversity.

2. **Generate service dependency network using Formal Concept Analysis (FCA):** The Web service composition consists of many WSs to satisfy the users request which cannot be satisfied using single WS. All WSs in a composition should be able to integrate with input and output values of other WSs in the composition. It is difficult to find WSs that can integrate with each other. To solve this problem, a service dependency network is built using FCA. It consists of a directed graph with nodes as services and edges are the links between the nodes if the operations of that node satisfy the dependency condition. After building the graph it provides an algorithm for creating operation dependency network using FCA.

3. **Finding and Selection of optimal composition for discovered WSs set:** This step finds and selects WSs that satisfy the user requests which is done by implementing an extended DivRank algorithm. An experiment was performed on real world WSs to test and measure the performance of the proposed model. The result of the experiment indicates that the proposed model improved the WS recommendation performance in terms of the precision when properties like relevance, diversity and density of WSs are considered.

The reduction is redundancy must be playing an important role in user satisfaction level and this is why the paper is a good alternative to service discovery. This also leads to performance of web service recommendation in terms of precision.

Among all the three papers, [9] and [6] both address the same problem of reducing the redundancy in the web-services search results. The paper [9] is more focused on the theorems while [6] is inclined towards the practical application of the concept of indexing. The [9] makes use of four level of indexing while [3] uses single level of indexing and is focused on using semantic based approach for indexing. The paper [9] addresses the problem of adding, deleting and updating individual web-services which are part of a composite web-services while [3] addresses the problem of finding the web-services stored deep inside large number of web-pages.

## 3. IMPLEMENTATION

Implementation Steps for User's Social Profile-based Web Services Discovery [5]

1. **Built social relationship filtering and User past invocation history:**

   We have successfully created social model for User using Neo4j (Graph Database). The social model consists of 10 User nodes and 50 to 60 Web Service nodes (This number will change as we get more web services WSDL files). Each User node has **name** , **age** and **hobby** as property where as each Web Service node has **name** , **end-point** , **description** , **link** and **category** as property.

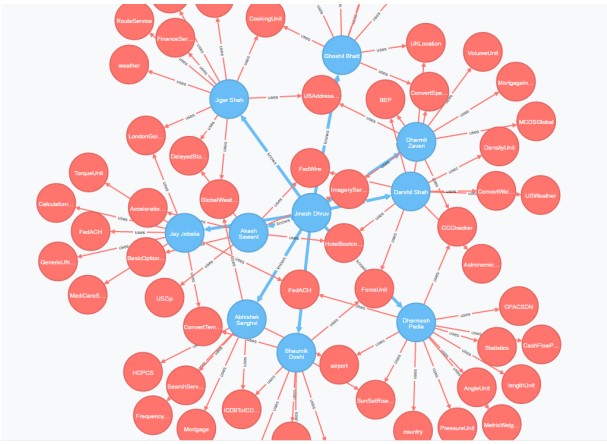

**Figure 1: Ego-centric graph**

[H] The social graph consists of 2 types of relationships which are User-To-User(UTU) and User-To-Web Services(UTWS) relationships. In UTU relationship, the user 1 **knows** user 2 (i.e. both the users are friends). Each relationship of **knows** has **mutual friends** as property. In UTWS relationship, the user uses one or more web services (i.e. User node is connected to Web Service node if and only if that user has in the past used this web service). Each relationship of user has **UsedCount** as property (i.e. How many time in the past the user has used this web service). We created a synthetic social model as such type of dataset does not exist. Using the paper as reference we created synthetic graph database using Neo4j. We created 10 user nodes with UTU relationships. We created 5 categories of Web Services and searched the different web services that comes under this category. After getting WSDL links for different Web Services, we wrote a code which fetches the WSDL document and extracts the required information from the document. Using this information, we created Web services nodes and random relationship uses between the web service node and different users. This way we have built our user social model which will be our input. The social graph database was created using Neo4j Java drivers.

2. **Matching user Query and web services:**

   We have successfully implemented the algorithm mentioned in the paper for matching the user query and web services using friend of a friend (FOAF) model in Java. The system will take user query and graph database as input. Based on the user query, first the User past invocation history will be checked, if user had used any web services that matches the query then it will be displayed. Also, all the close friends of the user are extracted from the graph data base. The close friends are the ones which have maximum number of mutual friends. The code will extract close friends which has mutual friends greater than a threshold value. After getting a list of all close friends, all the past web service invocation history of close friends is searched. Only, those web services that matches the user query are extracted and shown to the user.

3. **Web service ranking (Use of alpha similarity degree):**

   We have successfully implemented this step of ranking the extracted web services from the previous step in the appropriate order. The extracted web services will be displayed to the user based on the alpha similarity degree. The extracted web services of the closest friends will be ranked based on the number of mutual friends count. After showing the result to the user, if the user selects and uses any of the recommended Web service then its entry will be made in the graph database (i.e. A relationship link will be created between the user and the newly used web service)

## 3.1 Results

The three methods which are used for evaluation are Precision, Recall and F-Measure. The formula to compute the above metrics are as follows:

$$Precision = \frac{R(U)}{N(U)}$$

$$Recall = \frac{R(U)}{M(U)}$$

$$F - measure = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

where R(U) is the number of relevant web-services to the user u from all the retrieved web-services, N(U) are the total number of retrieved web-services and M(U) is the sum of relevant web-services to the user u from all the retrieved web-services and the number of relevant web-services which are not listed to the user.

The comparison can be seen in **Figure 2** below. We can observe from the graph obtained for the ten users that the value of F measure, Precision and Recall comes out to be consistently higher for FoaF based search. The **Figure 3** below shows the Output of web services that we can obtain from the three methods. As we can see, the number of outputs that we obtain are lower and more relevant in case of service discovery by FoaF based search. This is better than that we obtained from the UDDI search as it will list lots of results or very few depending on the choice of keywords.

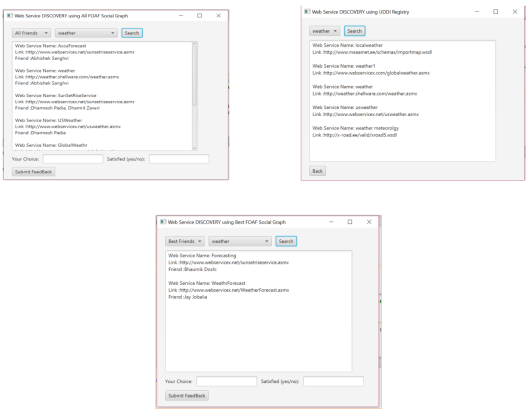

**Figure 2: Output Result for the three methods.**

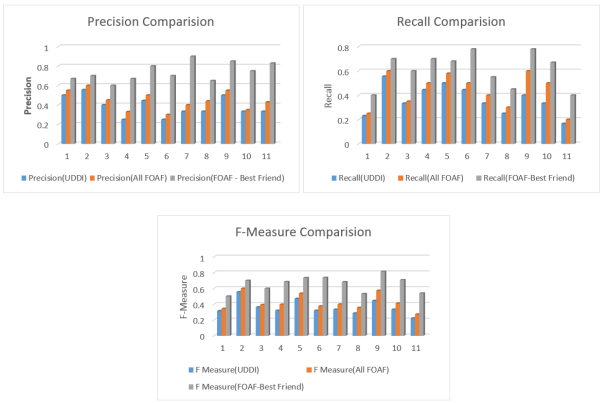


**Figure 3: Evaluation Result for the three methods.**

# 4. CONCLUSION

We have successfully implemented this paper[5]. This paper proposed a system which uses user social network and user query to produce satisfactory web service discovery results. This process takes user query and user social network as input. Now, close friends from user social network are found using similarity degree. The web services relevant to the user query are searched across the extracted list of user close friends using their past web service invocation history. The web service discovery using this technique produced better results compared to the traditional UDDI based discovery. Apart from implementing the entire paper[5] we also added some new functionality and found some areas where more works needs to be done for making robust system. In our modified system, user can externally add new web services in its past invocation history. Also, we found that the paper[5] finds close friends based on only higher mutual friends between the user and the friend which is not the best method. It is possible that best friends can have zero or less mutual friends. For example, if two persons are friends and they belong to the same university and have more friends from the same university, now it is possible that the two persons have large number of mutual friends but this does not indicate that they are best friends. So, number of mutual friends and some other attributes should be used to find the close friends and not alone mutual friend count. For evaluation of the system, we compared the web service discovery results obtained using traditional UDDI, FOAF and close friends in FOAF. The results showed that close friends in FOAF produced the best results.

# 5. FUTURE WORK

1. Enhance the existing system by adding new functionality for handling complex user queries.

2. Develop the web service management mechanism which removes outdated web services from the network, add more users and web services dynamically to the social graph.

3. The web service management mechanism will also check whether the user and web service already exists in the system before adding them to the system.

4. Build a logic with which we can find close friends not only by considering mutual friends count but also other properties (i.e. Improve similarity degree formula).

# 6. REFERENCES


[1] F. Chen, C. Lu, H. Wu, and M. Li. A semantic similarity measure integrating multiple conceptual relationships for web service discovery. *Expert Syst. Appl.*, 67:19–31, 2017.

[2] W. Chen, I. Paik, and P. C. K. Hung. Constructing a global social service network for better quality of web service discovery. *IEEE Trans. Services Computing*, 8(2):284–298, 2015.

[3] B. Cheng, S. Zhao, C. Li, and J. Chen. Misda: Web services discovery approach based on mining interface semantics. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 332–339, June 2016.

[4] N. Faci, Z. Maamar, H. Abdeldjelil, and D. Benslimane. Towards a framework for weaving social networks principles into web services discovery. In *2011 11th Annual International Conference on New Technologies of Distributed Systems*, pages 1–8, May 2011.

[5] A. KalaÃŕ, C. A. Zayani, and I. Amous. User's social profile – based web services discovery. In *2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 2–9, Oct 2015.

[6] H. Naim, M. Aznag, M. Quafafou, and N. Durand. Probabilistic approach for diversifying web services discovery and composition. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 73–80, June 2016.

[7] T. H. A. S. Siriweera, I. Paik, J. Zhang, and B. T. G. S. Kumara. Big data analytic service discovery using social service network with domain ontology and workflow awareness. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 324–331, June 2016.

[8] R. Verma and A. Srivastava. A novel web service directory framework for mobile environments. In *2014 IEEE International Conference on Web Services*, pages 614–621, June 2014.

[9] Y. Wu, C. Yan, Z. Ding, G. Liu, P. Wang, C. Jiang, and M. Zhou. A multilevel index model to expedite web service discovery and composition in large-scale service repositories. *IEEE Trans. Services Computing*, 9(3):330–342, 2016.

[10] J. Zhang, R. Shi, W. Wang, S. Lu, Y. Bai, Q. Bao, T. J. Lee, K. Nagaraja, and N. Radia. A bloom filter-powered technique supporting scalable semantic service discovery in service networks. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 81–90, June 2016.