IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 32-43

Accurate and Faster Web Service Discovery Mechanism based on Conceptual and Semantic Meaning

M.Krishnamoorthi[1], M.Lingaraj M.Sc.,M.Phil[2].
Research Scholar, Department of Computer Science, Sankara College of Science and Commerce, Saravanampaty, Coimbatore, Tamilnadu, India
Assistant professor, Head of the department, Department of Computer Science, Sankara College of Science and Commerce, Saravanampaty, Coimbatore, Tamilnadu, India

Abstract: *Web service discovery has received the most focus in the research field practically. Developing an accurate web service that can meet the user demands is a sophisticated task that also takes a vast amount of time. The web service discovery process has to be carried out with accuracy and rapidity so that the performance is improved. In the already available research work, it is performed employing a technique known as Web Service Operations Discovery Algorithm (WSODA). The web services yields the same functionality but has adiverse feature. Quality had become a significant challenge, which was not taken into consideration in the available work. Moreover, the time taken for discovering the web services is more in the system available. This is solved in the newly introduced research approach by presenting the technique known as Accurate and Faster Web Service Discovery Model (AFWSDM). In this technique, at first, a conceptual web services description model is presented, where the type path is included for the interaction interface parameters along with the conventional text description. The extraction of all the content in the web service model from the basic description files can be donewith straightforward XML tag extraction techniques. Optimization of a web service is performance-based and it is measured by Quality of Service i.e. QoS that is inclusive of Response Time, Availability, Throughput, Success ability, Reliability, Best Practices, and Latency. The QoS factors are measured by utilizing the Firefly Algorithm (FA). The k-mean clustering algorithm is utilized for the clustering interaction interface names and partitions under the supervision of co-occurrence probability. At last, a web service operations Discovery algorithm (OpD) is proposed. The experimental evaluation indicates that the newly introduced technique exhibits a better performance in comparison with the available research works.*

Keywords: *web service discovery, accuracy, precision, interface, conceptual meaning, semantic extraction.*

I. INTRODUCTION

Web services are defined to be self contained and self descriptive applications, which can be published, found and called over the web [1]. These are basically XML based components, which can be executed by any application on the World Wide Web, with no regard of the platform [2]. The primitive web services can be integrated to deal with sophisticated requirements to create value added composite services. Web service yield a platform, which permits for interoperability between software applications that run on various platforms and frameworks [3]. The implementation of Web services are done with standards like UDDI, SOAP, WSDL, etc [4]. Web services are designed and published by various vendors employing UDDI. It is the technique used for registering and discovering the web services. The details about a web service are given in the WSDL document. It renders the format to define the web service and the way in which they are tied to a network address. Definitions, operations and service bindings associated with web services comprise the WSDL components. XML is utilized by WSDL for expressing the definitions of a web service [5]. The web service Operations are of four kinds including one way message sent without a reply, simple request and reply, solicit response and sending notifications.

Web services are accessed from the internet via SOAP [6]. With the expansion of the abbreviation as Simple Object Access Protocol, this helps the programs, which run on various operating system to communicate utilizing HTTP and XML. SOAP has the responsibility of finding the right and effective web service. The input, output, preconditions and effects mentioned by the user are exploited in the discovery of the web service [7]. QoS parameters are utilized for ranking the web services discovered and the best one is chosen. SOAP is dependent on XML, and it communicates through the internet, platform/language independent, snoop around firewalls and can be extended [8].

It is vital to observe that the concept of a service is actually a logical one instead of being a physical concept. In order to achieve efficiency, a so-known container of a virtual hosting environment like the Apache Tomcat servlet container may be utilized to executed more than one service or interface in the same process or thread [9]. The service interfaces of a service may, but not mandatorily, be implemented on the same host. They may be distributed over a number of hosts across the LAN or WAN and even find their footprint in administrative domains. This idea lets defining about a coherent interface bundle in an abstract fashion irrespective of the physical implementation or deployment choices. It mentions about a distributed (local) service, if it is known and there is a need to emphasize that the service interfaces are in fact implemented across multiple hosts (or on the same host). Generally, a service is consistent (long spanned), but it may also exhibit transience (short lived, momentarily instantiated as per the user's request).

The important objective of this research is to present the system that can optimally find the web services from the web domain satisfying the user demands. This is accomplished by presenting the different methods and concepts, which can guarantee the optimal and robust extraction of the web services. The research work is organized on an overall as below: This section provides an elaborate description about the web service discovery process and their requirement. Section 2 explains about different relevant research approaches in terms of the extraction of the optimal web services along with their operational procedure. Section 3discusses about the proposed research approach with necessary examples and description. Section 4 describes about the simulation results along with the percentage evaluation. At last, Section 5 provides the conclusion of the research work on the basis of the simulation result achieved.

II. RELATED WORKS

Tsai et al [10] studied about the addition of verification strategy to the UDDI servers including check-in and checkout of Web services. The core concept is that the test scripts has to be joined with the Web service, and both the Web service providers and clients use these test scripts, prior to admitting a new Web service into the service directory. This new Web service has to be tested by the respective test scripts, and admitted when the test was a success. Prior to the usage of a particular Web Service, a client can make use of the appropriate test scripts for testing the WS and it is utilized only when the test was successfully performed. This technique is a process that consumes time as both the service provider and the requester has to make use of the test scripts.

Liu et al [11] introduced a domain-oriented UDDI Registry architecture and dealt with concepts including service property schema for every service type that generates a property table and stores the service property information and service relationship in the database. It is further classified into Complementary relationship, functional relationship, reference relationship and service constraint that are created by the extraction of the binding information and service interface definition files.

Wu et al [12] introduced Semantic Web service, which makes use of DAML-S in the form of a substitute of WSDL to denote the Web services' Capabilities. The changes, which let the software agents or search engines to automatically get appropriate Web services through Ontologies and reasoning algorithms with modern techniques are proposed. For every category, a similarity evaluation technique has been provided. In Web service match making process, these similarity assessment evaluations can be used in conjunction or autonomously. A set of similarity techniques that can be utilized combined with the current UDDI API and for supporting a more automated service-discovery process have been provided, by identifying among the potentially helpful and the probable irrelevant services through the ranking of the candidates based on their significance.

Tretola G and Zimeo E [13] suggested a novel strategy for improving the key word based search and syntactic match known as structure match. In this technique, the semantic of services is perceived from their structure and it is chiefly focused on its operation and parameter kinds. This technique discovers the similarity between the requested service and the set of services available. The authors have realized a novel algorithm that falls under match making framework using one or more match making approaches with cascading pipes and filter architecture with the aim of improvising the results of matching.

Bener et al [14] devised a super peer network protocol for combining the centralized and Peer-peer networks. The chief intent of this research work is to reduce the number of messages that are routed through the network and to prevent flooding in the networks, and the authors have suggested CAN structure(Content Addressable network) that contains distributed hash table for P2P communication. The newly introduced architecture comprises of self-clustering networks, where the peer groups categorize the definitions of web services and also every peer becomes the owners of the classification in a dynamic fashion. Wu et al [15] studied about a model that partitions the entire UDDI servers in to three kinds including root, super domain, and normal servers .The Root server performs the recording of the information of super domain servers, and the function of super domain servers lies in the management of the adjacent servers and it is accountable for registration and inquiry services and finally the UDDI servers allows the users to publish and search for the information of web services.

Nawz Et al [16] suggested a push model for the discovery of web services. Here, in this model, a service notification is provided to the service requestors before the web service discovery is done. The authors have made use of semantic based web service matching, where the service descriptions are matched with OWL-

S (Ontology language for web service description) .They have classified the system into two stages, namely the Subscription phase in which a subscriber gets him/herself registered into the registry for notification purposes and the notification stage which functions once a new service gets introduced on the registry. Liang et al [17] has introduced an effective concurrent access control approach. The method consists of optimistic time stamp ordering based synchronization. This scheme lets the non-conflicting queries to run simultaneously and in the case of conflicting requests, the queries get buffered. Tewari et al [18] have employed the classification approach and carried out a validation test on web services. The system is used with clustering data approaches. With this approach, the system is able to suggest the second best suitable service to the customer and in addition, the authors have exploited reviews and ratings given for the available service registry to enhance the web service request with service information. Ma. et al [19] has suggested a novel web service access and discovery mechanism that integrates the search engine approaches and semantic web concepts. A syntax level keyword matching is proposed and in addition, semantic information is added to the services, and the results achieved from the service result list is then weighed by keyword matching and service semantic vector for efficiently retrieving the services.

Accurate and Faster Web Service Discovery Model

Web service discovery is the most typically observed problem in the real environment that tries to meet the user demands in a trust worthy manner. In order to attain this, first a conceptual web services description model is proposed in which the type path for the interaction interface parameters is included along with the conventional text description. The extraction of all the content in the web service model can be done from the elementary description files employing direct XML tag extraction techniques. A web service's optimality is dependent on its performance and it is measured by Quality of Service i.e. QoS that is inclusive of Response Time, Availability, Throughput, Success ability, Reliability, Best Practices, and Latency. The QoS factors are measured with the help of Firefly Algorithm (FA). The k-mean clustering algorithm is utilized for clustering the interaction interface names and classifies under the monitoring of co-occurrence probability. At last, a web service operations Discovery algorithm (OpD) is proposed. The elaborate description of the research approach is provided as follows

Conceptual Web Service Description Model

In order to clearly explain about the Web service, the conventional description model is extended by importing the type path of the interaction interface parameters. Most of the Web services on the Internet yield very less description information and just yield the interaction parameters; they do not render function descriptions. The tags defining the operations and parameters are even rare in Web service description files in WSDL. At the same time, the operations and their parameters yield more elaborate interaction details compared to the description itself. Hence, the most efficient information for expressing the interaction reduces to the names of operations and their parameters. Several researchers have considered operation names and parameters to be significant objects used for semantic mining. But, the information present in the names is quite less, rendering it hard to mine and use their semantics. Through the analysis of the Web services description files WSDL, it is observed that the type paths for interaction parameters also have significant semantic information, and can aid in applying the semantics pulled from the interaction interface parameters. Consider a weather forecast to be an example. The single arrows define the traces of explaining the types and parameters. For instance, the parameter name "Place Name" refers to an element parameter name for the class type "Weather Forecast" and "Weather Data" stands for an element type name for the class type "Array of Weather Data." Extraction of just the parameter names, like "Place Name" does not much captures the truth that this service renders weather forecasts. Hence, the type paths of interaction interface parameters is added to the conceptual models. At first, the parameters type path is formalized. The parameter path pPath is a type-specifying path expressed as:

$$pPath = <p_1, p_2, \ldots, p_n>$$

where p_i refers to the parent node of P_{i+1}, if i=m, and p_i stands for the name of the parameter, where p_m is named as "Leaf Node;" else, p_i refers to the type name of the parameter, therefore it is named as "Parent Node."

The type set ATOM is imported to help in deciding if a parameter is a Leaf Node. ATOM is the fundamental XML type set like "int," "string," etc. ATOM is basically an XML Schema. In case the type of a parameter belongs to ATOM, then it is a Leaf Node; else, it is a Parent Node. For instance, in Fig. 1, the $\mathcal{P}path$ of "Place Name" is "Get Weather By Zip-Code Soap Out. Get Weather By Zip Code Response.

Weather-Forecasts. Place Name," and the$\mathcal{P}path$ of "Min Temperature C" is "Get Weather By Zip Code Soap Out. Get Weather By Zip Code Response. Weather Forecasts. Array Of Weather Data. Weather Data. Min Temperature C."

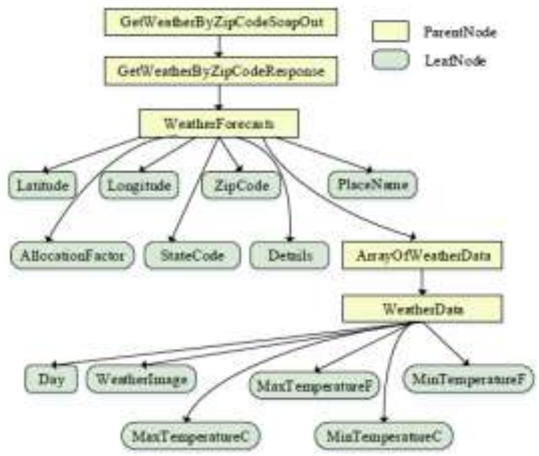

Fig. 1: Weather forecast output interface parameters

It is also noticed that not all the "ParentNode" nodes has resourceful information. Few "ParentNode" nodes have information full of redundancy, like "ArrayOfWeatherData" and "WeatherData," and few "ParentNode" nodes have meaningless information, like the "of" in "ArrayOfWeatherData." Hence, the $\mathcal{P}path$is different from class diagrams in UML or an OWL-S Profile, and it is applied only to help in mining the interface semantics.

XML Tag Extraction Method

An XML-relational mapping scheme is utilized for creating a relational schema associated with the "filtered" hierarchy of an XML document. Truly, both an XML document and a relational database can be seen as trees. In Figure 2, a tree representation of the XML document below, such as SampleFlow.xml3 defining workflow steps (i.e., tasks) is shown.

```
<?xml version = "1.0">
<!DOCTYPENetworkTask SYSTEM "NetworkTask.dtd">
<NetworkTask id = "1">
 <Task id = "2">
 <Name>SampleFlow</Name>
 <TaskType> Non-transactional workflow </TaskType>
 </Task>
 <SimpleSubTaskList>
 <Task id = "3">
 <Name> Start </Name>
 <TaskType> Human </TaskType>
 </Task>
 <Task id = "4">
 <Name> Close </Name>
 <TaskType> Transactional </TaskType>
 </Task>
 </ SimpleSubTaskList>
</NetworkTask>
```

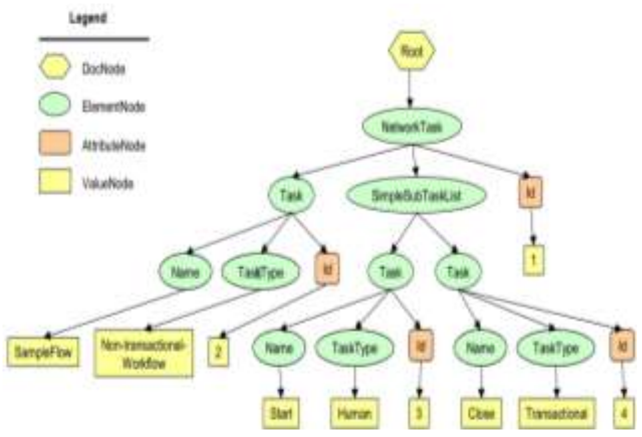

Fig. 2: A Tree representing the XML document hierarchy

Filtering is implemented in terms of (1) mapping just the "chosen" elements to relations, and (2) mapping only "chosen" elements or attributes to relational attributes. To attain (1), TABLE and TABLE_MAP structures are utilized for defining the relations and paths to get the respective elements in the XML tree.

```
public static final String [ ]
TABLE = {
"xmldoc", "net work task", "task", "dataobject", "roledomain", "wf role"
};
public static final String [] TABLE_MAP = {
"", // special root table (no data from XML document itself) "Network Task Task", "Network Task Simple Sub Task List Task *", "Data Class", "Role Domain", "Role Domain Role *"
};
```

For instance, the path for a network task is "Network Task Task"4 (the orange path in Figure 3), and the path for a task is "Network Task Simple Subtask List Task *" that is shown as a turquoise path. Star (*) indicates that task is multi valued. Extractor makes use of these specifications for the recursive traversal of XML tree structure from the Root downward making use of the interfaces and classes created by DXML. The elements, which will be extracted, and shown in Figure 3, comprise a filtered subset of elements in the tree and are not mandatorily the leaf nodes of the paths (e.g., Network Task). It is to be noted that the entire extraction algorithm is not explained here, due to space constraints.

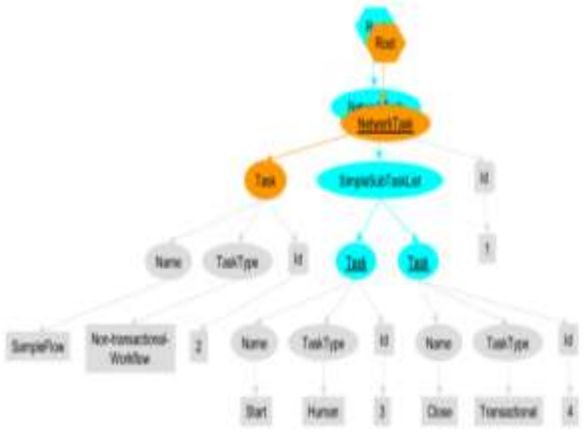

Fig. 3: Specification of tables and table map

The property (2) above is implemented by filtering just a chosen subset of attributes corresponding to the leaf nodes of the paths. In fact, these paths comprise the lattices having a single start node and a single termination node when just one of the multivalued elements is kept. Therefore, each attribute/ element of a termination node is really correlated to the start node 4 that indicates a relation. For instance, Name and Task Type refer to the children of a leaf node (i.e., Task) and comprise of the filtered elements and attributes for

NetworkTask relation (Figure 4). It is to be noticed that these elements/attributes are not correlated to their direct parent (i.e., Task), as it is not filtered in the form of a relation in the earlier step.

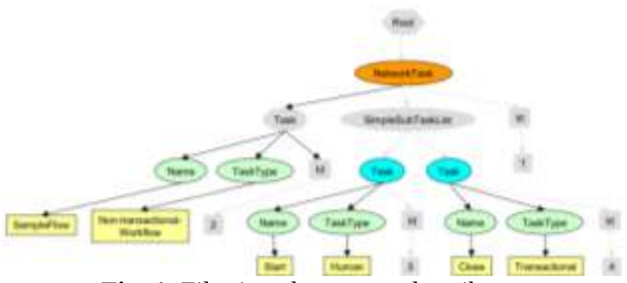

Fig. 4: Filtering elements and attributes

The two filtering approaches ((1)&(2)) let the users to manage the generation of database schemas and yield an effective storage approach for the chosen components, as rather than the entire document, the essential portions of the document for a user are defined and then extracted.

Quality of Service Aware Web Service Selection

Web service description language (WSDL) has rendered a standard model for specifying the service functionality by isolating the abstract definitions of service input and the output messages from the concrete descriptions of every end point's associations. But, currently there is no standardized description framework available for including all of the aspects of non-functional characteristics of the service. In spite of the absence of standardized framework, examining the associated solutions on the basis of a persistent view of non-functional service characteristics is still a necessary task. This section models the associated work to study about a QoS model of Web service that can be utilized for discussing the proposed service selection approach here in. It is to be noted that this research work does not intend to show the non-functional issues of Web service with an elaborate list consisting of QoS attributes, and introduces their respective definitions in a realistic way.

Response Time: The maximum time, which is elapsed from the moment that a web service gets a SOAP request until it generates the respective SOAP response. It is computed as

$$RT = T_1 - T_2$$

Where T_1 = Time at which web service generates the soap response.
T_2 = Time at which web service gets the soap request.

Execution Time: The execution time provides a measure of the the anticipated delay between the time when a request is sent and the time instant when the result are obtained. It is represented by

$$q_{du}\ (s, op) = T_{process}(s, op) + T_{trans}(s, op)$$

Throughput: The number of Web service requests *R* for an operation *o* , which can be processed by a service S within a certain time period.

$$tp\ (s, op) = \frac{\#R}{timeperiod}\ (\text{in second})$$

Where,
S service within a certain time period.
#R number of web service request.

Scalability: A Web service, which is scalable has the capability to not be overloaded by a huge number of parallel requests. It is computed as,

$$SC = \frac{t_{rt}}{t_{rt}(\text{throughput})}$$

Where, t_{rt}(throughput) refers to the round trip time that is assessed during the throughput test.

Reputation: The reputation of a service provides the measure of its reliability. It is chiefly dependent on the end users experience of the service. Various end users may have diverse preferences on the same service. The value of the reputation is defined as the average ranking provided to the service by the end users. It is computed as,

$$q_{rep} = \frac{\sum_{i=1}^{n} R_i}{n}$$

Availability: Availability of the web service is defined as the probability of the service's being available. It is computed with the expression below,

$$q_{av}(s) = \frac{T_i(s)}{n}$$

Accessibility: It is defined as the capability of serving to the Web Service request. The Web service might be available though not accessible due to excessive volume of requests. Accessibility can be denoted by the formula below:

$$P_{accessability} = \left(1 - \left(\frac{downtime}{uptime}\right)\right)$$

The unit of time measurement is minutes. In this research work, the firefly algorithm is used for selecting the most appropriate web service that can meet the QoS factors that are above mentioned.

Firefly Algorithm

The Web service selection approach is defined as the process of choosing single or composite services. The selection approach makes use of the service framework and service classes from the other related service. As per the user requests, the services having the same functional characteristics and diverse non-functional characteristics are integrated to optimize the performance. These non-functional characteristics are associated with weight vectors that contribute more to the process of selection because when the weight vector differs, the result may also vary in accordance, therefore the differing weightage vectors are optimized in order to provide the best services possible in various iterations employing firefly algorithm. The process of Web service selection begins with service discovery stage, which is followed by selection step, where the web service selection is performed with normalized QoS matrix and FA optimized weight vector.

Input: Input and Output of the web services and seven QoS attributes (Response Time, Availability, Throughput, Success ability, Reliability, Best Practices, and Latency)
Output: Best web service corresponding to the requirements of the user in addition to optimized weight vectors for the best service.
Step 1. Data set upload
Step 2. Receiving the input from the user
Step 3. Service discovery: Depending on the web service Input and web service Output, the services were filtered, i.e., matching of has Input and has Output tag values in each of the Owl files of web service is got and just the precise match was considered to be filtered result.
Step 4. QoS matrix is created using the filtered web service QoS attributes. QoS matrix refers to the nxm matrix where n refers to the no of web services (i.e., rows) and m stands for the types of quality attributes for every service (i.e., columns). Normalization of QoS Matrix: QoS matrix was normalized with qmax and qmin values and at last it is formulated within the range [0-1] where qmax=max value of the quality attribute, qmin= min value of the quality attribute in the QoS matrix.
Step 5. Applying weight vector over the normalized matrix utilizing FA algorithm

```
Initialize a population of n firefly's positions randomly.
Get the best solution depending on fitness.
While stopping criteria not met do
For each Firefly$_i$ do
For each Firefly$_j$ do
If firefly j is brighter than firefly i then
Move firefly i towards firefly j employing equation (1).
Else
End
Assess the positions of every firefly.
End
```

Clustering Interaction Interface Names

The Interface Semantic Mining module performs the mining of the interface semantics within and generates the index library depending on the results of mining. An underlying interface semantic mining technique is proposed, which is inclusive of synonyms, abbreviations, and fusions of un-sequenced fragments depending on their greater probability of co-occurrence. Making use of the results of mining, an indexing system is created for searching the results from a number of Web services rapidly and get compatible operations. By making use of statistical approaches on Web services interface parameter names, it is evident that most interface parameters are fusions of synonyms, abbreviations, and disarranged fragments. For instance, the interface for expressing the name of a place can be expressed as "*PlaceName,*" "*PlaceN,*" "*locationName,*" etc. These fusions generally decrease the precision/recall rate in Web services discovery matching operations. The name strings are not always generated straight from rightly spelled words, but they can be interpreted by most of the humans; but, they are hard for computers to interpret.

Being humans, we can recognize synonyms, abbreviation, and disarranged fragments and then substitute them with entirely spelled words or synonyms and sort them in a sensible order, there by facilitating an interpretation of the interface parameter names. Human beings generally possess a "consensus library" that constitutes of dictionary vocabulary words and usage habits (like abbreviations). This consensus library acts in the form of a metrics database to yield a base for interpreting the sophisticated constructions like interface parameter names that generally do not have only vocabulary words. Hence, this interface clustering technique tries to get this "consensus library" depending on dictionary vocabulary, substrings of fragments having a higher co-occurrence probabilities, and divide the combination series with greater probabilities of co-occurrence. The high co-occurrence probability is applied to show how people most frequently make use of the terms. The whole process of clustering happens under the monitoring of high co-occurrence probability. Name fragments are extracted, dividing the results into elementary clustering units. Thereafter, the set of synonym fragments is found utilizing the lookups to a synonyms dictionary like WordNet. In this research work, k mean clustering is applied for carrying out the interaction interface clustering.

k-means is one among the easiest unsupervised learning algorithms, which resolve the popular problem of clustering. The procedure uses a simple and convenient means of classifying a data set given through a particular number of clusters (suppose k clusters) fixed apriori. The chief concept is to specify the k centers, one for every cluster. These centers has to be placed in an intelligent manner as diverse locations results in diverse results. Therefore, the better option is to keep them far to the maximum extent possible from one another. The next subsequent step is to take every point that belongs to a data set given and then correlate it to the closest center. If there is no point pending, then the first step is finished and an early group age is done. At this juncture, k new centroids has to be re-calculated as bary center of the clusters resulting from the earlier step. Once these k new centroids are obtained, a new binding has to be performed between the same data set points and the closest new center. A loop has been created. Due to this loop, it may be noticed that the k centers modify their location step after another step till no more modifications are performed or to be said in other words, there is no movement of the centers any more. At last, this algorithm targets at an objective function minimization known as squared error function expressed by:

$$J(V) = \sum_{i=1}^{c}\sum_{j=1}^{c_i}(\|x_i - v_j\|)^2$$

where,

'$||x_i - v_j||$' refers to the Euclidean distance between x_i and v_j.

'c_i' stands for the number of data points in i^{th} cluster.

'c' refers to the number of cluster centers.

Algorithmic Steps for k-means Clustering

Let $X = \{x_1,x_2,x_3,\ldots\ldots..,x_n\}$ refer to the set of data points and $V = \{v_1,v_2,\ldots\ldots.,v_c\}$ indicate the set of centers.

1) Randomly choose the 'c' cluster centers.

2) Compute thethe distance between every data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum among all the cluster centers.

4) Re-compute the new cluster center applying:

$$v_i = \left(\frac{1}{c_i}\right)\sum_{j=1}^{c_i} x_i$$

where, 'c_i' denotes the number of data points in i^{th} cluster.

5) Re-compute the distance between every data point and the new cluster centers obtained.

6) If no data point was reassigned then terminate, else repeat from step 3).

Web Service Operations Discovery Algorithm

The process of Main Web Services Discovery is dependent on the Index Library. Since the Index Library has completed the mining of the semantics prior to the beginning of the discovery process, only the relevant indexes set dependent on the request terms is used and ignore the extensive traversal of the Web services library. To accomplish a higher recall rate, this discovery consists of two kinds of results, including

"Single" and "Composite." Next, the process is introduced. This technique focuses on interface matching, and therefore, the input and output are taken to be the main discovery request content. At first, the discovery request is formalized to present this discovery process easily. The Webservice request is defined as

Request = <Input, output, φ>

Where

Input refers to a condition set to begin the discovery request function; it comprises of various keywords submitted by users and can be defined as *Input*=<*in*1,2⋯*inn*>.

Output indicates a result set, which refers to the completion of the discovery request; it also comprises of various keywords submitted by discovery users. It can be defined as *Out-put*=<*out*1,2…*outm*>.

The value *φ*refers to a step number to limit the composition search process by guaranteeing that the discovery system returns just those compositions that are found prior to *φ* steps.

In order to rank the discovery results easily, a structure is proposed, which defines the results of the discovery operation. A Discovery Result Operation (*DROp*) can be expressed as:

DROp= <Oid, weight>, where

Oid refers to the id for a Webservice operation that can be a combined form of the Webservice URL and an operation name, and weight refers to a value, which defines the confidence level of the interface matching process.

Results and Discussion

The experimental evaluation of the research work proposed is realized in the java simulation environment for the data set given with different topics. This comparison analysis is carried out between the newly introduced technique, known asAccurate and Faster Web Service Discovery Model (AFWSDM) and the available techniques, which are named Web Service Operations Discovery Algorithm (WSODA).The Performance analysis of the newly introduced technique is carried out in terms of Precision, Recall, Accuracy and F-measure parameters. Precision measure is computed depending on the formula

$$\text{Precision}=\frac{fp}{tp+fn} \tag{2}$$

Recall is computed using the formula

$$\text{Recall} =\frac{tp}{tp+fn} \tag{3}$$

Accuracy is computed applying the formula

$$\text{Accuracy} =\frac{tp+tn}{tp+tn+fp+fn} \tag{4}$$

F-Measure is computed applying the formula

$$F=2.\frac{\text{precision.recall}}{\text{prrecision}+\text{recall}} \tag{5}$$

Where tp – True Positive (Correct result), tn – True Negative (Correct absence of result), fp – False Positive (Unexpected Result), fn – False Negative (Missing result). The results of simulation are obtained for the evaluation of the newly introduced technique against different performance measures such as Precision, Recall and Accuracy.This technique achieves a much better rate of accuracy rate in comparison with the available system.

Accuracy is defined to be the correctness of the proposed technique in retrieving the accurate documents, which satisfactorily meet the user demands. The accuracy of the technique is computed for various topics that are present in the BBC news dataset like politics, entertainment, business, sports and technology. The comparison analysis of the newly introduced technique AFWSDM and available techniques such as WSODA is illustrated in Figure 3.

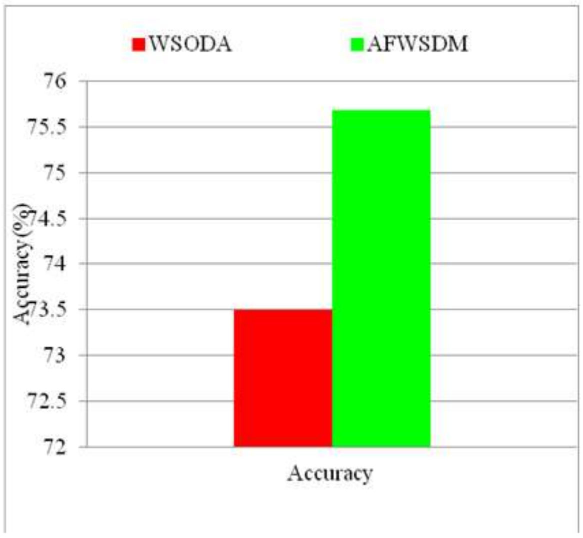

Fig. 3: Accuracy comparison

Figure 3 illustrates the results of accuracy comparison in which the newly introduced AFWSDM generates a 89.28% accuracy compared to the available techniques WSODA with a maximized percentage of 2-6% for accuracy parameter.

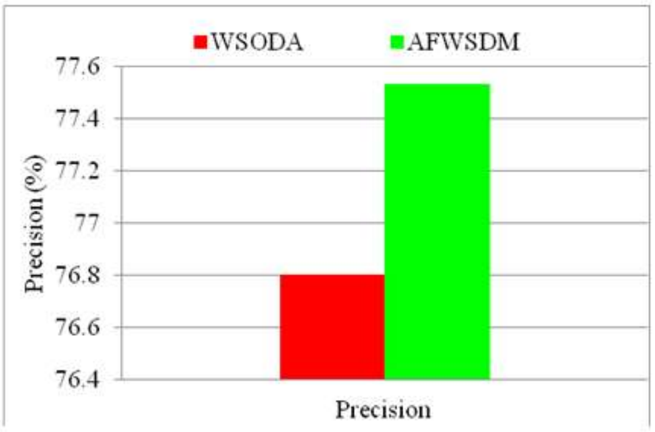

Fig. 4: Precision comparison

Figure 4 illustrates the results of the precision comparison in which the newly introduced AFWSDM generates 89.28% precision performs superior with a rise in percentage of 2-4% rather than the available research techniques.

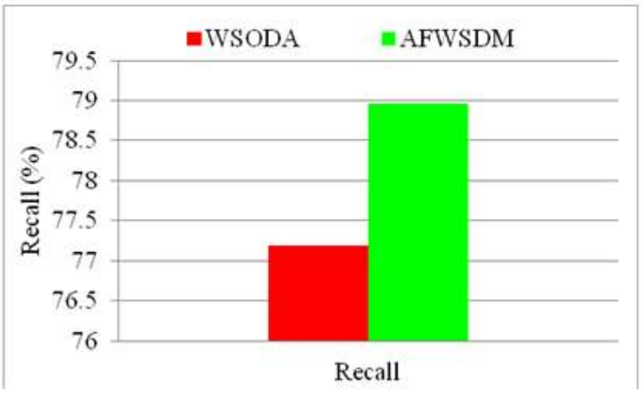

Fig. 5: Recall comparison

Figure 5 illustrates the results of recall comparison in which the newly introduced AFWSDM generates a 89.36% recall value that is quite higher with a rise in percentage of 2-4% obtained for recall parameter.

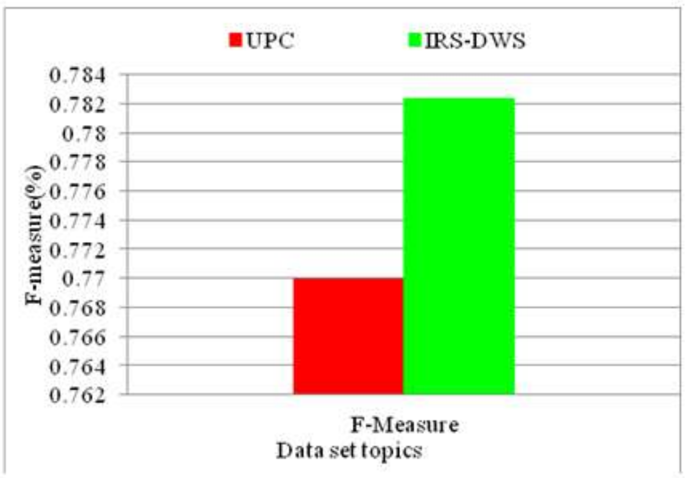

Fig. 6: F-measure comparison of similarity measures

Figure 5 illustrates the results of the F-Measure comparison in which the newly introduced AFWSDM exhibits 84% improvement in performance rather than the available research techniques in terms of increased F-Measure value.

III. CONCLUSION

In this technique, at first, a conceptual web services description model is proposed in which the type path is included for the interaction interface parameters along with the conventional text description. The extraction of all the content in the web service model can be done from the elementary description files employing direct XML tag extraction techniques. Optimality of a web service is dependent on its performance and it is measured via Quality of Service i.e. QoS that is inclusive of Response Time, Availability, Throughput, Success ability, Reliability, Best Practices, and Latency. The QoS factors are measured with the help of Firefly Algorithm (FA). The k-mean clustering algorithm is utilized for clustering interaction interface names and fragments under the monitoring of co-occurrence probability. At last, a web service operations Discovery algorithm (OpD) is proposed. The experimental evaluation reveals that the newly introduced approach exhibits superior performance compared to the available research works.

REFERENCES

[1]. Q.Z. Sheng, X. Qiao, A.V. Vasilakos, C. Szabo, S. Bourne, X. Xu. "Web services composition: A decade's overview", Information Sciences, vol. 280, pp. 218-238, 2014.

[2]. J.K.Y. Lau, J.P. Bruno. "U.S. Patent No. 9,936,333", Washington, DC: U.S. Patent and Trademark Office, 2018.

[3]. S. Deng, H. Wu, D. Hu, J.L. Zhao. "Service selection for composition with QoS correlations", IEEE Transactions on Services Computing, vol. 9(2), pp. 291-303, 2016.

[4]. H.Y. Paik, A.L. Lemos, M.C. Barukh, B. Benatallah, A. Natarajan. "Web Services–SOAP and WSDL", Web Service Implementation and Composition Techniques, pp. 25-66, 2017.

[5]. M. Suchithra, M. Ramakrishnan. "A survey on different web service discovery techniques", *Indian Journal of Science and Technology*, vol. **8**(15), 2015.

[6]. S. Kumari, S.K. Rath. "Performance comparison of soap and rest based web services for enterprise application integration", International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1656-1660, 2015.

[7]. S. Samir, A. Sarhan, A. Algergawy. "Context-based web service discovery framework with QoS considerations", 11th International Conference on Research Challenges in Information Science (RCIS), pp. 146-155, 2017.

[8]. C. Garcia, R. Abilio. "Systems Integration Using Web Services, REST and SOAP: A Practical Report", Revista de Sistemas de Informação da FSMA, vol. 1(19), pp. 34-41, 2017.

[9]. S. Agrawal, R.D. Gupta. "Development and comparison of open source based Web GIS Frameworks on WAMP and Apache Tomcat Web Servers", The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 40(4), 2014.

[10]. W.T. Tsai, R. Paul, Z. Cao, L. Yu, A. Saimi, B. Xiao. "Verification of Web Services Using an Enhanced UDDI Server", Proceedings of The Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 131-38, 2003.

[11]. J. Liu, N. Gu, Y. Zong, Z. Ding, S. Zhang, Q. Zhang. "Service Registration and Discovery in a Domain-Oriented UDDI Registry", Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 276-83, 2005.

[12]. W. Jian, W. Zhaohui. "Similarity-based Web Service Matchmaking", Proceedings of the IEEE International Conference on Services Computing, 2005.

[13]. G. Tretola, E. Zimeo. "Structure Matching for Enhancing UDDI Queries Results", IEEE International Conference on Service-Oriented Computing and Applications, 2007.

[14]. E. Ayorak, A. Bener. "Super Peer Web Service Discovery Architecture", IEEE Proceedings-International Conference on Data Engineering, pp. 287-94, 2007.

[15]. W. Libing He, Y. Wu, D. Jianqun. "A novel interoperable model of distributed UDDI", Proceedings of the IEEE International Conference on Networking, Architecture, and Storage-IEEE NAS, pp.153-54, 2008.

[16]. F. Nawaz, K. Qadir, H.F. Ahmad. "SEMREG-Pro: A Semantic based Registry for Proactive Web Service Discovery using Publish Subscribe Model", Fourth International Conference on Semantics, Knowledge and Grid, IEEE Xplore, pp. 301-08, 2008.

[17]. Q. Liang, J. Chung. "A Federated UDDI System for Concurrent Access to Service Data", IEEE International Conference on eBusiness Engineering, pp. 71-78, 2008.

[18]. T. Vandan, D. Nirmal, S. Inderjeet Garg, N. Garg, P. Soni. "An Improved Discovery Engine for Efficient and Intelligent discovery of Web Service with publication facility", 5th World Congress on Services, pp. 63-70, 2009.

[19]. C. Ma, M. Song, K. Xu, X. Zhang. "Web Service Discovery Research and Implementation Based on Semantic Search Engine", IEEE 2nd Symposium on Web Society, Beijing, pp. 672-77, 2010.