SWSRS: Semantic Web service Retrieval System based on Functional and QOS Evaluation

Muruganadam. M
Department of Information Technology
Syed Ammal Engineering College, Ramanathapuram , Tamilnadu, India.

Gopu. K
Department of Information Technology
Syed Ammal Engineering College, Ramanathapuram , Tamilnadu, India.

Shanmuga Priya. R
Department of Information Technology
Syed Ammal Engineering College, Ramanathapuram , Tamilnadu, India.

Abstract

In this paper, Web service discovery involves the service selection based on parameter-based service extraction and semantic similarity-based matching. In this approach, SWSRS is considered as the Web service Retrieval System. This approach assumes that the web service interfaces are defined in WSDL files. In this approach the similarity evaluation between the two interfaces are evaluated, higher the similarity less are the difference among their interfaces. This work based on the structure of WSDL and Semantic Web documents. The proposed approach incorporates with QOS requirements. However, earlier the QOS based non-functional requirement search has not been addressed. The lack of textual information makes keyword-based search models unable to filter irrelevant search results, and therefore, become very primitive means for effectively discovering web services. Therefore, the proposed approach i.e., SWSRS not only discussed the pertinent web service, it provides the services based on functional and QOS aspects.

Index Terms – Web service discovery, information search and retrieval, WSDL, OWL, QOS

I INTRODUCTION

Web service is a famous implementation of Service Oriented Architecture (SOA). SOA principles such as reusability, discoverability and platform independent that are forced the developer and organizations relying on the web services. Web service is a software component which can be accessed via internet using standard protocols. The protocols such as XML for message exchange, SOAP for communication, Web Service Description Language (WSDL) for describing the services and UDDI for service discovery. The advent of internet and web service which opens a new channel to the client and service providers to publish, locates, discover and access the web services. Due to the proliferation of web services, many service

A. Problem and Motivation

The user allies want to search a CurrencyConverter web service. There are more than 100 web services are available in the internet and ready to satisfy the request, identifying which service will be the best to meet the requirement has become challenge. Furthermore, the requester wants to search a web service whose response time < 200 milliseconds, availability > 99%, the traditional systems will not support the particular search. Therefore QOS based discovery has been introduced.

B. Limitation of Normal Search Engine

Search engine such as Google, Yahoo, AlltheWeb and Baidu are not well filtered for discovering services. Because their retrieval techniques specially developed to obtained pertinent data on web pages and not on web services. Search engines are mainly devoted to web sites. Its syntax and semantic of the search query are designed for web pages, it will not suitable for web services. The metadata content of the web services are very less and the search engines are not able to understand the semantics. The keyword based search are not able to capture the underlying semantics, they may miss some results and return a lot of irrelevant results. In order to investigate for a single operation usually needs several steps. Therefore it is difficult for the service providers to give a relevant result for the user search query. It will be very difficult for the consumer to find the web services which will be changed in future. If any changes will be done to the services then service consumer again have to repeat the discovery process to find the appropriate services.

C. Limitation of Keyword Search

The keyword based text document search is an intuitive approach. In IR community, document matching and classification is a long-standing topic and widely use

in most search engines. Due to the fact that the great success of search engines promotes the Web-related search very much, it might be a natural idea to employ the current search techniques for similar Web service discovery SWSRS is totally different from normal search engine as SWSRS purely based on web service results. The efficiency of search results produced by this work is more and search results are accurate and desired as per the user requirements. The need of aforementioned approach dealing with the Web service discovery processes. However, the QOS non-functional requirement search has not addressed. Therefore, the proposed system i.e., SWSRS not only discussed the pertinent web service which provides the services based on its performance.

D. Quality of Web Services (QWS)

QWS is the significant differentiator among the competing implementation. A client can search a web service according to non-functional requirements. A client can also search a web service based on its scalability such as response time, throughput etc. The proposed system consisting response time, throughput, reliability, availability are taken as major parameters for evaluating the performance of the web services. The qualities of services involve response time, throughput, availability and reliability. The various values of QWS are normalized and utility score value is calculated for each service. The QOS attributes such as response time, throughput, reliability and availability are taken as major parameters. The response time is the time taken to send a request and receive a response. Throughput is the Total number of invocations for a given period of time. Reliability denotes the ratio of the number of error messages to total messages. Availability defines the number of successful invocations/total invocations.

II RELATED WORKS

Web service substitution can be performed both at design time and at runtime and might occur, for instance, in case of Web service failures, or in case of Web service is unreachable. It is difficult for an enterprise user to dynamically invoke the most appropriate Web Service among a series of services with the same function. Tree representation and nested comparison enough information to establish a connection with a Web service, this specification lacks details on the real goal of the whole Web service and the constituting operations as well. J.Garofalakis et al. [3] proposed Web Service descriptions that may have been previously unknown, which meet the criteria expressed by the service requestor. The related works give the general overview of current web service discovery mechanisms and propose a categorization with respect to architectures, standards, QoS awareness, and data models. It was lacking behind to give the detail about the retrieval algorithms. E. Stroulia and Y. Wang et al. [4] this approach does not take into account the number of operations and parameters.

Xuanzhe Liu et al. [7] proposed architecture for web service discovery during which is based on agglomerative clustering algorithm. The search is based on input and output operation. This related work provides simpler and more efficient Web service discovery, which may align the requirements of the user-centric Web environment. It supports the users to search by typing "input" and "output" Operations. UDDI is mainly based on keyword search, which may bring several irrelevant results so that the consumers have to do the "view-select-request" process several times. Once the desired service is not available in UDDI, the consumer has to restart the discovery process.

The lack of textual information makes keyword-based search models unable to filter irrelevant search results, and therefore, become very primitive means for effectively discovering web services. Web services contain much more complex structure with very little text description. Therefore it makes the dependency on information basic retrieval techniques very infeasible. The existing systems text document search approaches are insufficient in the Web service environment.

The problem of automatically matching schemas investigates the clues of underlying semantics from the schema structure and suggests the matches based on them. In current WSDL the information about the retrieval of software components is not available. For constructing ontology as a semantic one for a large number of distributed Web services is really not easy. Ontology for manual annotation requires that the annotator have some skills in ontology engineering which is a quite high requirement for normal consumers.

Pierluigi Plebani and Barbara Pernici et al. [2] proposed an architecture called URBE (UDDI Registry By Example) for Web service discovery. In URBE, Quality and negotiation matchmaking represent, at this stage, the biggest open issues and about the functional matchmaking is matter of computation time. Web service registry managing should be deeply investigated as well and Semantic based approaches suffer of the need of services semantically described. Web services retrieval must be, first of all, *usable*! The problem in [1] is that all services will be discovered even the requester may not aware of irrelevant services.

III PROPOSED METHODS AND ALGORITHMS USED

A. Collection of WSDL Documents

OWL-S Service Retrieval Test Collection version 3.0 consists of 10073 indexed OWL-S services from the following 7 domains such as education, medical care, food, travel, communication, economy and weapon.OWLS-TC is available at semwebcentral.org :http://projects.semwebcentral.org/projects/owlstc/. The web services are retrieved from various sources such as www.xmethod .net, www.ceekda.com, and www.webserviceclient.com. The WSDL documents are automatically created once the web service has been created. SAWSDL descriptions are obtained adding

annotations to the WSDL previously derived from OWL-S files.

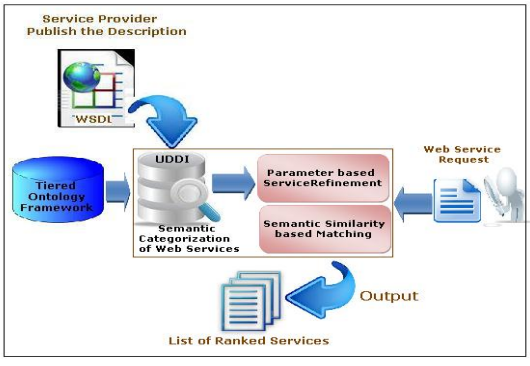

Fig. 3. Automatic Discovery of Semantic Web Services

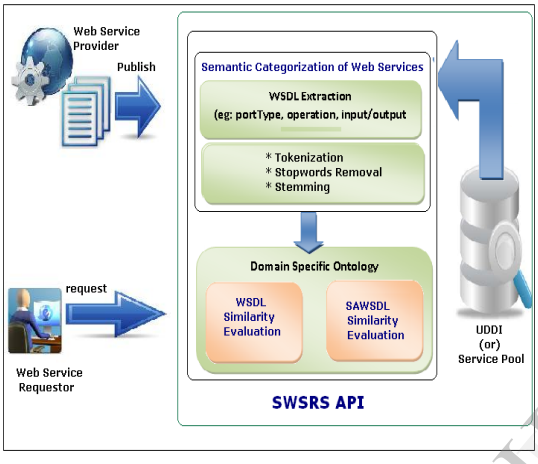

Fig. 4. Architecture of SWSRS

B. Refinement of WSDL Documents

The collected WSDL files are stored. The WSDL viewer can be used analyse the WSDL structure. The WSDL elements can be viewed in detail and clear way. The WSDL viewer can also be used to extract the WSDL elements such as portType, operation, input/output parameters. The extracted WSDL elements are stored in the service pool of SWSRS. SWSRS focus on the WSDL elements such as portType, operations and input/output parameters. So these elements are stored in the table of service pool. The tokenization is the process of splitting up of character of strings into list of words or terms. The terms resulting from the tokenization are stemmed. It involves removal of prefix and suffix using *stemming algorithm.*

C. Semantic WSDL Structure Matching

The semantic WSDL structure matching algorithm is an extension to it. It also tries to find an optimal mapping between source and target service components based both on the similarity of their syntactic structures and also the semantic similarity between the identifiers of data types, operations and services to assess service similarities. The intuition behind it is that the chosen names of the types, operations, and services usually reflect the semantics of the underlying capabilities of the service. The identifier-matching process is similar to that of the original WSDL structure matching. It starts by comparing the names of the data types (identifiers) involved in the two WSDL specifications. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target data-types. The next step in the process is the matching of the service operations. The result of this step is a matrix assessing the matching scores of all pair-wise combinations of source and target operations. The degree to which two operations are similar is decided on the semantic distance between operations' names and how similar their parameter lists are, in terms of the identifiers they contain. Finally, the overall score for how well the two services match is computed by matching the services' names and by identifying the pair-wise correspondence of their operations that maximizes the sum total of the matching scores of the individual pairs.

If two words are identical or synonymous (regardless of words' senses), they are assigned a maximum score of 10 and 8 respectively. Otherwise, if two words are in a hierarchical semantic relation, i.e. they are hypernyms, hyponyms or siblings to each other we count the number of semantic links between these words along their shortest path in WordNet hierarchy. The identifier-similarity score between two such terms is calculated by dividing 6 by the number of links found between them. Thus, the term-similarity score is a function of the terms' semantic distance in the WordNet hierarchy: terms that are farther away from each other have smaller similarity scores than terms that are located closer to each other in WordNet. Similar to the WordNet-based information-retrieval step, word senses are not disambiguated.

D. Semantic Categorization of Web Services

The Service Categorization refers to an entry in some ontology or taxonomy of services. The value of the property is an instance of the class ServiceCategory. ServiceCategory describes categories of services on the bases of some classification that may be outside OWL-S and possibly outside OWL. Service Parameter is an expandable list of properties that may accompany a profile description. The value of the property is an instance of the class ServiceParameter. OWL-S is ontology, within the OWL-based framework of the Semantic Web, for describing Web services. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. OWL-S will help make the Semantic Web a place where people can not only find information but also get things done.

IV SWSRS SIMILARITY EVALUATION

A. Name Similarity evaluation

The related words are considered to be the similar words. The Ontology can be built by the Domain experts and also analyzing the terms in the Web Services published in the registry. Domain-specific ontology offers more accuracy in the relationship of the terms. It includes terms related to a given application domain.The ***goRankOntology*** **(***http*://www.gorank.com/seotools/ ontology /**)** lookup tool checks the top 1000 Google results for the keyword by running related word query.

```
double matchDocumentTerms (term1, term2) {
        maxScore = 10;
        if (term1 is identical to term2)
            score = maxScore;
        else if (term1 and term2 are synonymous)
            score = 8;
        else if (term1 and term2 have hierarchical relations)
            score = 6 / number of hierarchical links;
        else score =0;
        return score;
}
```

Domain experts will assign the score value is based on the relationship between the terms

Fig. 5. Terms matching based on ontology

The domain-specific ontology includes terms related to a given application domain.

B. Data Type Similarity Function

In a WSDL description, data types are expressed by XML Schema Definition (XSD) specifications. Simple data types (e.g., xsd:string, xsd:decimal, and xsd:dateTime) as well as derived data types(e.g., xsd:integer, xsd:short, and xsd:byte) are included. The function dataTypeSim: {dtq; dtp}→ [0…1] calculates the similarity between two data types.

C. Web Service Similarity Function fSim

PortType Name Maximum Similarity Evaluation:

It involves the **portType** name similarity between two Web Services which is extracted from WSDL.

$$\max Sim(WS_1PT, WS_2PT) = \frac{(Sum\ of\ \max\ SimVal\ between\ tokens)}{(number\ of\ tokens\ in\ WS_1PT}$$

Where,

$WS1_1\ PT(or)\alpha_q PT$ – First Web service portType (eg. CurrencyWS).

$WS_2PT(or)\alpha_p PT$ – Second Web service portType (eg. ExchangeRate).

As per the Fig. 7. Mentioned above,

$$\max Sim(WS_1PT, WS_2) = 0.8$$

Operation Name Maximum Similarity Evaluation:

It involves the **operation** name similarity between two Web Services which is extracted from WSDL.

$$\max Sim(WS_1PTOp_i, WS_2PTOp_j) = \frac{(sum\ of\ \max\ SimVal\ between\ the\ optokens)}{(no.\ of\ tokens\ in\ WS_1OP_i)}$$

Where,

$WS1_1\ PTOP_i(or)\alpha_q OP$ – List of operation in first Web service (eg. CurrencyWS).

$WS_2PTOP(or)\alpha_p OP$ – List of operation in second Web service (eg. ExchangeRate).

As per the Fig. 7. mentioned above,

$$\max Sim(WS_1PTOP_i, WS_2PTOP_j) = 1.0$$

Input name and type Maximum Similarity Evaluation:

It involves the **input** name similarity between two Web Services which is extracted from WSDL.

$$\max Sim(WS_1PTOP_iInp_x, WS_2PTOP_jInp_y) = \frac{(sumof\ \max\ SimValbetweenthetokens)}{Tot\ WS_1PTOP_iInp_x}$$

$Tot\ WS_1PTOP_iInp_x$ = number of tokens in $WS_1PTOP_iInp_x$

Where,

$WS_1PTOP_iInp_x$ – List of input in first Web service (eg. CurrencyWS).

$WS_2PTOP_jInp_y$ - List of input in second Web service (eg. ExchangeRate).

As per the Fig. 7. mentioned above,

$$\max Sim(WS_1PTOP_iInp_x, WS_2PTOP_jInp_y) = 1.0$$

Output type Maximum Similarity Evaluation:

It involves the **output** type similarity between two Web Services which is extracted from WSDL. Similarly the output datatype can be evaluated as mentioned above.

$$\max Sim(WS_1PTOP_iOut_x, WS_2PTOP_jOut_y) = 1.0$$

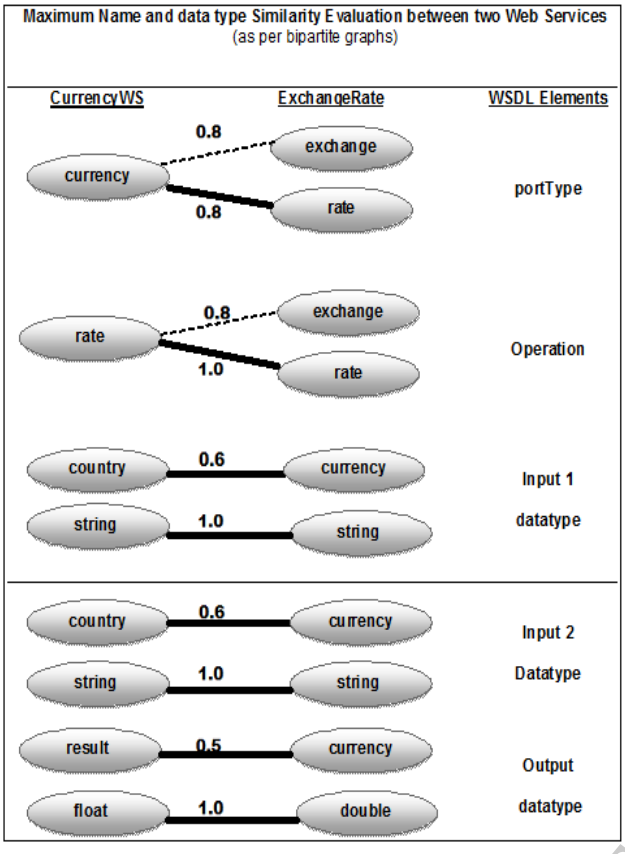

Fig. 7. Bipartite graph implementation for the WSDL Elements

D. *Similarity Evaluation Algorithm*

This work follows the algorithm given below. With $\sum = \{\alpha_p\}$ which represent the Web service registry, in which the interfaces of all the available services are published. Thus, α_p identifies a generic published Web service interface. This work returns $\sum = \{\alpha_q\}$ as the set of Web services interfaces included in the registry which are similar to the query. Thus, according to a query-by-example approach, the user defines the characteristics of the desired Web service in the same way as the published ones.

- *fSim = similarity degree between two Web services whose value ranges from [0…1].*
- *T*he fSim function relies on two main functions: *Name similarity function and Data type similarity function*

```
Input α_q;    // receive the query
Input thSim;  // set threshold
Let ∑α_q = θ;  // initial the result
for(p = 0; p < |∑ |; p++)
    sim = fSim(α_q, α_p)  // similarity evaluation
    if(sim >= thSim) then
      add(∑α_q, α_p);
    end if
end for
```

Fig. 9. Similarity Algorithm to evaluate ***fSim***

After the process of tokenization and stemming, the tokens are extracted from portType, operation and input/output parameters. Then the tokens are compared to find the name similarity by using bipartite algorithm. After applying the maximum value on the fSim equation, the fSim will be evaluated. The fSim values lies between {0….1}. If the fSim value is 1 then the query WSDL can be replaced with the services with fSim value as 1.0. Based on the threshold value the services can be predicted whether it is similar to the query service.

V RESULTS AND PERFORMANCE ANALYSIS

A. *Experimental Results*

The web services are retrieved from various sources such as www.xmethod.net, www.ceekda.com, and www.webserviceclient.com. More than 2500 real world web services are taken for experiments. More than 25000 operations are extracted from the WSDL documents. More than 100000 datatypes of input/output parameters are compared. For example more than 15 currencyconverter services are taken for evaluating the similarity among the cluster of currency converter services.

TABLE 1 : List of Web services with WPTNameSim, WOpNameSim & fSim

Service Name	Service URL	WPTNameSim	WOpNameSim	fSim
CurrencyConvertor (query WSDL)	http://www.webservicex.net/CurrencyConvertor.asmx?wsdl	1.0	1.0	1.0
CurrencyRates	http://ws.strikeiron.com/HouseofDev/currencyrates?WSDL	0.85	0.87	0.8
CurrencyRates	http://ws.strikeiron.com/HouseofDev/currencyrates151?WSDL	0.85	0.87	0.8
CurrencyServerWeb Service	http://www.currencyserver.de/webservice/currencyserverwebservice.asmx?WSDL	0.85	0.83	0.8
ExchangeRates	http://webservices.lb.lt/ExchangeRates/ExchangeRates.asmx?wsdl	0.70	0.83	0.7
ExchangeRates	http://www.gama-system.com/WebService/exchangerates.asmx?wsdl	0.70	0.90	0.8
ForeignExchangeRate	http://ws.strikeiron.com/StrikeIron/ForeignExchangeRate2/ForeignExchangeRates	0.75	0.83	0.7
ExchangeRateService	http://www.example.com/exchangerate/services/getrate	0.70	0.82	0.7
DOTSCurrencyExchange	http://ws2.serviceobjects.net/ce/CurrencyExchange.asmx	0.85	0.95	0.9

$$\begin{cases} (q)_{\max} = \begin{cases} \max_{s\in S} q_i(s) & if Q_i \in QoS^{Positive} \\ (q_i)_r & if Q_i \in QoS^{negative} \end{cases} \\ (q)_{\min} = \begin{cases} \min_{s\in S} q_i(s) & if Q_i \in QoS^{Positive} \\ (q_i)_r & if Q_i \in QoS^{negative} \end{cases} \end{cases}$$

Each QoS criteria will be transformed into real value between 0. $Q_{\min}(k)$ and $Q_{\max}(k)$ represent maximum and minimum value of k th criteria of a web service.

If the QoS value is positive the utility value also increasing direction (reliability, availability), if the QoS value is negative, the utility value is in a increased direction (response time, throughput).

$$Utility_i(q_i) = \begin{cases} \dfrac{q_i - (q_i)_{\min}}{(q_i)_{\max} - (q_i)_{\min}} & if Q_i \in QoS^{Positive} \\ \dfrac{(q_i)_{\max} - q_i}{(q_i)_{\max} - (q_i)_{\min}} & if Q_i \in QoS^{Negative} \\ 1 & if\ (q_i)_{\max} - (q_i)_{\min} \end{cases}$$

Now the Quality matrix Q is transformed into:

$$\begin{pmatrix} q_1',1 & q_1',2 & \ldots\ldots & q_1't \\ q_2',1 & q_2',2 & \ldots\ldots & q_2',t \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ . & . & \ldots\ldots & . \\ q_s',1 & q_2',2 & \ldots\ldots & q_s',t \end{pmatrix}$$

TABLE 2 : QoS Attributes for web service selection

Parameter	Definition	Unit of measurement
Response time	Time taken to send a request and receive a response	Millisecond
Throughput	Total number of invocations for a given period of time	Invocations per second
Reliability	Ratio of the number of error messages to total messages	Percent
Availability	Number of successful invocations/total invocations	Percent

The QOS values of the following services are listed in the following table.

TABLE 3 : QOS Utility score value

Service Name	QOS Utility Score Value
CurrencyConvertor	0.614
CurrencyRates	0.604
CurrencyRates	0.7
CurrencyServerWebService	0.54
ExchangeRates	0.507
ExchangeRates	0.507
ForeignExchangeRate	0.39
ExchangeRateService	0.58
DOTSCurrencyExchange	0.595

C. Performance Analysis

The evaluation of this work consists of two main steps. First, we consider a small set of Web services and we tune the algorithm, i.e., we identify the best values for the parameters which the algorithm depends on. Second,

over a larger set of Web services the performance of the algorithm and its semantic extension are evaluated and compared with related approaches. Precision and recall are measures for the entire result set without considering the ranking order.

$$\Pr ecision\ (P) = \frac{\left| \{\alpha_i \in \sum \alpha_q \mid \alpha_i \in \Re_{\alpha q}\ \} \right|}{\left| \sum \alpha_q \right|}$$

$$\text{(or)}\ \Pr ecision(P) = \frac{A}{A+B}$$

$$\text{Re}\, call\ (R) = \frac{\left| \{\alpha_i \in \sum \alpha_q \mid \alpha_i \in \Re_{\alpha q}\ \} \right|}{\left| \Re_{\alpha q} \right|}$$

$$\text{(or)}\ \text{Re}\, call(R) = \frac{A}{A+C}$$

Where α_q is the query, $\sum \alpha_q$ the returned services after submitting the query, and $\Re_{\alpha q}$ is the relevant services for the given query. Precision and recall are measures for the entire result set without considering the ranking order. In Recall, A stands for the number of returned relevant operations, B stands for the number of lost relevant operations, and A + B stands for the total number of relevant operations. In precision, where A stands for the number of returned relevant operations, C stands for the number of irrelevant operations.

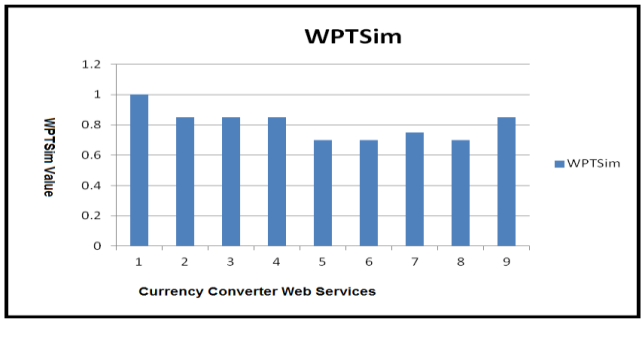

Fig. 10. PortType similarity (WPTSim) graph.

The Fig. 11. mention the operation name similarity. The operations in various services are extracted and compare with the other operations among the cluster of currency converter web services. The x-axis represents the list of currencyconverter web services and y-axis represents the operation name similarity.

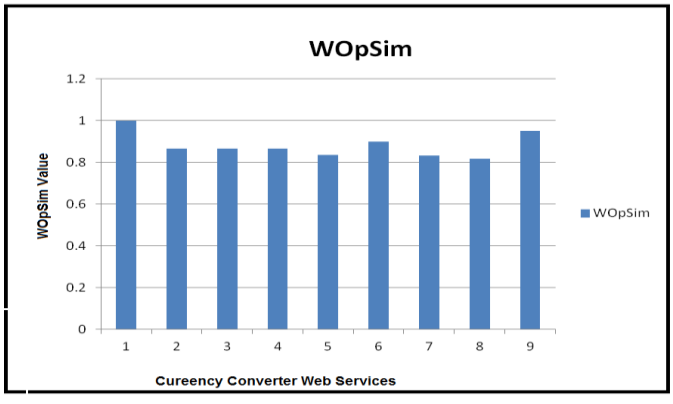

Fig. 11. Operation Name Similarity (WOpNameSim) graph.

Fig. 12. Comparison graph for Keyword and Operation search

The above Fig. 12. represents the comparison of the searched results of keyword and operation search. As per the overall search results of operation search precision is high as compared to keyword search. The search results will be ordered or listed by the QWS values.

VI CONCLUSION

In this work, SWSRS is represented as an approach for evaluating the similarity between Web service interfaces for substitutability purposes. The Web service requestor, after querying for the desired Web service a list of similar Web services will be listed out based on functional and QoS Evaluation. The evaluation of the similarity between Web services considers the structure of a WSDL description. The semantic analysis takes into account the names adopted to describe the elements composing a Web service (operations and parameters), whereas the structure analysis takes into account the number of operations as well as the number and data types of the parameters. This approach involves the service selection based on parameter-based service extraction and semantic similarity-based matching. The future work will focus on improving performance in terms of execution time. About the functional matchmaking is matter of computation time as Web service registry managing should be deeply investigated as well.

REFERENCES

[1] Aabhas V.Paliwal, Basit Shafiq, JaideepVaidy, Nabil Adam, "Semantics-Based Automated Service Discovery", IEEE VOL. 5, NO.2, APRIL-JUNE 2012.

[2] Pierluigi Plebani and Barbara Pernici, "URBE:Web service Retrieval Based on Similarity Evaluation", Vol. 21, No. 11, November 2009

[3] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Contemporary Web Service Discovery Mechanisms," J. Web Eng.,vol. 5, no. 3, pp. 265-290, 2006.

[4] E. Stroulia and Y. Wang, "Structural and Semantic Matching for Assessing Web Service Similarity," Int'l J. Cooperative Information Systems, vol. 14, no. 4, pp. 407-438, 2005.

[5] T. Syeda-Mahmood, G. Shah, R. Akkiraju, A.Ivan, and R. Goodwin, "Searching Service Repositories by Combining Semantic and Ontological Matching," Proc. IEEE Int'l Conf.Web Services (ICWS '05), pp. 13-20, 2005.

[6] Yiqiao Wang and Eleni Stroulia , "Semantic Structure Matching for Assessing Web-Service Similarity ", Computer Science Department,University of Alberta, Edmonton, AB,T6G2E8,Canada{yiqiao,stroulia}@cs.ualbe-rta .ca

[7] Xuanzhe Liu, Gang Huang, "Discovering Homogeneous Web Services Community in the User-Centric Web Environment", IEEE VOL. 2, No. 2, APRIL-JUNE 2009.

[8] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Pen, "Swoogle: A Semantic Web Search and Metadata Engine".

[9] N. Seco, T. Veale, and J. Hayes, "An Intrinsic Information Content Metric for Semantic Similarity in Wordnet," Proc. European Conf. Artif - icial Intelligence (ECAI '04), pp.1089-1090, Aug. 2004.

[10] Debajyoti Mukhopadhyay,Aritra Banik, Sreemoyee Mukherjee, Jhilik Bhattachary, "A Domain Specific Ontology Based Semantic Web Search Engine "

[11] Tim Bray, *What is RDF? http://www.xml.com/lpt/a/2001/01/24/rdf.html*

[12] W3C, *RDF Primer, W3C Working Draft 23 January 2003, http://www.w3.org/TR/2003/ WD-rdf-primer-20030123/*

[13] Sean B Palmer,*The Semantic Web:An Introduction,2001, http://infomesh.net/2001/ swintro/*

[14] OWL-S Service Retrieval Test Collection http://projects.semwebcentral.org/projects/owls2 wsdl/

[15] http://code.google.com/p/owls2wsdl/

[16] http://www.gorank.com/seotools/ontology

[17] S. McIlraith, T. Son, and H. Zeng, "Semantic Web Services," IEEE Intelligent Systems, vol. 16, no. 2, pp. 46-53, Mar. 2001.

[18] S. McIlraith and D. Martin, "Bringing Semantics to Web Services,"IEEE Intelligent Systems, vol. 18, no. 1, pp. 90-93, Jan. 2003.