Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Discovering the Best Web Service: A Neural Network-based Solution

Eyhab Al-Masri and Qusay H. Mahmoud
Department of Computing and Information Science
University of Guelph, Guelph, ON, N1G 2W1 Canada
College of Information Technology
Zayed University, United Arab Emirates

***Abstract*— Differentiating between Web services that share similar functionalities is becoming a major challenge into the discovery of Web services. In this paper we propose a framework for enabling the efficient discovery of Web services using Artificial Neural Networks (ANN) best known for their generalization capabilities. The core of this framework is applying a novel neural network model to Web services to determine suitable Web services based on the notion of the Quality of Web Service (QWS). The main concept of QWS is to assess a Web service's behaviour and ability to deliver the requested functionality. Through the aggregation of QWS for Web services, the neural network is capable of identifying those services that belong to a variety of class objects. The overall performance of the proposed method shows a 95% success rate for discovering Web services of interest. To test the robustness and effectiveness of the neural network algorithm, some of the QWS features were excluded from the training set and results showed a significant impact in the overall performance of the system. Hence, discovering Web services through a wide selection of quality attributes can considerably be influenced with the selection of QWS features used to provide an overall assessment of Web services.**

Keywords*— *UDDI, Service Registries, Web Services, Quality of Service, Neural Networks.

I. INTRODUCTION

Development and deployment of Web services using a service-oriented architecture (SOA) is one of the best approaches for businesses to adapt their business processes to meet new requirements. Web services are becoming the ideal platform for SOA particularly due to their ability in achieving interoperability by dynamically consuming data regardless of language platform, operating system, or hardware type. However, one of the main challenges in achieving this dynamic environment is assessing the behaviour of Web services to meet client requirements. Although clients' perception of the behaviour of Web services may vary, providing an overall assessment rating covering a wide variety of Quality of Web Service (QWS) features is a fundamental step toward the differentiation of Web services that share similar functionalities.

In order to assess the behaviour of a Web service, it is essential to continuously monitor and collect various QWS features. QWS should therefore cover a large number of qualitative and quantitative properties (non-functional properties) that provide an overall measure of the Web service performance in delivering the required functionality and in accordance with existing standards. Using QWS for Web services, query optimization can be achieved when looking for relevant Web services. However, the key challenge is how to define appropriate properties to characterize QWS and therefore provide service-based queries.

To address the above mentioned issues, this paper introduces a comprehensive model for QWS and defines QWS properties for individual Web services. The rest of this paper is organized as follows: Section Two discusses some of the related work. Section Three discusses the classification of Web services. Section Four describes the use of Artificial Neural Networks (ANNs) for the implementation of the current system. Section Five describes how QWS data is collected Section Six discusses results and performance of the current system. Finally conclusion and future work are discussed in Section Seven.

II. RELATED WORK

Several Web services may share similar functionalities, but with different non-functional attributes. When discovering Web services, it is essential to take into consideration the functional and non-functional properties for finding relevant Web services of interest and therefore, rendering an effective selection process.

Although there have been numerous efforts to provide QoS support for Web services [10], very little research has been conducted to explain the collection of service qualities can be achieved in a transparent and fair manner. In addition, there is a high probability that service providers may perform changes or updates to a Web service and as a result QoS values may also be affected. Very little research has been conducted on how to guarantee that the QoS information collected at runtime contain the most recent QoS information. The proposed solution provides an active monitoring tool that continuously collects the most recent and up-to-date QoS values and ensures that QoS computations are performed in an open and dynamic manner.

In [14], an approach for certifying QoS information by a Web service QoS Certifier is proposed. In this approach, the Web service provider has to provide QoS information at the time of registration. Registration will be placed on hold until a Web service Certifier issues a certification ID. Once the certifier verifies the claims of the QoS information supplied by the service provider, the UDDI then registers the Web service. However, this approach does not provide a reliable mechanism and an adequate solution for solving the support of QoS properties for Web services. This solution proposes the extension of the current UDDI design which might not be feasible, concentrates on verifying QoS properties at the time of registration, does not provide any guarantees of having up-to-date QoS information (i.e. in case of Web service updates or changes), and requires service providers to go through additional steps during the publishing process which enforces service providers to think of ways on how to measure the QoS of their Web services. In addition, the current solution does not differentiate between QoS properties directly supplied by service providers [8] (i.e. classification, and cost) or how the QoS certifier handles such parameters when issuing a certification.

Other approaches focused on the use of QoS computation and policing in improving the selection of Web services [11,20], developed a middleware for enhancing Web Service composition for monitoring metrics of QoS of Web services [15], using an agents based on distributed reputation metric models [16], and using conceptual model based on Web Service reputation [13]. Many of these approaches do not provide guarantees as to the accuracy of QoS values over time or having up-to-date QoS information. In addition, preventing false ratings using reputation metric models is not present and therefore, false information may be collected and as a result may significantly impact the overall ratings of service providers.

There are other approaches focused on the use of text document matching [9,18,19,12] and mainly depend on the analyzing the frequency of terms. Other approaches studied the use of supervised classification and unsupervised clustering of Web services [7], artificial neural networks [1], or using unsupervised matching at the operation level [5]. However, many of these approaches do not take advantage of applying neural networks to identify Web services based on their classification when discovering relevant Web services. Using neural networks, we can determine Web services that are considered to be first class objects and find any patterns or relationships with other similar Web services of interest (i.e. service-links) based on their behaviour and measured QWS parameters.

III. Classification of Web Services

It is very common that business entities may offer the same Web service in many forms and variations. A service provider may offer the same service but with different feature set and pricing. For instance, Amazon.com provides the Amazon Web Service (AWS) which enables developers to partially access its technology platform. However, the service provider, in this case Amazon.com, may provide different qualities of the AWS service at varying QWS-related features such that AWS Enterprise provides a class of services with a maximum throughput of 1,000 invocations/second (i.e. type Platinum) while AWS Basic provides a class of services with a maximum throughput of 200 invocations/second (i.e. type Bronze) as shown on Table I.

TABLE I Sample Web Service Classes for AWS

AWS Service Offering	Platinum	Gold	Silver	Bronze
Enterprise	✓			
Professional		✓		
Ultra			✓	
Basic				✓

The categories representing various service offering differ in service qualities or properties such that the Platinum provides higher levels of quality (i.e. less response time, cheaper flexible licensing information, among others) while those under the Bronze class offer the same service properties but at a lower quality (i.e. slightly higher response times, strict licensing information, among others) as shown in Table II.

TABLE II Sample Web Service Classes for AWS

QW Property/Category	Platinum	Gold	Silver
Response Time (ms)	3.45	6.54	8.90
Throughput (max. req.)	1000	400	200
Availability (%/month)	100%	95%	90%
Reliability	100%	90%	85%

The measured QWS values are used by the neural network to classify those services that have higher ratings in terms of QWS. Treating all QWS parameters with equal weights, we use the confidence intervals of the standard deviation based on a relevancy function called WsRF [4] to determine the normal distribution and classify each Web service to a corresponding service group. WsRF is used to measure the relevancy ranking of a particular Web service. QWS parameters help determine how Web services can be treated or categorized. The classification scheme is used by the Artificial Neural Network (ANN) as a classifier or label for training the network.

IV. Artificial Neural Networks

Finding the correct feature set for the physical process in this study is a common problem when it is associated with classification and/or selection. There is no unique feature extraction technique that is capable of satisfying the requirements of all types of data. Therefore, a solution using Artificial Neural Networks (ANNs) is used in order to increase the success rate of finding the best available service over ordinary solutions. There are many reasons for using artificial neural networks for this system as a classifier: (1) artificial neural network provides a simple representation for physical implementation; (2) weights that represent the solution are generated through iterative training; and (3) ANN produces correct results for inputs that are not present in the training set. Neural networks can be used for classification of Web services. In fact, Web services can be represented as a graph in which nodes represent QWS parameters and arcs denote relationships or patterns. In this paper, a backpropagation neural network (BPNN) algorithm is implemented.

Classifying Web services into class types using artificial neural networks listed in Table I is not a simple classification problem. In order to solve the service classification problem, one hidden layer is used in the feedforward neural network. In the proposed system, all neurons use sigmoid activation function. Random values, which serve as weights, are generated for all connections from input to hidden (referenced by v_{hi}) and from hidden to output layers (reference by w_{ij}). In addition, biases are assigned random values at the hidden nodes (θ_i) and the output node (τ_i). The activation functions at the hidden layer (referenced by b_i) are calculated using the following equation:

$$b_i = f(\sum_{i=1}^{n} a_h \times v_{hi} + \theta_i) \qquad (1)$$

where f(x) is the logistic sigmoid threshold function f(x) = 1/(1+e-x). The activation values at the output layer (referenced by c_j) are calculated as follows:

$$c_j = f(\sum_{i=1}^{n} b_i \times w_{ij} + \tau_j) \qquad (2)$$

where f(x) is the logistic sigmoid threshold function f(x) = 1/(1+e-x).

The backpropagation neural network has been trained with moderate values for the learning rate (α) and momentum (μ). The weights are recalculated every time a training vector is presented to the network. The exit strategy or the termination condition for the network is based on the sum square error until it reaches a certain threshold assigned prior to running the network. The network should be able to classify a given Web service into any of the class types discussed in Table I. The neural network can be used for classification of Web services in which it can determine patterns, trends or relationships with other Web services in the same domain of interest based on the QWS values or their overall assessment in delivering the required functionality. In this manner, the neural network is capable of identifying first class objects and be able to also recommend services of interest in a ranked manner based on their classification and assessment.

V. Monitoring QWS for Web Services

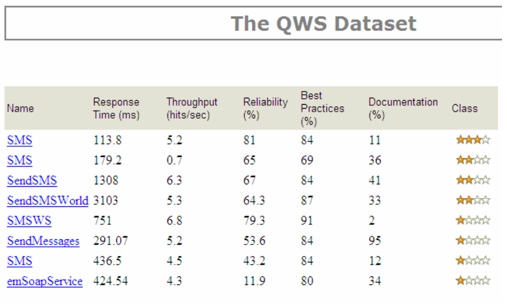
The key in assessing a Web service is to continuously monitor its behaviour in delivering the required functionality or over a given period of time. Prior to monitoring services, it is crucial to be able to collect as many Web services that are available over the Web from accessible resources including service registries, search engines' databases, service portals, sharing platforms, among others. To achieve this task, we used our Web Service Crawler Engine (WSCE) [3,21] that continuously collects Web service information available from any accessible resource. Crawling Web services is a critical step toward obtaining meaningful QWS values.

Ideally, to assess the quality of a particular Web service, it is essential that it contains at least one accessible operation which means that a service endpoint has to be valid. However, some Web services may contain one or more operations but the service endpoint is not accessible and hence could not be monitored or considered serviceable. An automated engine such as WSCE can determine the serviceability of Web services which can then be used for monitoring. Once a Web service passes through WSCE serviceability tests, it is stored into the Web Service Storage (WSS) which activates the Quality Web Service Manager (QWSMan) to begin monitoring newly added services. WSCE, WSS, and QWSMan are part of our larger Web Service Broker (WSB) framework [2].

Although receiving a successful response when invoking a Web service indicates its serviceability, there is no mechanism to determine if the response is only dependent on the service provider generating it. For example, a response of a particular Web service may depend on two or more other subcomponents or external resources, and therefore it would be beneficial to determine the processing time. The processing time is the time it takes for a Web service provider to process an incoming request and sending a response. However, for the purpose of this paper, we assume that Web services exclusively reside on the service providers host servers and do not depend on any external subcomponents.

VI. Data, Results, and Discussion

In order to suit the purpose of the backpropagation artificial neural network, we used the publicly available Quality of Web Service (QWS) dataset [17]. QWS dataset is composed of a set of QWS measurements for 2,507 real Web service implementations that exist on the Web today. The Web services were collected using the Web Service Crawler Engine (WSCE) [3] and quality measurements were conducted using the Quality of Web Service Manager (QWSMan) [4]. The majority of the Web services collected in this QWS dataset were obtained from public sources including Universal Description, Discovery, and Integration (UDDI) registries, service portals, and Web search engines. Each Web service measurement consists of nine QWS parameters measured using commercial Web service benchmark tools. A sample result (available on the Web in [17] – Demo section) for a query with the keyword "sms" to the WSB framework yield the results shown in Figure 1. Table III presents QWS parameters, description and measuring units which were used for this method



The QWS Dataset

Name	Response Time (ms)	Throughput (hits/sec)	Reliability (%)	Best Practices (%)	Documentation (%)	Class
SMS	113.8	5.2	81	84	11	★★★☆
SMS	179.2	0.7	65	69	36	★★☆☆
SendSMS	1308	6.3	67	84	41	★★☆☆
SendSMSWorld	3103	5.3	64.3	87	33	★★☆☆
SMSWS	751	6.8	79.3	91	2	★☆☆☆
SendMessages	291.07	5.2	53.6	84	95	★☆☆☆
SMS	436.5	4.5	43.2	84	12	★☆☆☆
emSoapService	424.54	4.3	11.9	80	34	★☆☆☆

Figure 1. Sample screenshot of our system using QWS Dataset

TABLE III. QWS PARAMETERS

ID	Parameter	Description
1	Response Time	Time to send a request and receive a response
2	Availability	Successful invocations/total invocations
3	Throughput	Total Number of invocations /period of time
4	Successability	Ratio: # response messages / # request messages
5	Reliability	Ratio: number of error messages/total messages
6	Compliance	Extent a WSDL follows a specification
7	Best Practices	Extent of following WS-I Basic Profile
8	Latency	Time to process a given request
9	Documentation	Measure of documentation in WSDL
10	WsRF	Rank for Web Service Quality
11	Classification	Levels representing service offering qualities
12	Service Name	Name of the Web service
13	WSDL Address	Location of the (WSDL) file on the Web

Figure 1 shows a sample output for a query performed to the WSB framework in which services that match a given query are considered in the output result. The WsRF is computed for all matching Web services. The ANN is then used to classify each Web service into a specific class (i.e. Platinum, Gold, Silver, Bronze). A service that is categorized as Platinum is shown with 4 stars in Figure 1 while one that is classified as Bronze is represented with 1 star. For example, “emSOAPService” is classified as type Bronze (from Table I) because it has long response time, and low reliability rate.

A sample cross section of the QWS dataset is shown in Table IV for six collected Web services. The IDs in Table IV header represent QWS identifiers from Table III. QWS parameters shown in Table IV are used as inputs to the backpropagation neural network. In order to select the best Web service or one that is considered to be first class object, there exists a list of Web services in which the neural network must be able to select or propose.

The samples in the QWS dataset are classified into one of the given categories presented in Table II (i.e. when a query is performed). A client, for example, makes a query to the WSB framework, and our ANN-approach will classify Web services based on their overall WsRF ranking, as shown in Figure 1. The WsRF rating (i.e. 10th parameter from Table III) is calculated based QWS measurements conducted by QWSMan. More information on our ranking using QWS parameters can be found in [4].

A program was written to allow clients to define weights that represent level of importance to any of the QWS properties. Each client preference is represented by a percentage and then converted to a scale of 0 to 1 to be later used as weights. For example, if a client searches for a particular Web service and assigns more weight to QWS parameter 3 (throughput) with 100% priority, then the program would choose the Web service with the highest throughput from services within the QWS dataset.

In order to determine the performance of the neural network, an acceptable error value is used to determine if the network is able to converge into a possible solution or not. Due to the fact that the neural network uses a logistic function in order to detect the error for each iteration, the input must be within the range of 0 and 1 (since the sigmoid will make sure that the nodes at the output node will never be 1 or 0 but either 1-e or e). For all of the samples in the QWS dataset, the inputs must be normalized. In order to achieve this task, two steps were necessary: first step is to (center) subtract its average, and second step is to (scale) divide by its standard deviation. In this manner, the input can still be fed into the standard logistic function and work with the backpropagation algorithm implemented.

In order to measure the performance of the neural network selection scheme defined as the performance rate, the following formula is used:

$$PR = 100 \times \frac{CorrectlyClassified}{TotalWebServices} \quad (3)$$

where PR represents the performance rate.

Since the number of hidden nodes using the backpropagation algorithm has to be defined prior to the training process, the structure could not be determined by the training algorithm. Hence, only a portion of the QWS data with equal number of samples for each classification group or Web service type is used to find the network configuration with the

TABLE V. CONSTANT USED FOR THE BPNN CONFIGURATION

Learning Rate	0.1
Momentum	0.1
Acceptable Error	0.005
Range	[-0.7 0.7]
Number of Epochs	647

TABLE IV. SAMPLE QWS DATA FOR SIX WEB SERVICES

Service ID	1	2	3	4	5	6	7	8	9	10	11
Service 1	45	83	27.2	50	97.4	89	91	43	58	100	1
Service 2	70	100	5.4	83	79.3	100	75	63	91	90	1
Service 3	120	100	3.6	91	82.7	100	84	94.17	10	75	2
Service 4	129.34	56	5.1	48	74.5	78	77	37.69	89	76	2
Service 5	125.44	100	13.5	86	86.4	78	80	125.33	91	86	1
Service 6	1292	100	9.3	67	65	78	84	1249	36	64	3

highest performance. Table V summarizes the constants used with different network configuration.

Using different network configurations, the highest performance rate is shown to be using the network configuration with eight input nodes, 14 hidden nodes, and 1 output node yielding 99.07% performance rate. Based on the highest performance rate of 99.07%, the network configuration of 10x14x1 was applied to training all of the samples in the QWS dataset. Only 77% of the training dataset were considered for the testing mode which yields a success rate of 95%. The results from the training phase of the neural network with 99.07%.

The convergence rate based on the number of epochs and

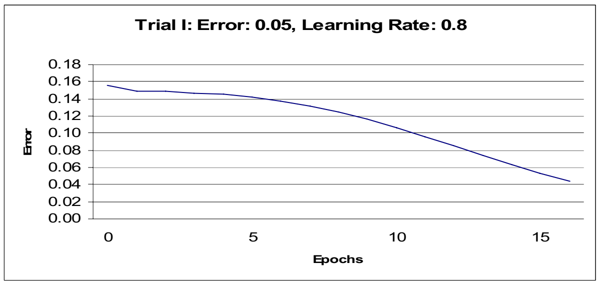


Figure 2. Number of epochs versus error (Trial 1) with learning rate of 0.8

iterative decrease in error is shown in Figure 2 (from Trial 1) with varying the network's learning rates.

In order to test for the accuracy of the system, an additional set of trials was conducted with setting the error convergence

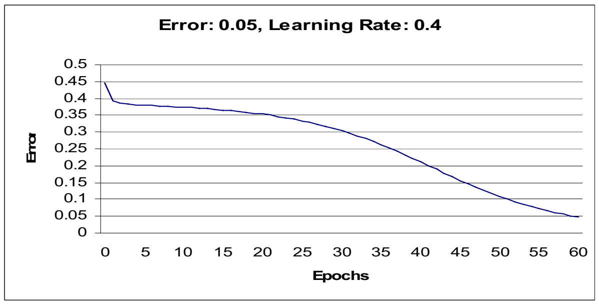


Figure 3. Number of epochs versus error (Trial 1) with learning rate of 0.4

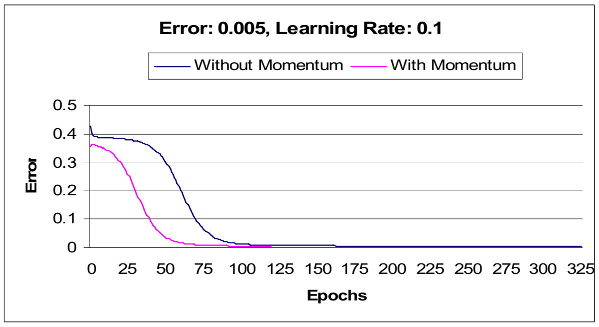


Figure 4. Number of epochs versus error (Trial 2) with learning rate of 0.1

rate to 0.05 and varying the learning rate for the neural network. Figures 3 and 4 show the obtained results.

To effectively test the accuracy of the system, a test for significantly lowering the error rate to 0.0005 yields the graph in Figure 5. Figure 5 shows the convergence of the backpropagation neural network to a solution within an acceptable error value of 0.0005. As illustrated in Figures 3-5, increasing the error value results in a smoother curve and the network converges into a solution at a faster rate (i.e. number of epochs decreases). Due to the fact that using momentum in a backpropagation algorithm takes into consideration the previous delta of the previous inter-connection, the system converges into a solution at a slightly faster rate than running

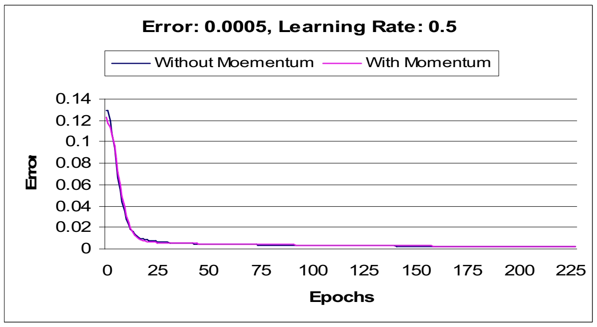


Figure 5. Number of epochs versus error (Trial 3) with learning rate of 0.5

the neural network without momentum as can be shown in Figure 4.

It was noted that varying the learning rate and the number of nodes have considerable impact on the performance of the system. The higher the leaning rate, the faster the system converges; however, there is trade off in terms of the decreasing performance of the network in selecting the best Web service. This is due to the fact that higher learning rates allow the backpropagation neural network to learn quickly and therefore does have the necessary time to adapt or learn enough to make accurate distinctions. As shown in Figures 4 and 5, the lower the acceptable error, the longer the network will converge.

Results from Figure 5 show that learning rates can have a significant impact on the speed and processing time of the neural network and that the network takes more iterations to converge into a solution. Results from Figure 5 also show that using smaller acceptable error such as 0.0005 will eventually take longer for the network to converge and find the most suitable Web service. These results suggest that the smaller the acceptable error, the longer the network will take to converge. Therefore, when choosing a neural network as an acceptable solution for selecting the best Web service match, a use of combination of learning rates and acceptable error values must be selected with caution particularly when running the system in real-time environments.

Finding the best combination of QWS properties for the selection of Web services can become very challenging. In the QWS dataset, the nine collected/measured QoS properties appear to have solid results with a reasonable performance.

However, it would desirable to determine how the network will behave if the number of QWS parameters is reduced. The outcome of this test can enable the determination of those QWS features that can be considered to provide valuable information or are essential when determining the overall quality of a Web service. In addition, this test will enable the creation of default QWS templates that vary the selections of QWS parameters such that they could be used to accommodate various clients' perceptions (i.e. cost-driven, compliance-driven, network-driven, among others).

To determine the accuracy of this analysis, an additional test was performed by removing several QWS parameters from the neural network and tested the network to evaluate its performance. We performed a test by removing the two QWS parameters: response time and throughput and determined that the network performance rate degraded significantly to 70.4% success rate. Applying the same test but slightly the QWS parameters removed (i.e. removing successability and reliability), the network performance is 87.3%. The outcome of these events suggest that the more QWS parameters we have, the higher the performance of the neural network.

VII. Conclusion

A backpropagation based neural network has been presented in this paper for the purpose of discovering Web services of interest based on service classification. The use of non-functional properties of Web services can significantly improve the probability of having relevant output results. To achieve this task, we used the publicly available QWS Dataset as inputs for the neural network. The feature set is mainly dependent on several factors such as response time, throughput, availability, compliance, among others. Results show that there exists a relationship between possible non-functional properties although having non-uniform metrics. The use of neural networks provides a way to optimize the selection of the best available Web service. The average performance rate of the neural network is 95%. In all of the three trials conducted for testing this system, the neural network always converged into a solution which suggests that the use of neural networks in discovering the most suitable Web service positively can be used.

The proposed solution provides an effective discovery mechanism for finding the high quality Web services based on non-functional properties; however, there is room for improvement. It is observed that the backpropagation neural network takes long time during the training mode due to the large data size which could become an issue when implementing such system in real-time manner. In addition, the ability of the system configuration to quickly adapt to current data being fed into it might become infeasible since the proposed method defined the number of hidden nodes prior in the training mode. Nonetheless, the proposed method has shown that neural networks can be applied and used for applications in the discovery and selection of Web services as a starting point in which can serve as basis for other types of neural networks. The ability to use other types of neural networks such as ARTMAPs, Fuzzy ARTs, or Self Organized Map is a solution that can be explored in the future and compared against the backpropagation algorithm.

References

[1] Al-Masri, E., and Mahmoud, Q.H., A Context-Aware Mobile Service Discovery and Selection Mechanism using Artificial Neural Networks, in Proceedings of the International Conference on E-Commerce (ICEC06), Fredericton, NB, Canada, pp. 594-598, 2006.

[2] Al-Masri, E., and Mahmoud, Q.H., "A framework for efficient discovery of web services across heterogeneous registries," IEEE Consumer Communication and Networking Conference (CCNC), pp. 415-419, 2007.

[3] Al-Masri, E., and Mahmoud, Q.H., "WSCE: A crawler engine for large-scale discovery of web services," ICWS, pp. 1104-1111, IEEE International Conference on Web Services (ICWS 2007), 2007, pp. 1104-1111.

[4] Al-Masri, E., and Mahmoud, Q.H., QoS-based Discovery and Ranking of Web Services, 16th International Conference on Computer Communications and Networks, 2007, pp. 529-534.

[5] Dong, Z., Similarity Search for Web Services, 30th VLDB Conference, August-September, 2004.

[6] Gunther, N., The Practical Performance Analyst, McGraw-Hill, New York, NY, USA., 1998.

[7] Hess, A., and Kushmerick, N., Learning to attach semantic metadata to web services. In Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, Oct 2003.

[8] Kumar, A., El-Geniedy, A., and Agrawal, S., A Generalized Framework for Providing QoS based registry in Service-Oriented Architecture, Proceedings of the IEEE International Conference on Services Computing, July 2005, pp. 295-301.

[9] Larkey, L.S., Automatic Essay Grading Using Text Classification Techniques, ACM SIGIR, 1998.

[10] Lin, W., Li, C., Chao, K., and Younas, M., Fuzzy Consensus on QoS in Web Services Discovery, Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Vienna, Austria, April 2006, pp. 791-798.

[11] Liu , Y., Ngu , A., Zeng, L., QoS computation and policing in dynamic web service selection, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, May 19-21, 2004, New York, NY, USA

[12] Makripoulias, Y., Makris, C., Panagis, Y., Sakkopoulos, E., Adamopoulo, P., and Tsakalidis, A., Web Service discovery based on Quality of Service, Proceedingss of the ACS/IEEE International Conference on Computer Systems and Applications, Dubai, UAE, March 2006, pp. 196-199.

[13] Maximilien, E., Singh, M., Conceptual model of Web service reputation. ACM SIGMOD Record, 31(4), December 2002.

[14] Ran, S., A Model for Web Services Discovery With QoS, ACM SIGecom Exchanges 4(1):1-10, 2003.

[15] Sheth, A., Cardoso, J., Miller, J., Koch, K., Web Services and Grid Computing, Proceedings of the Conference on Systemics, Cybernetics and Informatics, Orlando, FL, July 2002.

[16] Sreenath, R.M., and Singh, M.P., Agent-based service selection. Journal of Web Semantics,Vol 1, issue 3, 2004.

[17] The QWS Dataset, http://www.uoguelph.ca/~qmahmoud/qws, Last accessed January 2008.

[18] Yang, Y., and Pedersen, P., A comparative study on feature selection in text categorization. In International Conference on Machine Learning, 1997.

[19] Yu-jie, M., Jian, C., Sheng-Seng, Z., and Jian-hong, Z., Interactive Web service choice-making based on extended QoS model, Journal of Zhejiang University, Science A, Vol 7, No 4, April 2006, pp. 483-492.

[20] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q.Z., Quality Driven Web Services Composition Proc. 12th Int'l Conf. World Wide Web (WWW), May 2003.

[21] Al-Masri, E., and Mahmoud, Q.H., "Investigating Web Services on the World Wide Web". 17th International World Wide Web Conference (WWW2008), pp. 795-804, 2008.