Probabilistic QoS Analysis of Web Services

Waseem Ahmed, Yong Wu

Probabilistic QoS Analysis of Web Services

Waseem Ahmed[1], Yong Wei Wu[1]
[1] Department of Computer Science and Technology,
Tsinghua University, Beijing, China
{Waseem Ahmed} amw.inbox@gmail.com

Abstract. In such a competitive world, quality assurance can make the difference between a successful business and bankruptcy. For Internet services, the presence of low performance servers, high latency or overall poor service quality can translate into lost sales, user frustration and customers lost. In this paper, we propose a novel method for QoS metrification based on Hidden Markov Models. The techniques we show can be used to measure and predict the behavior of Web Services under several criteria, and can thus be used to rank services quantitatively rather than just qualitatively. We demonstrate the feasibility and usefulness of our methodology by drawing experiments on real world data. Our results have shown how our proposed methods can help the user to automatically select the best available Web Service based on several metrics, among them system predictability and response times variability.

Keywords: Hidden states, Probability, Quality of Service

1 Introduction

The Internet made the world a smaller place. Companies from all around the world may now compete over different service offerings not only with their local adversaries, but do now under a global scale. Escalating the competition and lead in industry segment can often be a matter of offering and, perhaps even most importantly, assuring the good quality of the services offered. In the Web this should be no different; controlling quality for Web Services (WS) is done by enforcing Quality of Service (QoS) policies and assuring needed quality conditions are always met.

On the user's side, the increased number of services means more and more offerings to choose from. Unfortunately, due the explosive growth in the number of WSs available in the world, selecting the best WS to solve a given task has become a quite challenging task. Currently, users cast their choice based on the reviews and experiences of other users. User-created ranks are often the first resource for finding reliability information regarding a particular service, often given in terms of response time, throughput, availability, security and reliability. Interestingly enough, data quality has never been considered as a key factor when analyzing QoS parameters.

There is no standard way, however, for the users to weigh their options directly and individually, for themselves. This paper aims to fill this gap providing a standard

way to measure and assess WS quality using Hidden Markov Model (HMM). Although web service reliability can be defined as producing cohesive results when invoked by different users with similar parameters [1], even though, sometimes web service even with best rank provides different results to end users. Systems which are designed to produce different results to different users at same time interval are out of the scope of this paper.

Existing papers such as [1-6] have discussed in detail QoS attributes of web services in terms of response time, throughput, reliability, availability. Nonetheless, there is still a lack of analyzing quality of data received from web services. Users across the IT industry associate bad quality of data or difference in response of web services against same request at same time with improper use of technology such as:

- Improper use of instance variables
- Incorrect caching
- Wrong mapping of data in lookup tables (in case of DB operation)
- Service is unable to recognize received category
- Servers are behind cluster and node responsible for reply during time t behaves badly
- Improper or mutated coding

However, as web services are owned and hosted by other organizations, most of the above mentioned aspects are difficult to monitor. In this paper we have designed a framework based on Hidden Markov Model (HMM) that will help end users to find a relation among web service responses and different hidden states producing them. Later, we have further extended this framework to predict behavioral patterns of these hidden states that will help users in making decision for web service selection.

In our framework, we have randomly selected web services with similar functionality (e.g. in our case it represent weather forecasting) from the list provided by [4] (who have claimed to have almost all available services by crawling web) and webserviceslist.com to analyze quality of data, that were ranked as best by different users around the world (as shown in table1). Status of all selected web services with similar functionality is more or less similar as described in [4].

Table 1 Web services with similar functionality along with their Ranks

Web Services	Count
Total web services with similar functionality(Weather forecasting)	23
Available	11
Broken link	8
Resource Cannot be found	3
Security exception	1

With HMM QoS attribute of WS in terms of data can be analyzed in two stages.

- Stage one will require us finding relations among hidden states in a remote WS and different data categories produced by web services.
- The second stage requires us to use this information to find probability of result produced by hidden states in future.

Our contribution in this paper can be summarized as below:

- We have analyzed the reason of variation in data generated by web service when invoked by different users at the same time with same input parameters.
- Defined a mechanism to build a relation among web service response / result and various hidden states responsible for producing it.
- Predicted the probability of variation in data / response of web services during nth time interval to select WS with better QoS attributes in terms of data.

The rest of the paper is organized as follows: section 2 introduces related work section 3 describes details about our conceptual framework section 4 presents our experiment and results and finally section 5 concludes the paper.

Table 2. Independent Variables

#	Independent variables
1.	Status (S)
2.	Temperature (T)
3.	Visibility (V)
4.	Due Point (D)
5.	Humidity(H)
6.	Wind(W)
7.	Pressure (P)

Table 3. Common Hidden States

#	Hidden States
1	Wrong Data Mapping
2	Bad Node in server clustering
3	Mutated coding
4	Composite WS

2 Related Work

Analyzing QoS attributes of remote web service is one of the important research areas in SOA based distributed applications. Most of the researchers have analyzed these attributes and proposed frameworks to facilitate end users for selecting or integrating web service with better QoS attributes. In this section we have presented a review of existing methods or framework proposed by different researchers. S. Maheswari and G.R. Karpagam [5] have proposed a framework that considered seven QoS attributes i.e. Response time, Execution Time, throughput, scalability, reputation, accessibility, and availability for better web service selection. Yilei Zhang and Zibin Zheng [3] have proposed model-based QoS prediction framework called WSPred. Their main contribution was time-aware personalized QoS prediction approach that analyzes latent features of users, service and time by performing tensor factorization. Emra and Pinar [1] proposed a method where they tracked QoS parameters automatically when required. However the issue with this approach is that, they did not consider network latency or communication delay in their calculations. Daniel A. Menascé [6] have proposed a way to calculate throughput of web services that have been used in single web service to accomplish its task. Ping Wang [7] has used fuzzy logic to locate and select web services based on user ratings. Other papers that have been produced to estimate QoS attributes to select better web service for integration are San-Yih Hwang [8], Vuong Xua [9], Xifeng [10] ,Hong Qing [11], Chunli [12], Wei, Z. [13].

Hidden Markov Model has already been used in analyzing quality factors of distributed computing systems. Nonetheless, they have their own issues, constraints and shortcomings. For instance Vathsala and Hrushikesha [14] have used HMM for predicting response time pattern of web services for different network's hidden states, however they did not consider the reliability of various hidden states with in the remote web service as discussed in Table II. As survivability also affects performance of any application, LeiLei Chen [15] has designed a framework to evaluate survivability of SOA based application using Hidden Markov Model. The main idea of their framework revolves around monitoring activities based on service logs or run time statistics provided by service provider. The problem with this approach is that it is restricted to statistics which have been provided by the service provider itself. Besides, the author did not provide a discussion about the possible hidden states and other probabilistic characteristics inherent to WSs. The HMM has also been successfully used in the prediction of other QoS aspects for SOA applications. One of such works is the work by Rahnavard and Meisam [16], who have used HMMs to detect WS anomalies, such as intrusion detection. However, their strategy could not be used to gauge arbitrary QoS attributes of WSs. Similarly, Flex Selfner [17] has proposed the use of HMMs to categorize and distinguish error patterns leading to failures. This author also suggested a mechanism for predicting the future occurrence of failures or errors. Zaki Malik[18] has used HMM to assess failures during certain time in future. In short HMM has been successfully used to analyze various aspects in distributed computing systems. The reliability of a WS can be established only once the reliability of hidden states have been ensured [20]. In this paper we have design a framework based on HMM for estimating probabilistic insight details of web service.

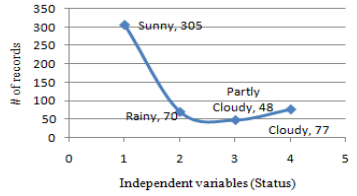
3 HMM Based Quality of Service Estimation

For probabilistic QoS analysis of web services in terms of data variation, our strategy is based on following steps:

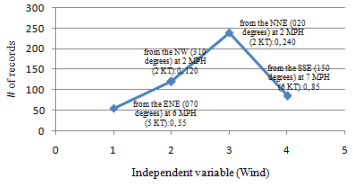
- Analyzing variation among data values of web services response when invoked by a number of users with similar input parameters.
- Estimating current state of internal system of WS using HMM and then defining probabilistic relationship among data values and various hidden states.
- Predicting behavioral pattern of hidden states for analyzing data variance during nth time interval.

3.1 Similarity Analysis

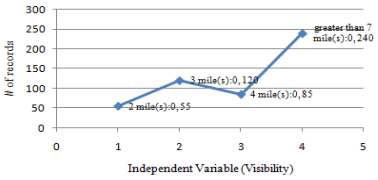
Estimating QoS attribute in terms of data variance requires us to elucidate web service response into a set of independent variables. This will help us to analyze data variation in more detail. For instance in case of weather forecasting, the consequent response can be divided into N number of different independent variables as defined in table 2. As each web service is being invoked by M number of users and we have K number of web services having similar functionalities. To find the similarity among



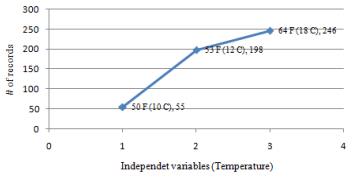
a. Variance in Status variable (as mentioned in table2)



b. Variance in wind



c. Variance in Visibility



d. Variance in temperature

Fig. 1. Variance in result when WS invoked by more than 500 parallel threads.

such data values we can represent above information in 3-dimentional N x M x K matrix. Web services with higher rank were invoked by 500 parallel threads with same input parameters in distributed environment and results were analyzed as shown in Fig. 1. It is apparent from Fig.1 that web services even with higher rank are replying with uncertain results during time interval t. Fig.1a shows variation in status (Sunny, Rainy, Partially cloudy, Cloudy) received by different users for same request parameters, whereas Fig.1 (b, c and d) depicts variation in independent variables (Wind, Visibility, and Temperature). For proof of concept we have shown only some independent variables, nonetheless, we have found variation in all independent variables. Region specific WS for instance weather forecast (US only) have produced better performance in comparison to other web services.

3.2 Quality of Service Analysis with Hidden Markov Model

Similarity analysis shows that independent variables can have different values within one observation when invoked in parallel with same request parameters. To figure out consequent data categories, it is therefore essential to analyze QoS attributes in terms of data variance. Then computing most likely hidden states and observation sequence using HMM, these categories can be linked with certain hidden states inside a WS. For instance, if observations for weather forecasting WS (as shown in table II), produced by finite number of hidden states (as shown in table III), then HMM can help us to establish a probabilistic relation between hidden states and consequent sequence of observations. There are two fundamental assumptions in our approach:

- Consequent observations are linked with execution pattern of hidden states with in a remote web service. This linkage can give us probability of possible scenarios used in implementation of WS. “Execution pattern” defines situation where sometime more than one hidden state is producing similar observations.

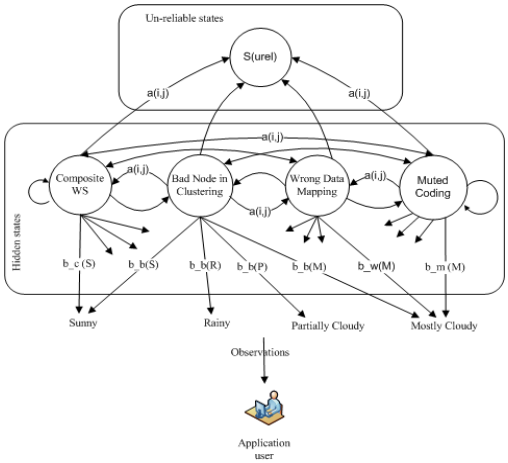


Fig. 2. Mappy variance in generating results to hidden markov model

It is important to analyze relevant hidden states along with other QoS attributes such as response time and throughput, which will further help us to predict probability of scalability of hidden states in future.

- States responsible for generating data are hidden and unknown.

The HMM have been successfully used in pattern recognition applications[17]. The first assumption is based on the fact that every hidden state has some special functionality linked with it. For instance, sometimes it is required to connect with database to verify certain results or call another web service or performing heavy calculations etc. Based on the execution of a certain hidden state the system may lead to a similar pattern of data output. As per first assumption these states can be identified by recognizing execution pattern. Whereas the second assumption perfectly matches with definition of HMM, as these states are hidden and produce results in time t. Furthermore, any hidden state can generate results when invoked during WS execution or access from other hidden states during error propagation, so model is of ergodic type. Based on these assumptions we have used HMM to find insight details of remote web service.

Estimating QoS attribute of WS in terms of data variance requires analyzing response time, throughput and quality of result produced. Response time represents duration which a web service is taking in executing some operation excluding network latency and communication delay. Throughput is the amount of work done by the web service within specified period of time. It is possible for certain hidden states to produce similar observations despite of having different implementation. Thus for a given time interval during various service invocations we can define feature vector including values defined in table III and by considering WS description to predict probability of implementation of hidden states. Because of difference in implementation of hidden states clear identification among feature values is required which in machine learning is referred to as Feature Normalization [21]. These features may be categorized in terms of data mapping, server clustering, mutated coding, calling other web services. This will help to define initial transition and emission probabilities of hidden states. Fig.2 shows general implementation of a web service with different observation sym-

Services / # of users	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	.....
Service1	R	UR	R	R	R	R	R	R	UR	R	R	R	R	R	R	UR	R	R	UR	R	R	.....
Service2	R	R	R	R	R	R	R	R	R	UR	R	R	R	R	R	R	R	R	R	UR	UR	.....
Service3	R	R	R	R	UR	R	R	UR	R	R	R	R	R	R	UR	R	R	R	UR	R	R	.....

Fig. 3: Training sequence

bols dependent on hidden states. In our framework, status(s) (as defined in table II) can be further divided into sub categories such as sunny, rainy, partially cloudy, and mostly cloudy. At the time user receives inconsistent data set underlying hidden state transits to an unreliable state, labeled as S_{urel}. Any state defined in Table III can be transit to an unreliable state. By initializing HMM parameters it can be ensured that the model transits to unreliable state once inconsistent data occurs in the training sequence. Emission probabilities are represented by relevant hidden state and output value. For instance, probability of output value "Mostly Cloudy" from hidden state "mutated coding" is represented by b_m(M). Here b represents observation probability matrix, m shows "mutated coding and M represents output value "Mostly Cloudy". So we can define various parameters of HMM as:

- States: S number of states where each state will have unique output dependent on its functionality (Fig. 2).
- Observations: Distinct output observations V i.e. categories defined in table I, such that output observation at time t is O_t where sequence of observation is $O= O_1, O_2,..., O_t$
- $A_{i,j}$ represents transitional probability of hidden state S_i following S_j.
- $B_{i,j}$ represents probability of hidden state generating output being produced from state S_j.
- Initial state distribution π

States as defined in table III may exist in one web service or there may have at least one or more states available. As per definition of HMM we have:

$$\lambda = (A, B, \pi) \tag{1}$$

3.3 Data Quality Prediction

State of WS during time interval t producing data D can be considered as vector of probabilities that WS is in hidden state S_i during time interval t having observations $o=\{O_1, O_2,..., O_n\}$. Current state of WS can lead us to predict $P_{urel}(s_k)$ of WS during kth time interval under various operational conditions. Where $P_{urel}(s_k)$ is the probability of state s_k that the Markov process defined by Hidden states produces data d_k during time interval k is unreliable. Current state of WS during time interval t can be computed with the help of HMM i.e using VITERBI algorithm:

$$\delta_t(i) = \max_{HS_1, HS_2, ..., HS_{n-1}} P(HS_1, HS_2, \ldots, HS_{n-1}, HS_n = i, O_1, O_2, ..., O_n \mid \lambda) \tag{2}$$

Here $\delta_t(i)$ represents the state of WS i.e. it represent maximum probability (computing maximum over all possible hidden states sequences) that the model went through

hidden states $HS_1, HS_2, \ldots, HS_{n-1}$ and the system is in state i at hidden state n. i.e. $HS_n = i$ while observing $O_1, O_2, \ldots, O_n$. To detect data quality one has to define valid data values for each data element collected so that the system would know what we are measuring against. In our framework, we are concerned only with data variance so we counted them where independent variables had differences in values when invoked by same input parameters by multiple threads (as shown in Fig.1). The data with maximum count i.e. received by most of the users in parallel invocation is considered to be reliable. Later we linked each of the counted value with corresponding hidden state using Eq.2. Now if we define criteria for valid data values then it can be analyzed that which particular state is not producing data as required. However in this paper, we are dealing only with data variance and this can be computed by verifying data values of all independent variables available in the data as defined in Section III-A using relation below:

$$\text{Rel (D)} = 1 - \sum_{i=0} n \left(V_i / D_k \right) \tag{3}$$

Here V_i represents the number of independent variable (as mentioned in table-1) in data D_k produced during time interval k. The eq.3 can be used recursively to find data variance in whole data D. The Rel(D) will be considered as unreliable, provided one or more independent variable in eq.3 will have invalid value. The probability $P_{urel}(s_k)$ is calculated by "First Passage Time Distribution". Let T_k be the time (known as First Passage Time) when hidden state s_k produces data Ds_k then:

$$T_k = \text{MIN (Rel } (Ds_k): S_k = S_{urel}) \tag{4}$$

Where S_{urel} represents the unreliable state, which implies hidden state S_k has produced unreliable data Ds_k during time interval k. Probability distribution among hidden states can be computed as below:

$$P_{urel}(S_k) = \sum_{i=0} n \, P\,(T_k \leq n \mid S_j = i)\; P\,(S_j = i) \qquad \text{s. t.} \quad j=0 \tag{5}$$

Where P (S_j= i) is the probability that WS is in hidden state j at current time as computed in eq.2, and P (T_k <= n | S_j = i) is the probability of going through hidden state S_{urel} and computing data reliability during kth time interval starting from j=0 which can be recursively computed with the help of the Baum-Welch algorithm[19]. The eq.5 represents probability distribution that the system produces unreliable results S_{urel} during nth time interval at time k which can be further scrutinized using dynamic programming to efficiently compute for various time intervals.

3.4 Training the Model

To train the model we have used observation sequences obtained during real web services invocations as described in similarity analysis section i.e. Section 3.1. These sequences are first labeled as "unreliable" using Eq.3, where at least one observation symbol i.e. independent variable has inconsistent value. In our framework such res-

ponses are modeled and represented by hidden state S_{urel} as shown in Fig.2. Whenever data with uncertain results is obtained, underlying state transits to unreliable state, i.e. S_{urel}. The hidden state S_{urel} can be any state which is defined in Table III. Observation sequence having uncertain values is shown in Fig.1 (b, c and d) where occurrence of uncertainty is indicated by difference in values. Training sequences from this information can be obtained by defining 'R' for reliable and 'UR' for unreliable value in observation sequence using the strategy defined in Section 3.4 as shown in Fig.3. Each column in Fig.3 represents the data consistency of the WS invocations. Each row in Fig.3 represents responses of single service invoked by 500 parallel threads; however, for proof of concept we have shown only a few values. Then by initializing HMM parameters in Eq.1 i.e. initial, transition and emission probabilities, such that states representing as unreliable S_{urel} are the only states that produce results with "UR', it can be ensured that model transits to unreliable state when uncertainty appears in the training sequence.

4 Experiments and Results

Based on domain information about implementation complexity of various computing techniques as described in table 1, initial guesses for probabilities can be exploited to:

- Adjust model parameters and determine current state of the system. Find the relation among output value and hidden states
- Predict QoS attributes of various hidden states using training sequences based on real data for various web services having similar functionality and select the WS with better QoS attribute.

In our experiment we have selected two web services with higher rank as mentioned in table-2 and invoked them using 500 parallel threads in a distributed environment. Fig.4b and Fig.5b represent data variance of independent variables Wind and Status (sunny, rain, cloudy, partially cloudy) respectively, as defined in table-1. Purpose of this experiment was to use our proposed model for analyzing their QoS attribute in terms of variance of data for selecting better web services and to predict their QoS values for anytime in the future.

4.1 Adjusting the Model Parameters

To predict the QoS attribute of hidden states, it is necessary to train the model to get estimated transition and emission probabilities. These values are then used in eq.2 to compute most probable hidden state sequences. Purpose of training the model is to find the optimal HMM parameters i.e. A, B & π such that the model best fits the training sequences. Baum-Welch algorithm a particular case of expectation-maximization (EM) is used to train the model. It iteratively improves the basic model which provides convergence to local optima. After training the model, current and future state is predicted using VITERBI algorithm as discussed above.

Wind Status	# of records
greater than 7 mile(s):0	241
2 mile(s):0	55
3 mile(s):0	119
4 mile(s):0	85

b. # of records for each independent variable

S-Successful, US - Unsuccessful				
	R	UR	S (%)	US(%)
State 1	153	53	30.6	10.6
State 2	88	193	17.6	38.6
State 3	0	0	0	0
State 4	0	13	0	2.6
R-Reliable, UR-unreliable				

d. # of records emitted by each hidden state

S-Successful, US - Unsuccessful				
	R	UR	S(%)	US(%)
State 1	100	0	20	0
State 2	0	238	0	47.6
State 3	141	0	28.2	0
State 4	0	21	0	4.2
R-Reliable, UR-unreliable				

f. Predicted records emitted by each hidden state

S-Successful, US - Unsuccessful				
	R	UR	S(%)	US (%)
State 1	101	0	20.2	0
State 2	0	208	0	41.6
State 3	122	0	24.4	0
State 4	9	60	1.8	12
R-Reliable, UR-unreliable				

h. Actual records emitted by each hidden state

Fig. 4: QoS attribute of Web services (WS1)

4.2 Current State

The state of component web service during time interval *t* is a vector of probabilities that system is in the hidden state HS_i when observation O_i is observed. VITERBI algorithm is used to calculate the most probable hidden state sequence (as discussed in section 3.3) that has generated the training sequence as shown in Fig.4 & Fig.5. Fig.4d and Fig.5d represent the current state of various hidden states of two web services W1 and W2 respectively. It can be analyzed that both web services produce variance in data because of inconsistent behavior of hidden states 1 and 2. Although these services are ranked as best by service users, even though both the services are producing over 50% data variance when invoked in parallel with same input parameters at the same time. To select better WS we further elucidated received results in more detail i.e. which particular hidden state of both web services is producing relatively better result. For instance, State1 of WS2 in Fig.5d shows that it has produced 44% of successful results whereas State1 of WS1 in Fig.4d indicates that it has produced 30% of overall successful result. This implies that if we can analyze or predict QoS attribute of each hidden state during the nth time interval, then we can select a better WS among the list of functionally equivalent web services.

4.3 Predicting Data Variance in Terms of Hidden States

As HMM is normally used to recognize patterns, therefore to predict behavior of the hidden states, idea is to classify suspicious data patterns i.e. patterns with observation symbols “UR”. This classification will indicate upcoming suspicious patterns. As per proposed technique, data values in a training sequence are divided into equal lengths slots. These slots having observations symbol “UR” are termed as “unreliable”.

First the model is trained using training sequences. Then based on trained HMM current status of the hidden state’s behavior is analyzed using VITERBI algorithm. Later, based on current state, future behavior of hidden states is predicted by calculat-

Weather	# of records
Sunny	278
Cloudy	79
Rain	79
Party Cloudy	64

b. # of records for each independent variable

S-Successful, US-Unsuccessful				
	R	UR	S(%)	US (%)
State 1	220	91	44	18.2
State 2	58	120	11.6	24
State 3	0	0	0	0
State 4	0	11	0	2.2
R-Reliable, UR-unreliable				

d. # of records emitted by each hidden state

S-Successful, US-Unsuccessful				
	R	UR	S(%)	US (%)
State 1	136	0	27.2	0
State 2	105	0	21	0
State 3	0	96	0	19.2
State 4	37	126	7.4	25.2
R-Reliable, UR-Unreliable				

f. Predicted records emitted by each hidden state

S-successful, US-unsuccessful				
	R	UR	S(%)	US(%)
State 1	110	0	22	0
State 2	88	20	17.6	4
State 3	0	79	0	15.8
State 4	30	173	6	34.6
R-reliable, UR-unreliable				

h. Actual records emitted by each hidden state

Fig. 5: QoS attribute of Web service (WS2)

ing “first passage time distribution” into unreliable state. Fig.4 (f and h) and Fig.5 (f and h) represent the predicted and actual state of hidden states of web services WS1 and WS2 respectively. It can be observed from the predicted value of WS1 in Fig.4f that only State1 and State3 will produce consistent results during time interval t. Nonetheless, State2 and State4 will produce inconsistent results. These predictions will help end users to design their system in a way that can entertain responses produced by only State1 and State3. Fig.4h shows actual values of the same web services W1 during time interval t. It can be seen that predicted values are almost similar to actual values except some consistent values which were also produced by State4. However, this is a small number which can be ignored to make the system reliable. Whereas, Fig.5f and h shows predicted and actual values of web service WS2 responses during time interval t. Predicted and actual values are almost similar in numbers except a marginal difference which can be ignored, however, in this case State2 and State4 have inconsistent behavior. Both the states are randomly generating consistent and inconsistent results which are hard to ignore during the live execution of the system. Therefore, it will be easy for end users to decide which particular web service can incorporated in the system. Such as in this case study WS1 appears to be more suitable compared to WS2. With these results it is apparent that HMM can predict probabilistic QoS attributes of remote WS in terms of data variance, for any time interval in the future. Our model can be used to further analyze probabilistic scalability of various hidden states under specified circumstances such as by increasing user load or increasing communication delay.

5 Conclusion

In this paper we have explained with experiments how HMM can be used to analyze and predict QoS attribute of a web services in terms of data discrepancy. Predicting QoS attributes will then help end users to select a better web service among the list of functionally equivalent web services. We have performed our experiments on real world web services. Later, we have analyzed in detail the behavior of web services for a different set of users having similar input parameters. Our framework gives information about the probabilistic insight of any remote web service. It can further predict QoS attribute of these hidden states in terms of data variance and can help to further examine scalability of these hidden states.

Acknowledgment:

I am extremely grateful to Cesar Roberto de Souza (Federal University of Sao Carlos) and for his support, encouragement & proofreading of the draft version of my paper.

6 References

[1] Askaroglu, E. and P. Senkul. Automatic QoS evaluation method for web services. in Computers and Communications (ISCC), 2012 IEEE Symposium on. 2012.

[2] D'Ambrogio, A., A Model-driven Approach to Describe and Predict the Performance of Composite Services. WOSP'07, 2007. February 5–8, 2007, Buenos Aires, Argentina.

[3] Yilei, Z., Z. Zibin, and M.R. Lyu. WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services. in Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on. 2011.

[4] Zibin, Z., Z. Yilei, and M.R. Lyu. Distributed QoS Evaluation for Real-World Web Services. in Web Services (ICWS), 2010 IEEE International Conference on. 2010.

[5] Maheswari, S., *QoS Based Efficient Web Service Selection.* European Journal of Scientific Research, 2011. European Journal of Scientific Research.

[6] Menasce, D.A., *QoS issues in Web services.* Internet Computing, IEEE, 2002. **6**(6): p. 72-75.

[7] Ping, W., et al. A Fuzzy Model for Selection of QoS-Aware Web Services. in e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on. 2006.

[8] San-Yih, H., et al., A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Inf. Sci., 2007. **177**(23): p. 5484-5503.

[9] Tran, V.X., H. Tsuji, and R. Masuda, *A new QoS ontology and its QoS-based ranking algorithm for Web services.* Simulation Modelling Practice and Theory, 2009. **17**(8): p. 1378-1398.

[10] Xifeng, W., et al. *Ontology-Based Reliability Evaluation for Web Service*. in *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*. 2011.

[11] Hong Qing, Y. and S. Reiff-Marganiec. A Method for Automated Web Service Selection. in Services - Part I, 2008. IEEE Congress on. 2008.

[12] Chunli, X., L. Bixin, and W. Xifeng. *A Staged Model for Web Service Reliability*. in *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*. 2011.

[13] Wei, Z., et al. *QoS-Based Dynamic Web Service Composition with Ant Colony Optimization*. in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*. 2010.

[14] Vathsala, A.V. and M. Hrushikesha, Using HMM for predicting response time of web services, in Proceedings of the CUBE International Information Technology Conference. 2012, ACM: Pune, India.

[15] Leilei, C., et al., Evaluating the Survivability of SOA Systems Based on HMM, in Proceedings of the 2010 IEEE International Conference on Web Services. 2010, IEEE Computer Society.

[16] Rahnavard, G., M.S.A. Najjar, and S. Taherifar. A method to evaluate Web Services Anomaly Detection using Hidden Markov Models. in Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on. 2010.

[17] Salfner, F., *Predicting Failures with Hidden Markov Models.* Proceedings of 5th European Dependable Computing Conference, 2005.

[18] Zaki, M., A. Ihsan, and B. Athman, Web Services Reputation Assessment Using a Hidden Markov Model, in Proceedings of the 7th International Joint Conference on Service-Oriented Computing. 2009, Springer-Verlag: Stockholm.

[19] Ramage, D., Hidden Markov Models Fundamentals. 2007.

[20] Waseem Ahmed, Yong Wei Wu, A survey on reliability in distributed systems, Journal of Computer and System Sciences, ISSN 0022-0000, 10.1016/j.jcss.2013.02.006

[21] Hoecke, S.V., *Modeling the performance of the Web service platform using Layered Queueing Networks.* Proceedings of Software Engineering Research and Practice. 2005, 627-633. , 2005.