Web Services Reputation Assessment Using a Hidden Markov Model*

Zaki Malik[1], Ihsan Akbar[2], and Athman Bouguettaya[3]

[1] Department of Computer Science, Wayne State University
Detroit, MI, 48202 USA
zaki@wayne.edu
[2] Department of Electrical Engineering,
Virginia Tech Blacksburg, VA. 24061 USA
iakbar@vt.edu
[3] CSIRO, ICT Center. Canberra, Australia
athman.bouguettaya@csiro.au

Abstract. We present an approach for reputation assessment in service-oriented environments. We define key metrics to aggregate the feedbacks of different raters, for assessing a service provider's reputation. In situations where rater feedbacks are not readily available, we use a Hidden Markov Models (HMM) to predict the reputation of a service provider. HMMs have proven to be suitable in numerous research areas for modelling dynamic systems. We propose to emulate the success of such systems for evaluating service reputations to enable trust-based interactions with and amongst Web services. The experiment details included in this paper show the applicability of the proposed HMM-based reputation assessment model.

1 Introduction

The next installment of the World Wide Web will be a shift from the current data-centric Web to a service-centric Web [14]. In this regard, the Web, services, and semantic technologies (e.g. in the form of ontologies) will come together to create an environment where users (and applications) can query and compose services in an automatic and seamless manner. The *Service Web* will build upon and extend the Semantic Web to treat services as first class objects. Web services are slated to be the key enablers of the new service computing paradigm [14]. A Web service is defined as a self-describing software application that can be advertised, located, and used on the Web using a set of standards such as WSDL, UDDI, and SOAP. The Service Web is expected to be a place where a large number of Web services will compete to offer similar functionalities [9]. Thus, enriching the Web with semantics would facilitate the organization and location of these services, and most importantly enable quality-based querying. It is expected that Web services would fully leverage the Semantic Web to outsource

* This work is funded in part by the U.S. National Science Foundation grant number 0627469.

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 576–591, 2009.

part of their functionality to other Web services [20]. In this case, some services may not have interacted before, while others may act maliciously to be selected, thus negatively impacting the quality of collaboration. A key requirement then is to provide *trust* mechanisms for quality access and retrieval of services [8].

Over the years, a number of techniques have been proposed for establishing trust online. These techniques fall under two main categories: security-based solutions, and social control-based solutions. The former includes mechanisms as authentication, access control, etc, while the latter is based on recommendations and reputation. In this paper, we focus on reputation as a means to establish trust among different services.

Reputation is regarded as a predictor of future behavior. It is a subjective assessment of a characteristic ascribed to one entity by another based on past experiences. In the context of the Service Web, we refer to the aggregated perceptions that the community of service requesters have for a given Web service provider as service *reputation*. Experimental studies have shown that people rely on reputation systems (e.g. eBay's Feedback Forum) to make trust-enabled decisions regarding their daily Web-enabled activities [8]. Reputation systems rely on the feedbacks or ratings provided by the members of the community for a given subject. At times, due to various reasons, majority of the members may not be willing to engage in the rating process. In such situations of *ratings scarcity*, the accuracy of the reputation system may be compromised. We address the issue of ratings scarcity by approximating ratings aggregation and *predicting* the reputation of a given subject based on historical data.

In terms of prediction accuracy, machine learning algorithms have provided better results over other traditional techniques [16]. For instance, Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs) exhibit high predictive power, especially with large data sets [16]. Since ANN performances depend greatly on the chosen architecture, and a user may need to perform extensive model training by considering almost every feature, we prefer to use HMMs for reputation prediction. Moreover, an HMM allows a user more control than an ANN, with comparable accuracy. HMMs have been used successfully in pattern recognition (voice and face), hand writing recognition, natural language domains, DNA sequence analysis, finger print matching, prediction of stock market prices, etc [16]. We build on that success to predict the reputation of Web services through HMMs.

Several reputation systems have been proposed in the literature. The spectrum of these reputation management solutions ranges from "purely statistical" techniques to "heuristics-based" techniques. While statistical techniques focus on providing a sound theory for reputation management, heuristics-based techniques focus on defining a practical model for implementing a robust reputation system. Bayesian systems [5], [13] and belief models [6], [24] are the major examples of purely statistical techniques. Bayesian systems work on binary ratings (honest or dishonest) to assess the reputation, by statistical updating of beta probability density functions. In a belief model, a consumer's belief regarding the truth of a ratings statement is also factored in reputation computation. The

techniques for combining beliefs vary from one solution to the other. For example, [24] uses the Dempster's Rule, while Subjective Logic is used in [6]. The complexity of purely statistical solutions has prompted researchers to present heuristics-based solutions. These solutions aim to define a practical, robust, and easy to understand/construct reputation management system. For example, [23] and [4]. In the following, we present a hybrid solution defining key heuristics, and a statistical model (HMM-based) for reputation assessment.

2 Web Services Reputation

A Web service exposes an interface through which it may be automatically invoked by Web clients. A Web service's interface describes a collection of operations that are network-accessible through standardized XML messaging [9]. Invoking a Web service involves three entities: the *service provider*, the *service registry* and the *service consumer*. The service provider is the entity that owns and/or manages the service. It advertises the capabilities of the service by publishing a description to the service registry. This description specifies how the service can be invoked (i.e., its address, operations, parameters, etc.) The service registry is a repository of service descriptions. Finally, the service consumer is the entity that invokes the service.

In traditional Web service models, service selection is not trust-based, and an invocation can be made directly after discovering the service through the registry. However, in our model this selection is based on the reputation of each individual service from the list retrieved through the service registry. The service consumer gathers the feedbacks of the providers from its peer service consumers, and then sorts the providers according to the assessed reputation. The higher the reputation of a service provider, the *better* the service. Service consumers then *invoke* the best available Web service through one of its listed operations. We assume that at the end of the interaction the service consumer *rates* the provider according to some pre-determined criteria (e.g., using an ontology[9], [23]). The service ratings are used to compute the provider reputations accordingly.

We view the reputation of a Web service as a reflection of its *quality*. The Quality of Service (*QoS*), is defined as a set of quantitative and qualitative characteristics of a system, necessary to achieve the required functionality of an application [17]. We adopt a similar definition of *QoS* and extend its application to the Service Web with related constraints (similar to [17], [10]). We term this as the quality of Web service (QoWS). QoWS is a mapping between a set of quality parameters defined through a common ontology, and a set of values or ranges of values. Examples of quality parameters include security, privacy preservation, a services' response time, availability, reliability, etc.

Let S and T be the set of provider Web services and the set of service consumers respectively. Let Φ be the universal set of quality parameters. Φ may be represented as a p-element vector (ϕ_1, ..., ϕ_p) where ϕ_k is the k^{th} quality parameter. Each Web service $s_j \in S$ advertises a promised quality $QoWS_p(s_j)$, which assigns values or ranges of values to each quality parameter ϕ_k. When a

service requester $x \in T$ invokes the service s_j, each quality parameter ϕ_k in Φ gets assigned a delivered quality value ϕ_k^{xj} (post-transaction completion). For this invocation of service s_j, the vector $QoWS_d(s_j, x) = \{\phi_1^{xj}, .., \phi_p^{xj}\}$ is called the delivered quality of Web service.

It is outside the scope of the current discussion exactly how values are assigned to different $QoWS$ attributes. We assume a service publication model presented in [17], where service providers publish their $QoWS_p$ values in the service registry (say UDDI) with the service descriptions ([20] proposes a similar technique). Other similar models where the QoWS information can be added to the WSDL file using WS-Policy, can also be used [8]. Post-transaction completion, observing the variation between $QoWS_p$ and $QoWS_d$, *reputation* values can be created [17], [20], [8].

We suggest that since the Service Web cannot be easily monitored due to its expanse, each service consumer records its own perceptions of the reputation of only the services it actually invokes. This perception is called *personal evaluation* ($PerEval$). For each service s_j that it has invoked, a service consumer x maintains a p-element vector $PerEval_j^x$ representing x's perception of s_j's reputation. Thus, personal evaluation only reflects the QoWS performance of a provider in the consumer's own view.

2.1 Reputation Assessment

A consumer intending to assess the reputation of a service provider may inquire several peer consumers in its community (the one its is registered with), and aggregate their respective personal evaluations for s_j. Identifying the entities responsible for collecting and disseminating reputations, and defining the procedures involved in such reputation exchanges are important aspects of a reputation management system, which require independent research. We assume a reputation collection model presented in [22], and extend it to the Web services domain using methods presented in [12]. Note that other collection models as [1], [17] can also be used. A single value is obtained as a result of the aggregation of personal evaluations collected. This derived value is defined as the service provider's *aggregated reputation* in that consumer's view. Different service consumers may employ different aggregation techniques. Therefore, the aggregated reputation value for the same provider may be different for each consumer, i.e., it may not be consistent across all consumers. Formally, the reputation of s_j, as viewed by a consumer is defined as:

$$Reputation(s_j) = \bigwedge_{x \in L} (PerEval_j^x) \tag{1}$$

where L denotes the set of service raters and $\bigwedge$ represents the aggregation function. It can be as simple as representing the union of personal evaluations where the output is a real number, or an elaborate process that considers a number of factors to assess a fairly accurate reputation value.

Equation 1 provides an approximation of how the service reputation may be calculated. In the following, we build upon this equation to define the "RATEWeb

metrics" for accurate reputation assessment. We aim to counter attacks related to deception in reputation management, i.e., identifying, preventing, and detecting malicious behavior of peers or a set of colluding peers acting as either service providers or raters. Problems as free riding, fake identities, ratings incentives, etc. are outside the scope of this paper.

Credibility of Raters: The foremost drawback of feedback-only based systems is that all ratings are assumed to be honest and unbiased. However, in the real world we clearly distinguish between the testimonies of our sources and weigh the "trusted" ones more than others [19]. A Web service that provides satisfactory service (in accordance with its promised quality ($QoWS_p$)), may get incorrect or false ratings from different evaluators due to several malicious motives. In order to cater for such "bad-mouthing" or collusion possibilities, a reputation management system should weigh the ratings of highly credible raters more than consumers with low credibilities [4], [3], [18], [15], [23]. In RATEWeb, the reputation score of the provider is calculated according to the credibility scores of the raters (used as the weight). Thus, Equation 1 becomes:

$$Reputation(s_j) = \frac{\sum_{x=1}^{L}(PerEval_j^x * C_r(x))}{\sum_{x=1}^{L} C_r(x)} \tag{2}$$

where $Reputation(s_j)$ is the assessed reputation of s_j as calculated by the service consumer and $C_r(x)$ is the credibility of the service rater x as viewed by the service consumer. The credibility of a service rater lies in the interval [0,1] with 0 identifying a dishonest rater and 1 an honest one. The processes involved in calculating raters' credibilities are described in detail in [8].

Personalized Preferences: Service consumers may vary in their reputation evaluations due to their differences in QoWS attribute preferences over which a Web service is evaluated. For instance, some service consumers may label Web services with high reliability as more reputable while others may consider low-priced services as more reputable. We allow the service consumers to calculate the reputation scores of the Web services according to their own *personal preferences*. Each service consumer stores its QoWS attribute preferences in a *reputation significance vector* (RSV). This allows the consumers the ability to weigh the different attributes according to their own preferences. Let $\phi_h(s_j, u)^x$ denote the rating assigned to attribute h by the service rater x for service provider s_j in transaction u, m denote the total number of attributes and RSV_h denote the preference of the service consumer for attribute h. Then, the local reputation for s_j as reported by service rater x is defined as:

$$PerEval_j^x = \frac{\sum_{h=1}^{m}(\phi_h(s_j, u)^x * RSV_h)}{\sum_{h=1}^{m} RSV_h} \tag{3}$$

Reputation Fading: Reputation information of a service provider decays with time [9][11]. Hence all the past reputation data may be of little or no importance.

For instance, a Web service performing inconsistently in the past may ameliorate its behavior. Alternatively, a service's performance may degrade over time. It may be the case that considering all historical data may provide incorrect reputation scores. In order to counter such discrepancies, we incorporate temporal sensitivity in our proposed model. The rating submissions are time-stamped to assign more weight to recent observations and less to older ones. This is termed as "reputation fading" where older perceptions gradually *fade* and fresh ones take their place. We adjust the value of the ratings as:

$$PerEval_j^x(t) = PerEval_j^x(t-1:t-v) * f_d \tag{4}$$

where $PerEval^{xj}$ is as defined above and f_d is the reputation fader. t is the current time instance and $t-1:t-v$ specifies the time interval from previous 1 to v transactions. In our model, the recent most rating has the fader value 1 while older observations are decremented for each time interval passed. When $f_d = 0$, the consumer's rating is not considered as it is outdated. The "time interval" is an assigned factor, which could be anywhere from a single reputation inquiry, ten inquiries or even more than that. All inquiries that are grouped in one time interval are assigned the same fader value. In this way, the service consumer can define its own temporal sensitivity degree. For example, a service can omit the fader value's effect altogether by assigning it a null value. We propose to use a fader value that can then be calculated as: $f_d = \frac{1}{\sqrt{P_u}}$, where P_u is the time interval difference between the present time and the time in which the rating was collected from the rater. This allows the convergence of reputation to a very small value as time passes. Note that the consumer can assign a group of ratings collected at different times to have the same time-stamp, and hence lie in the same time interval.

Incorporating the defined metrics together (denoted RATEWeb metrics), the equation for overall reputation calculation becomes:

$$Reputation(s_j) = \frac{\sum_{x=1}^{L} [\frac{\sum_{h=1}^{m} (\phi_h(s_j,u)^x * RSV_h)}{\sum_{h=1}^{m} RSV_h} * f_d * C_r(x)]}{\sum_{x=1}^{L} C_r(x)} \tag{5}$$

Through experimental evidence we have found that the above equation provides a comprehensive assessment of the reputation of a given service provider. Some evaluation results are presented in Section 3. For a thorough review, the interested reviewer is referred to [9]. Other aspects of our RATEWeb framework relating to reputation bootstrapping are defined in [10], and are outside the scope of this paper. As mentioned earlier, the providers' reputations calculated above may not always be available. This may either be due to the reluctance of the raters or other unforseen circumstances as power outages, network congestion, etc. We propose to use HMM-based "prediction" methods to evaluate service reputations based on past behavior in situations where the current feedbacks are not available.

The proposed methodology is shown in Figure 1. Each service consumer's HMM first trains itself using the feedbacks provided by its peers. Once a reliable

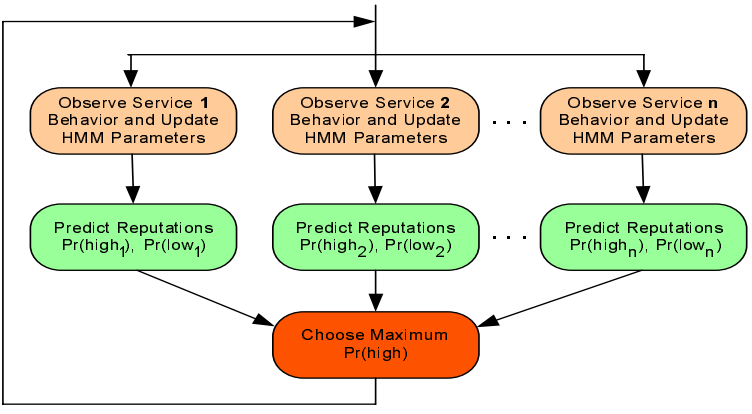

Fig. 1. Predicting Reputations using HMMs

model is developed, the high and low reputations of the services are predicted. In the next step, the service consumer compares all the predicted provider reputations. The provider that has the highest predicted reputation for the next time instance is chosen for interaction. After each interaction, the observed behavior values and present feedbacks are input to the HMM, and the model is refined.

2.2 HMM-Based Reputation Assessment

A Hidden Markov Model (HMM) is a finite state machine in which the observation sequence is a probabilistic function of a fixed number of states. In our case, it provides a probabilistic framework for modelling service reputations. Since their introduction in the 1970s, HMMs have proven to be very powerful prediction tools [16]. Some of the advantages of HMMs include: (1) strong mathematical and statistical basis, (2) more control through easy manipulation of training and verification processes, (3) mathematical/theoretical analysis of the results, (4) efficient prediction of similar patterns, and (5) ability to incorporate new knowledge robustly.

An excellent tutorial describing the basics and use of HMMs is available in [16]. In brief, an HMM (denoted ζ) is characterized by:

- the number of states in the model (N).
- the number of observation symbols per state (M), where each symbol corresponds to the actual output (here, reputation) being modelled.
- state transition probability matrix ($P = \{p_{ij}\}$), where p_{ij} represents the probability of transition from state i to state j.
- output probability matrix ($B = \{b_j(y_k)\}$), where $b_j(y_k)$ represents the probability of generating symbol y_k at state j.
- initial state distribution ($\pi = \{\pi_i\}$), where π_i gives the probability of being in a particular state at the start of the process.

With N and M (which depend on the observation data and the application objective) specified, an HMM is denoted as $\zeta = \{P, B, \pi\}$, with $\sum_j p_{ij} = 1$, $\sum_k b_j(y_k) = 1$, and $\sum_i \pi_i = 1$, where p_{ij}, $b_j(y_k)$, $\pi_i \geq 0$, $\forall i, j, k$.

There are established algorithms and techniques for the estimation of the parameters of an HMM. We have used the Bayesian Information Criterion (BIC) (one of the most accurate and frequently used order estimation techniques [16]) to estimate N for our reputation model: a 2-State HMM was selected. The estimation of the total number of states of HMM is an open question that still needs to be solved satisfactorily. "In the case of HMM, the problem of model selection (and in particular the choice of the number of states in the Markov chain component model) has yet to be satisfactorily solved" [7]. Similarly, in [2], it is stated that "the order estimation problem in an HMM is difficult because of our poor understanding of the maximum likelihood in HMMs." The likelihood of a process in case of HMMs, increases as the number of states increases. However, even though a higher state HMM is expected to perform well, it does not guarantee to provide optimal results [7], [2]. BIC is defined as:

$$\hat{x} = min[-2(\sup_{M^x} logPr(y_1^n)) + klog(n)] \tag{6}$$

where k is the dimension of a HMM, n is length of the observed reputations sequence, M^x represents the HMMs with different number of states, and $\hat{x}$ represents the selected HMM with optimal number of states. Using BIC over past reputations, we train HMMs with different number of states and obtain their corresponding log-likelihood (denoted l) values. The model with the minimum BIC value is then chosen. For instance, Table 1 shows the BIC order estimation process for a number of HMMs evaluated using experimental reputation data for different service providers. Experiment details are presented in the next section. Since 4.6861×10^3 is the lowest value obtained, BIC selects a 2-state HMM as the model of choice. Note that the number of HMM states that the BIC estimator selects, are specific to the training data. Thus, the number of states that different service consumers obtain for their respective HMMs, may also differ (since each consumer aggregates reputations according to his own preferences, knowledge, etc).

Note that all the models (2-state, 3-state, and 15-state HMMs shown) produce similar l-values, upon reaching the local maxima. Moreover, the number of iterations required to get to the local maxima are also similar. In terms of accuracy,

Table 1. Estimating BIC values of different HMMs

Model	k	$-l$	**BIC**
2-State HMM	4	2321.8056	4.6861×10^3
3-State HMM	9	2321.7719	4.7391×10^3
4-State HMM	16	2321.6676	4.8131×10^3
5-State HMM	25	2320.8490	4.9070×10^3
6-State HMM	36	2320.4865	5.0230×10^3

the different state models show similar results. But since greater the number of states in an HMM, greater is the complexity of the prediction process [21]. Therefore, we use the minimal state HMM (2-state HMM).

In defining M, we use a simple strategy that outputs one of two reputation symbols for the results obtained through Equation 5. One symbol represents a *trustworthy* service (i.e., high reputation), while the second symbol represents an *untrustworthy* (with low reputation) service. The choice of only two symbols is made only for simplifying the model explanation, and the number can be extended easily. Although the reputation values obtained from Equation 5 may be continuous, we distinguish the results to two discrete symbols by setting a threshold. For example, on a scale from 0 to 10 (0 being the lowest) with the threshold set at 5, reputation values graters than 5 are treated as trustworthy and untrustworthy otherwise. Other techniques (e.g., vector quantization) may also be used. Defining the system with just two reputation symbols suffices the need of our application. The service consumer either interacts with a trustworthy service, or it does not interact with an untrustworthy service. The need for "middle-ground" or "fuzzy" decision making arises only in cases where no trustworthy service is available. However, this can also be handled by adding a clause that lets the consumer choose the *best* provider from the remaining (untrustworthy) services.

In using the HMM, one major task is to find the probability (Pr) of generating the reputation sequence given the model ζ. This can be written as:

$$Pr(y_1^T|\zeta) = \pi B(y_1)PB(y_2)...PB(y_T) \tag{7}$$

where $y_1^T = [y_1, y_2, ..., y_T]$ with each y_k having either high or low reputation (i.e., one of the 2 symbols), and $B(y_k)$ denotes the probability of generating symbol y_k from different states. We use the "Forward-Only algorithm" to compute $Pr(y_1^T|\zeta)$. Moreover, we use the Baum-Welch Algorithm (BWA) [16] (a form of the Expectation-Maximization (EM) algorithm) to estimate the parameters, and find the model that best explains the observed data. Due to space restrictions, details of the algorithms are not presented here. The interested reader is referred to [16].

Figure 2 shows the steps (simplified to elucidate) involved in using an HMM to assess a provider's reputation. To decide on interacting with a particular service provider, feedbacks from different raters are collected and aggregated (to derive the provider's reputation). In situations of ratings scarcity, aggregate reputations from previous instances are used to predict future provider behavior. The reputations, initial conditions (if any), etc. are used as inputs to the BWA to extract HMMs (with different number of states). BIC is then used to estimate the optimal number of states for the HMM. The selected model is then used to predict future service reputations, to aid in the interaction decision process. Any subsequent re-estimations are performed on the selected model.

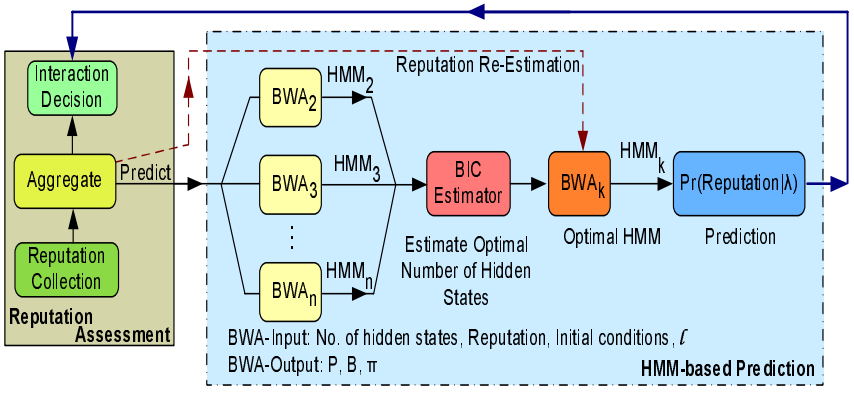

Fig. 2. HMM-based Prediction Processes

3 Evaluation

We performed preliminary experiments to evaluate the RATEWeb approach and show the accuracy of HMM-based reputation assessment. The experiments are divided into two phases. In the first phase, the effectiveness of RATEWeb metrics is evaluated. In the second phase, the assessed reputations from the first phase are used to train an HMM. The HMM is then used to predict future provider behavior. In the following, we provide the experiments related to HMM prediction (phase 2). Extensive evaluation of RATEWeb related to phase 1 is presented in [8], [9].

Setup: We created a Web services environment where the *actual* behavior of service providers is accurately captured, i.e, we can monitor each service's behavior. The providers' behaviors are simulated using data similar to the behavior of sellers at *eBay*. The service community consists of 100 Web services, and the interactions are conducted over 6000 time iterations. Although each QoWS parameter is rated individually, we use only the aggregated scalar aggregated reputation value to facilitate comparison. We assume that QoWS values are assigned accurately according to a pre-defined rating scheme. The minimum performance value is 0 while the maximum is 10.

The accuracy of the hidden Markov model is evaluated by observing the variance between the actual behavior and predicted reputation. We check the validity of our HMM-based reputation prediction by comparing the results with another formal prediction model (ANN) and with an ad hoc model in which no prediction is used. The type of ANN that we have used (through MATLAB) incorporates adaptive filtering which allows the network to adapt at each time step to minimize the error and predict in a relatively short time. The parameters of the HMM (from the processes in Figure 2) we use in the experiments are shown in Table 2.

Table 2. Parameters of the Chosen HMM

P	**B**	π
$\begin{bmatrix} 0.988 & 0.011 \\ 0.008 & 0.991 \end{bmatrix}$	$\begin{bmatrix} 0.051 & 0.948 \\ 0.990 & 0.009 \end{bmatrix}$	$\begin{bmatrix} 0.999 & 0.001 \end{bmatrix}$

The generation of the reputation sequence given the model is performed using the following procedure. We start from the π vector and select the initial reputation state by generating a uniformly distributed random number (within the desired reputation range). The reputation output symbol is obtained by again generating a uniformly distributed random number and then assigning a reputation output symbol according to the probability distribution of the B matrix. The next reputation state is then obtained by again generating a uniformly distributed random number and selecting the next reputation state according to the probability distribution of the previous reputation state from the P matrix. We then go back and find the reputation output symbol according to the probability distributions of different reputation symbols using the B matrix. We continue till the required number of reputation output symbols are obtained. An HMM is most useful for random data that shows some pattern. We have generated a data set in which service reputations oscillate between high (above 5) and low (less than 5). The reputation data is random such that the service provider can shift from a high to low state and vice versa at any time. The time spent in each state is random and not uniform. However, the data exhibits memory since the service follows the pattern of staying in one state and moving to the other after 'sometime.' Note that the service provider does not change states at almost every time instance (as in memoryless systems).

Figure 3 shows the original reputation data generated for one service provider, and the comparison of the original data against HMM-based and ANN-based predicted reputation. The original data is represented by a continuous line (in color: blue) while the predicted reputations are shown as dotted lines (in color: red). In Figure 3-A, the original values for 5000 iterations are shown. However, Figures 3-B and -C show the zoomed-in values for iterations 1200 to 1600 for explanatory purposes. We have trained the HMM and ANN over 1000 iterations and then predicted reputations for 5500 future iterations. The predicted reputations shown in Figure 3 are not *completely* identical to the original ones. There is some "error" associated with each reputation prediction. However, the error is not disruptive to the prediction process due to its small size. The predicted reputation values obtained using the HMM (Figure 3-B) and the ANN (Figure 3-C) are very close to the original reputation. Therefore, we can safely conclude that the generated values are representative of the original service behavior allowing fairly accurate trust decision making.

Both the prediction models predict the reputation in a fairly accurate manner. This proves that both ANN and HMM-based methods are viable. Note that the duration of the provider's "stay" in either state (high or low reputation) is random and no two intervals are equal. Still, the HMM and ANN are able

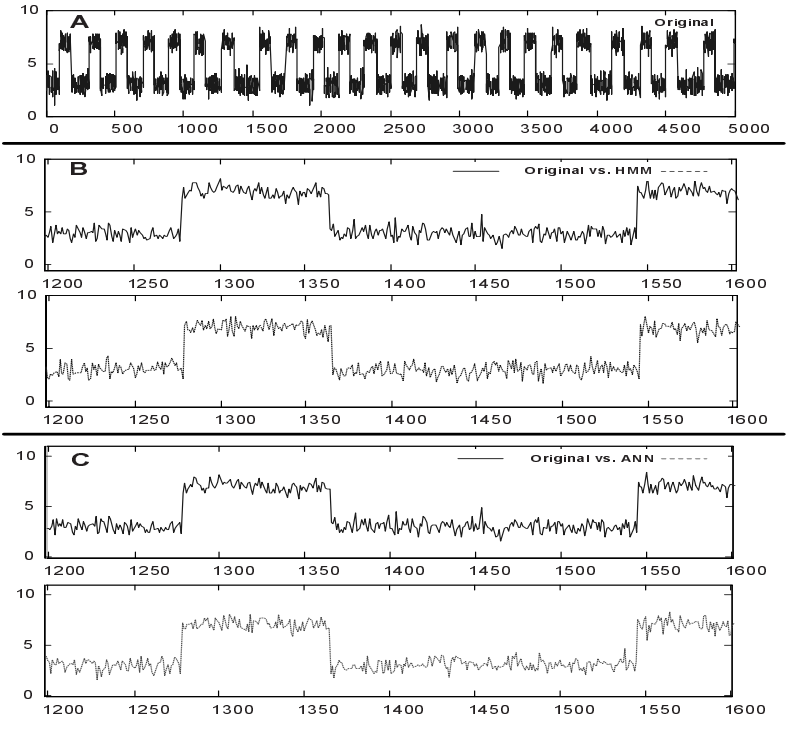

Fig. 3. Predicted Reputation Comparisons

to predict the reputation fairly accurately. However, the predicted values for HMM are closer to the original values in comparison with the ANN. Therefore, HMMs get our preference over ANNs. Moreover, the strong mathematical basis of HMMs is also a plus, which the ANNs lack. Since there is some cost associated with each of the above mentioned prediction models, either of these is of help to the service consumer only if the accuracy obtained through reputation prediction is more than the reputation values calculated in an ad hoc manner.

To capture the effects of reputation assessment when no prediction is involved, we have performed the experiments on the same data set of original reputations. Figure 4, shows the result of 1000 interactions out of a total 6000 interactions. The first 1449 interactions are those in which rater feedbacks are present. The 1450th. interaction onwards, no peer feedback is available about the service provider. Since no prediction is involved here, the future reputation evaluations hover around the reputation value that was observed at the last time instance. Since service consumer's *own* experience is also not factored in the reputation computation, we see an almost stationary reputation graph.

Figure 5 shows the effects of incorporating the service consumer's personal experience in calculating the provider reputation, when no peer feedbacks are available and no prediction is used. Around 600 iterations are shown for this case

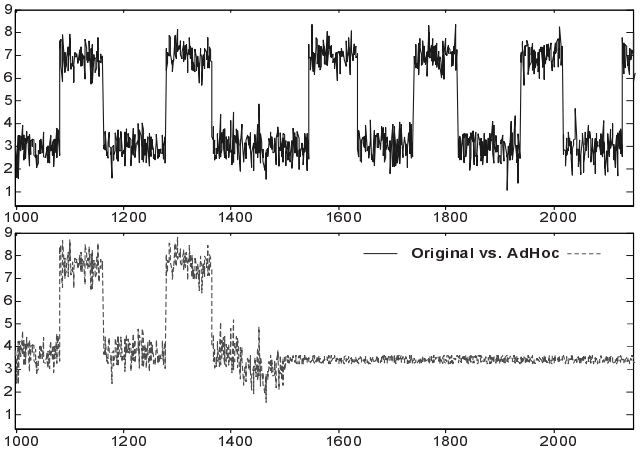

Fig. 4. Reputation Evaluation without Prediction and Personal Experience

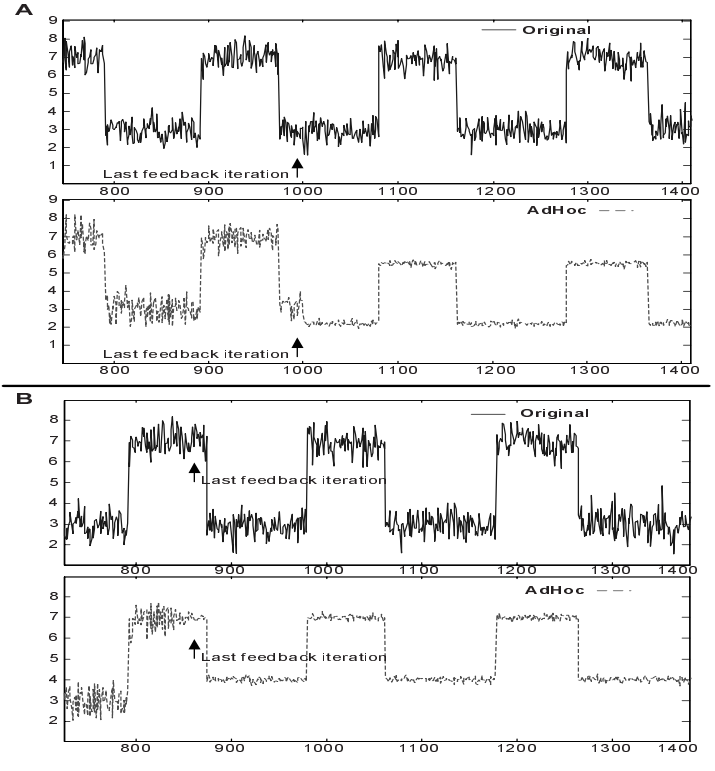

Fig. 5. Reputation Evaluation incorporating Personal Experience without Prediction

out of a total of 6000 iterations. In Figure 5-A, the last set of peer feedbacks is received at the 980th. iteration, which gives an overall *low* value (around 3) for the provider's reputation. Starting from the 981st. interaction, the service consumer incorporates his own personal experience into the last calculated aggregate reputation to reassess the provider's reputation. Since, the consumer's own testimony is weighed in highly, therefore the "general trend" of the evaluated reputation moves towards the original. However, since majority feedbacks are still centered around 3, an accurate assessment is not possible and the values are off by some degrees. Figure 5-B, provides a similar picture, but in this case the last feedbacks receiving iteration (960), leaves the service consumer with an aggregate reputation of 7. Subsequent reputation evaluations using the personal experience provide accurate results when the *actual* provider performance is high but inaccurate results when the actual provider performance is low. The reason is similar to the previous case, that since majority of the ratings are around 7, if the service consumer evaluates the provider as a low performer, the general trend of reputation evaluation moves in that direction but the high majority rating keeps the assessment inaccurate.

In light of the above experiments, we conclude that using a prediction model, we can assess the reputation of a service provider fairly accurately even if no rater feedbacks are present. We have also seen that HMMs have a slight edge over ANNs in computing service reputations. In contrast, if no prediction model is used then the reputation values that are assessed are inaccurate.

4 Conclusion and Future Work

We have presented an HMM-based reputation management framework to establish trust among Web services in situations where rater feedbacks may not be readily available. We have provided evaluation results for reputation prediction based on past service provider behavior using both an ANN and an HMM. In the future, we intend to build upon our proposed reputation management framework. We would refine the service interaction model to define a reputation model for *composed* Web services. Similarly, information dissemination techniques, change detection and interpretation for both individual and composed services will also be studied.

References

1. Buchegger, S., Le Boudec, J.-Y.: Performance Analysis of the CONFIDANT Protocol. In: Proc. of the 3rd ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing, June 9-11, pp. 226–236 (2002)
2. Cappe, O., Moulines, E., Ryden, T.: Inference in Hidden Markov Models. Springer, Heidelberg (2005)

3. Delgado, J., Ishii, N.: Memory-Based Weighted-Majority Prediction for Recommender Systems. In: ACM SIGIR 1999 Workshop on Recommender Systems: Algorithms and Evaluation (1999)
4. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: Certified reputation: how an agent can trust a stranger. In: AAMAS 2006: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 1217–1224. ACM Press, New York (2006)
5. Josang, A., Ismail, R.: The beta reputation system. In: 15th Bled Conference on Electronic Commerce (June 2002)
6. Josang, A.: A logic for uncertain probabilities. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 9(3), 279–311 (2001)
7. MacDonald, I.L., Zucchini, W.: Hidden Markov and Other Models for Discrete-valued Time Series. Chapman and Hall, Boca Raton (1997)
8. Malik, Z., Bouguettaya, A.: Rater Credibility Assessment in Web Services Interactions. World Wide Web Journal 12(1) (March 2009)
9. Malik, Z., Bouguettaya, A.: Reputation-based Trust Management for Service-Oriented Environments. VLDB Journal 18(4) (August 2009)
10. Malik, Z., Bouguettaya, A.: Reputation Bootstrapping for Trust Establishment among Web Services. IEEE Internet Computing 13(1) (January-February 2009)
11. Marti, S., Garcia-Molina, H.: Limited Reputation Sharing in P2P Systems. In: Proc. of the 5th ACM Conference on Electronic Commerce, New York, NY, USA, May 2004, pp. 91–101 (2004)
12. Medjahed, B., Bouguettaya, A.: Customized delivery of e-government web services. IEEE Intelligent Systems 20(6) (November/December 2005)
13. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, January 2002, pp. 2431–2439 (2002)
14. Papazoglou, M.P., Georgakopoulos, D.: Serive-Oriented Computing. Communcications of the ACM 46(10), 25–65 (2003)
15. Park, S., Liu, L., Pu, C., Srivatsa, M., Zhang, J.: Resilient trust management for web service integration. In: ICWS 2005: Proceedings of the IEEE International Conference on Web Services (ICWS 2005), Washington, DC, USA, pp. 499–506. IEEE Computer Society, Los Alamitos (2005)
16. Rabiner, L.R., Juang, B.H.: An introduction to hidden markov models. IEEE ASSP Magazine 3(1), 4–16 (1986)
17. Ran, S.: A model for web services discovery with qos. SIGecom Exch. 4(1), 1–10 (2003)
18. Sonnek, J.D., Weissman, J.B.: A quantitative comparison of reputation systems in the grid. In: The 6th IEEE/ACM International Workshop on Grid Computing, November 2005, pp. 242–249 (2005)
19. Tennenholtz, M.: Reputation systems: An axiomatic approach. In: AUAI 2004: Proceedings of the 20th conference on Uncertainty in artificial intelligence, Arlington, Virginia, United States, pp. 544–551. AUAI Press (2004)
20. Tian, M., Gramm, A., Ritter, H., Schiller, J.: Efficient selection and monitoring of qos-aware web services with the ws-qos framework. In: International Conference on Web Intelligence, Washington, DC, USA, pp. 152–158. IEEE Computer Society, Los Alamitos (2004)

21. Turin, W.: Digital Transmission Systems: Performance Analysis and Modeling. McGraw-Hill, New York (1998)
22. Udupi, Y.B., Singh, M.P.: Information sharing among autonomous agents in referral networks systems. In: 6th International Workshop on Agents and Peer-to-Peer Computing (May 2007)
23. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. IEEE Trans. on Knowledge and Data Engineering (TKDE) 16(7), 843–857 (2004)
24. Yu, B., Singh, M.P.: An evidential model of distributed reputation management. In: AAMAS 2002: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pp. 294–301. ACM Press, New York (2002)