

Research on running time behavior analyzing and trend predicting of modern distributed software

Junfeng Man

College of Computer and Communication, Hunan University of Technology, Zhuzhou, China
 College of Information Science and Technology, Central South University, Changsha, China
 Email: mjfok@tom.com

Zhicheng Wen, Changyun Li, Xiangbing Wen

College of Computer and Communication, Hunan University of Technology, Zhuzhou, China
 Email: zcwen@mail.shu.edu.cn, lcy469@163.com

Abstract—Interactive behavior trust of modern distributed software systems (MDSS) should be “monitored” and “grasped” at running time. The paper investigates the relationships between behaviors and their effects at running time in MDSS, uses statistical machine learning tools to analyze the laws of behavior traces, and presents a novel behavior analyzing and trend predicting method. We use hierarchical Dirichlet process and infinite hidden Markov model to converge monitored interface data to determine unknown events, and learn behavior patterns from event sequences including unknown events in terms of semi-supervised method. As determining unknown events and behavior patterns, Beam sampling has higher efficiency in sampling and inference compared with other method (e.g., Gibbs sampling). When behavior patterns reach a certain scale, MDSS can analyze and predict interactive behaviors in terms of unsupervised method. We adopt Viterbi algorithm of hidden Markov model to analyze optimal sequences of interactive events, which help to determine good and evil of current behaviors. MDSS can send early warning for hostile behaviors, actively predict subsequent trends for non-hostile behaviors. Simulation experiments testify that the novel method has unique predominance in software behavior analyzing and trend predicting.

Index Terms—modern distributed software systems, behavior trust, behavior analyzing, trend predicting, infinite hidden Markov model

I. INTRODUCTION

Modern Distributed Software Systems (MDSS) have different characteristics from conventional ones in that it is open and dynamic, loosely-coupled and behavior-complicated. From view of architecture, MDSS is integrated with many software entities (e.g., subsystems, soft components, agents or web services) scattered in network environment in terms of cooperation; from view of lifecycle, MDSS can implement dynamic evolution with the change of environments and requirements, which is showed in the change of entity elements and adjustment of architectural relationships, entity elements may join or leave MDSS at running time; from view of software behaviors, the complexity comes from the

heterogeneity of entity elements, collaboration relationships of elements, variety of system intentions and management policies.

In such open, dynamic and complicated MDSS, judging interactive entity trust by identity trust is insufficient. For example, college students may login digital resource servers ordered by universities by identity trust (IP address of universities), however, their behaviors may be not trusted. Some college students use tools to download large quantities of digital resources or privately set proxy servers to gain lawless earnings. Above instance illustrates that user identities are trusted but their behaviors are not always trusted.

Paper [1] defines the concept of software behavior trust that interactive behaviors and results of entity elements can be predicted and controlled at running time, namely, behavior states can be monitored, behavior results can be evaluated and exceptional behaviors can be controlled. Entity trust consists of identity trust and behavior trust. Identity trust is that the identity of interactive entities can be accurately judged, not be imitated, namely, the identity of entities is real and available. Behavior trust of entities is whether their behaviors can be predicted, evaluated and managed or not, whether interactive entities bring deceitful or fraudulent behaviors and destroy system data or not.

The nature of software is to substitute for people to carry out certain behaviors. It is especially important for MDSS to analyze interactive behaviors and effects of entity elements, and actively predict subsequent possible behavior trends according to analysis outcomes, which can effectively avoid deceitful or fraudulent behaviors in the course of interaction, and defend untrusted behaviors in advance in case of serious outcome.

Though distributed software has been applied almost 30 years, interactive behavior analyzing and predicting are still faced with many challenges in MDSS. Existing researches on software behavior mostly pay attention to outcome trust of interactive behaviors, or apply behavior trust to conventional distributed software. These technologies are difficult to adapt to MDSS, main reason

is that loosely-coupled entities in MDSS may be strange for each other, or the entities which have ever interacted each other may have new interaction in new situation. In such complicated environment, there is an unsettled problem how MDSS discern the nature of behavior patterns by analyzing interactive behaviors of entity elements and predict subsequently possible trends according to analysis outcome.

Because loosely-coupled entities in MDSS have their own profits, behaviors and rules, their running time behaviors have inherent laws, the collaboration of interactive entities makes them show some statistical characteristic in the mass at running time. MDSS should be "monitored" and "grasped" in open and dynamic environment, the intentions, situations and relationships between behavior and behavioral effects at running time are investigated. Firstly, Statistical Machine Learning (SML) technologies are adopted to analyze the laws of behavior traces and predict behavior trends; secondly, behavior predicting model is constructed, finally, the intentions of interactive entities are inferred, and future behavior trends are actively predicted.

The rest of this paper is organized as follows: Firstly, on the basis of analyzing related researches, a novel behavior analyzing and trend predicting method is presented; Secondly, SML technologies are adopted to learn unknown events and behavior patterns, Viterbi algorithm is adopted to determine good and evil of current behaviors, then subsequent possible trends are actively predicted; Finally, simulation experiments take digital resource services of campus for instance, which illustrates unique predominance of this novel method.

II. RELATED RESEARCH

Many researchers have had beneficial exploration in software predicting. Some people predict software subsequent behaviors by referring historical behaviors. Based on past behavior patterns, Nielsen et al. computed maximum expectation of future software behaviors [2]. Mello et al. used neural network to predict the behaviors of application [3]. Bouguila et al. used statistical method of Bayesian network to predict application accessing contents [4]. Using stochastic process and artificial intelligence, Dodonov presented a model for behavior predicting of applications. Above predicting methods have definite restriction, do not adapt to open, dynamic and complicated distributed software environment.

In open, dynamic and complex environment, interactive objects may be strange entities. The problem of how to predict their behavior trust is preliminarily explored by some researchers. Haller et al. tried to solve the trust problems of strange entity interaction in open and dynamic Internet environment [5]. Teacy et al. presented TRAVOS-C system and tried to solve trust problems of strange agent interaction [6]. Using nonparametric Bayesian model, Shi presented situation-sensitive method to model and learn multidimensional trust parameters, and applied the method to construct strange entity trust without pervious interactive history [7]. The methods of Haller and Teacy have not the

capacity of automatically and intuitively learning trust parameters. The method of Shi may simply predict future interactive events, but does not consider the prediction of interactive behaviors consisting of event sequence.

Lu et al. discussed how to capture buyers' intentions of acquiring information and buying goods, and used these intentions to help buyers and sellers to make decision [8]. Oyama et al. inferred intentions of human in pervasive computing which was convenient for system evolvement [9]. These researches mostly investigate the prediction of user intentions, and do not come down to the problems of behavior policies, intention analyzing and subsequent possible trend predicting of interactive entity elements.

Most of above models or methods adapt to conventional distributed software system. Though a few researches discuss the problems of software trust in MDSS, they do not present perfect solution to running time behavior analyzing and trend predicting. New software environments are faced with new problems, and a novel method should be presented to solve them.

III. THE MODEL OF BEHAVIOR ANALYZING AND TREND PREDICTING

A. Three key problems to be solved

1) Behavior analyzing and trend predicting should be "grasped" at running time. Based on real-time self-experience, interactive entity trust can be automatically computed, which helps to judge their trust in terms of real-time and objective methods. We present situation-sensitive method to model and learn multidimensional trust parameters, pervious similar situation can be used to currently interactive entities, which very adapt to solve the problems of behavior analyzing and trend predicting for strange entities.

2) Researching on predicting model of MDSS, field-independent and field-dependent hierarchical repository should be constructed. The intentions of entity elements in MDSS determine their behavior fashion. In concretely interactive situation, collaborative interactions and outcomes among these elements are related to behavior effects. The relationships among intentions, situations, behaviors and behavior effects are studied, field-independent and field-dependent hierarchical repository are constructed. On this basis, predicting model of MDSS is constructed, which provides powerful support for running time behavior analyzing and trend predicting.

3) Researching on the method of behavior analyzing and trend predicting of MDSS, SML tools are used to improve the efficiency. By analyzing laws of behavior traces, software can infer behavior intentions. Referring to repository, system can predict future behavior trends. SML tools make behavior analyzing and trend predicting more adaptable, active and smart.

B. Model of Behavior Analyzing and Trend Predicting

In MDSS, various entities are autonomous and potentially subject to different administrative and legal domains. In my opinion, interactive behaviors of entities behave like a stochastic process. With isolated insight,

the outcome of an event of interest cannot be predicted exactly. From the view of statistics, the probability that the outcome for the next execution of the event will be a particular point within the space of possible outcomes is described by a probability distribution. This space is the outcome distribution of the trusted entity, whose granularity determines precision of behavior predicting. Thus, we adopt statistical idea to collect information of interactive behaviors and their effects, which are saved into repository along with the knowledge of domain experts. When the rules and knowledge of repository reach a certain scale, MDSS can analyze and predict interactive behaviors in terms of unsupervised method.

The model of behavior analyzing and trend predicting is showed in Fig. 1. For running time interactive entities (1), monitors are adopted to monitor interface data of interactive events; these monitored raw data are preprocessed by filtering, dimensionality reduction and

normalization (3), then trust parameters that are used to compute interactive behavior are acquired. These trust parameters are added semantic tags according to events ontologies; Hierarchical Dirichlet Process (HDP) and infinite hidden Markov model (iHMM) are adopted to converge trust parameters to determine whether unknown events are produced or not (4). If so, event sequences including unknown events are learned to produce new behavior patterns in terms of semi-supervised method (6, 7), learning outcomes are added into rules and knowledge repository (5) by managers; if not, interactive event sequences are analyzed (8), which helps to determine the types of current behavior patterns and predict subsequently possible behavior trends; Effective predicting of interactive behaviors provides solid foundation for software hazard analyzing (9); the outcomes of hazard analyzing can effectively instruct subsequent running of software entities.

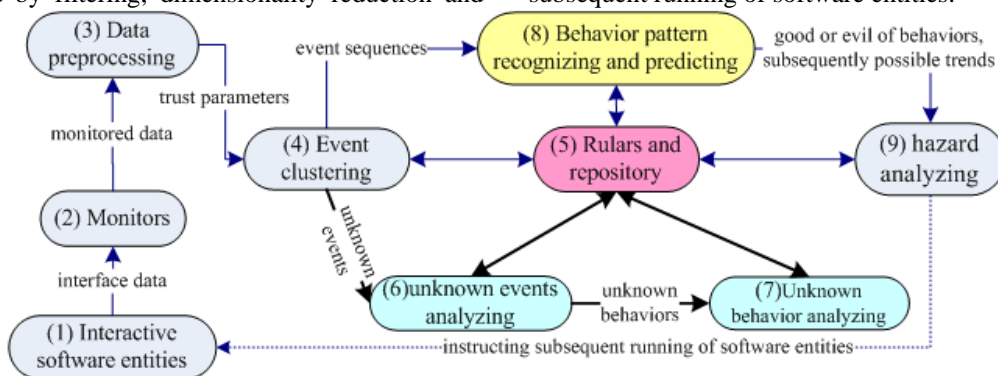


Figure 1. The model of behavior analyzing and trend predicting

The active and smart characteristics of this model are showed that it makes full use of SML tools to analyze interactive behaviors and predict subsequently possible trends. The model can effectively identify unknown events and behavior patterns. When the rules and repository reach a certain scale, the model can analyze and predict interactive behaviors in terms of unsupervised method. The Viterbi algorithm of Hidden Markov Model (HMM) is adopted to analyze optimal sequence of interactive events, which helps to determine good and evil of current behavior patterns. The method can predict behavior in advance before vicious users do any damages, which helps us to avoid deceitful or fraudulent behaviors.

In some situation, the parameters used to estimate interactive events cannot be acquired entirely, HDP may help to estimate other parameters conditioned on inadequate parameters, clusters that interactive event belongs to are determined by computing HDP posterior, which improves the adaptability of behavior analyzing.

1) Running Time Monitor

Monitoring is the base of running time predicting. At present, monitoring technologies of MDSS are very mature, we will make full use of them. Because MDSS have enormous scales and complicated behaviors, their running time behaviors cannot be completely monitored, and there is no need to monitor all them. The entity elements of MDSS may come from third party, namely

black box entities, which have open functions and hide implementation, whose inner structures and behaviors cannot be observed directly. Thus, mostly monitored data are interface data of interactive behaviors.

2) Rules and Repository

Predicting rules and repository is a field-independent and field-dependent hierarchical repository. By studying the relationships among intentions, situations, behaviors and behavior effects in general or concrete application, above contents are sorted and stored into initial repository by human. Using SML tools, undefined predicting rules are added into repository by managers in terms of semi-supervised method. For inexistent behavior patterns in repository, MDSS can automatically analyze and compute their behaviors and corresponding effects, and add them to repository. As the contents of repository become very, filtering, analyzing and predicting capacity of MDSS also becomes more powerful.

IV. BEHAVIOR RECOGNIZING AND TREND PREDICTING

A. The learning of unknown events and behavior patterns

Loosely-coupled entities may produce unknown events and behavior patterns in the course of interaction. How to identify and train them is a difficult problem in MDSS. For this problem, to the best of our knowledge, there is no very good solution.

The conventional inference methods for HMMs are the expectation-maximization (EM) method implemented via the Baum-Welch algorithm [10] and the variational Bayesian method [11]. However, in both methods the model structure must be specified initially, i.e. the number of states (events) is fixed. Knowing the correct model complexity requires expensive model selection and in some applications there may exist no such fixed “correct” model. We address this problem by using an HMM with a countably infinite state space, namely, the iHMM. Beal et al. firstly proposed the iHMM and provided an approximate sampling scheme for inference [11]. Teh et al. demonstrated that HDP could be used to recast the iHMM and provided a useful sampling scheme [12]. Jurgen et al. used Beam sampling and inferring method to improve computing efficiency of iHMM [13].

The diagram of identifying and learning unknown events is showed in Fig. 2. When unknown events are found in the course of converging trust parameters with HDP, Beam algorithm of iHMM is used to learn unknown events and behavior patterns. Qualitative events and behavior patterns are added into rules and repository. These behavior patterns are ready for training.

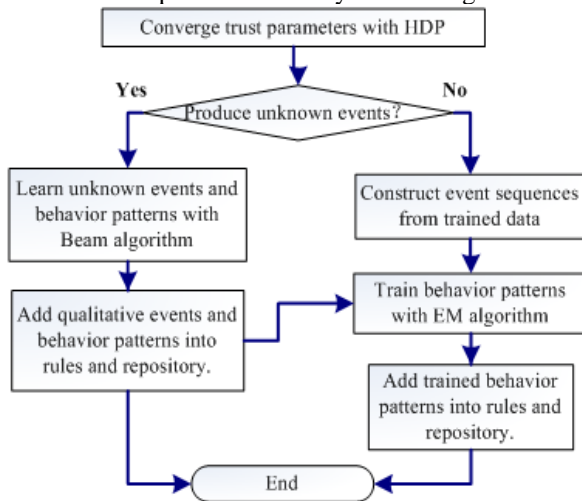


Figure 2. Training process of events and behavior patterns

1) HDP-iHMM

We complete the Bayesian description by specifying the priors. Let the observation parameters ϕ be iid (independent and identically distributed) drawn from a prior distribution H . With no further prior knowledge on the state sequence, the typical prior for the transition (and initial) probabilities are symmetric Dirichlet distributions.

A simple way to obtain a nonparametric HMM with an infinite number of states might be to use symmetric Dirichlet priors over the transition probabilities with parameter α/K and take $k \rightarrow \infty$. Such an approach has been successfully used to derive Dirichlet Process (DP) mixture models, but unfortunately does not work in the HMM context. The subtle reason is that there is no coupling across transitions out of different states since the transition probabilities are given independent priors [11]. To introduce coupling across transitions, one may use a hierarchical Bayesian formalism where the Dirichlet

priors have shared parameters and given a higher level prior, e.g.

$$\pi_k \sim Dirichlet(\alpha\beta), \beta \sim Dirichlet(\gamma/K, \dots, \gamma/K) \quad (1)$$

where π_k is transition probabilities out of state k and β is the shared prior parameters. As $k \rightarrow \infty$, the hierarchical prior (1) approaches a HDP [12].

A HDP is a set of DPs coupled through a shared random base measure which is itself drawn from a DP [12]. Specifically, each $G_k \sim DP(\alpha, G_0)$ with shared base measure G_0 , which can be understood as the mean of G_k , and concentration parameter $\alpha > 0$, which governs variability around G_0 , with small α implying greater variability. The shared base measure is itself given a DP prior: $G_0 \sim DP(\gamma, H)$ with H a global base measure. The stick-breaking construction for HDPs shows that the random measures can be expressed as follows: $G_0 = \sum_{k=1}^{\infty} \beta_k \delta_{\phi_k}$ and $G_k = \sum_{k=1}^{\infty} \pi_{kk} \delta_{\phi_k}$, where $\beta \sim GEM(\gamma)$ is the stick-breaking construction for DPs, $\pi_k \sim DP(\alpha, \beta)$, and each $\phi_k \sim H$ independently.

Identifying each G_k as describing both the transition probabilities π_{kk} from state k to k' and the emission distributions parametrized by ϕ_k , we can now formally define the iHMM as follows:

$$\beta \sim GEM(\gamma), \pi_k | \beta \sim DP(\alpha, \beta), \phi_k \sim H \quad (2)$$

$$s_t | s_{t-1} \sim Multinomial(\pi_{s_{t-1}}), o_t | s_t \sim F(\phi_{s_t}) \quad (3)$$

The graphical model corresponding to this hierarchical model is shown in Fig. 3. Thus β_k is the prior mean for transition probabilities leading into state k' , and α governs the variability around the prior mean. If we $\beta = (1/k, \dots, 1/k, 0, 0, \dots)$ where the first K entries are $1/k$ and the remaining are 0, then transition probabilities into state k' will be non-zero only if $k' \in \{1, \dots, K\}$.

Finally we place priors over the hyperparameters α and γ . A common solution, when we do not have strong beliefs about the hyperparameters, is to use gamma hyperpriors: $\alpha \sim Gamma(a_\alpha, b_\alpha)$ and $\gamma \sim Gamma(a_\gamma, b_\gamma)$. Teh et al. describe how these hyperparameters can be sampled efficiently [12], we will use this in the experiments to follow.

2) Beam Sampling and Inference algorithm

The forward-backward algorithm does not apply to the iHMM because the number of states, and hence the number of potential state trajectories, are infinite. The idea of beam sampling is to introduce auxiliary variables u such that conditioned on u the number of trajectories with positive probability is finite. Now dynamic programming can be used to compute the conditional probabilities of each of these trajectories and thus sample whole trajectories efficiently. These auxiliary variables

do not change the marginal distribution over other variables hence MCMC (Markov chain Monte Carlo) sampling will converge to the true posterior.

As opposed to the sampler in the previous section, the beam sampler does not marginalize out π nor ϕ .

Specifically, the beam sampler iteratively samples auxiliary variables u , trajectory s , transition probabilities π , shared DP parameters β and hyperparameters α and γ conditioned on all other variables.

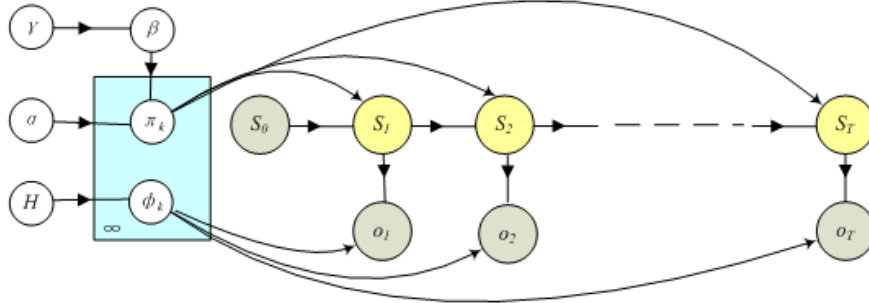


Figure 3. The graphical model of iHMM

Sampling u : for each t we introduce an auxiliary variable u_t with conditional distribution $u_t = Uniform(0, \pi_{s_{t-1}s_t})$ depending on π , s_{t-1} and s_t .

Sampling s : we sample whole trajectory s given auxiliary variables u and other variables using a form of forward filtering-backward sampling. The important observation here is that only trajectories s with $\pi_{s_{t-1}s_t} \geq u_t$ for all t will have non-zero probability given u .

There are only finitely many such trajectories and as a result we can compute conditional distribution over all such trajectories efficiently using dynamic programming.

First note that the probability density for u_t is $p(u_t | s_{t-1}, s_t, \pi) = \frac{I(0 < u_t < \pi_{s_{t-1}s_t})}{\pi_{s_{t-1}s_t}}$, where $I(C) = 1$ if condition C is true and 0 otherwise. We compute $p(s_t | o_{1:t}, u_{1:t})$ for all t as follows (we omitted the additional conditioning variables π and ϕ for clarity):

$$\begin{aligned} p(s_t | o_{1:t}, u_{1:t}) &\propto p(s_t, u_t, o_t | o_{1:t-1}, u_{1:t-1}) \\ &= \sum_{s_{t-1}} p(o_t | s_t) p(u_t | s_t, s_{t-1}) p(s_t | s_{t-1}) \\ &= p(o_t | s_t) \sum_{s_{t-1}}^{p(s_{t-1} | o_{1:t-1}, u_{1:t-1})} \mathbb{I}(u_t < \pi_{s_{t-1}s_t}) p(s_{t-1} | o_{1:t-1}, u_{1:t-1}) \quad (4) \\ &= p(o_t | s_t) \sum_{s_{t-1}, u_t < \pi_{s_{t-1}s_t}} p(s_{t-1} | o_{1:t-1}, u_{1:t-1}) \end{aligned}$$

Note that we only need to compute (4) for the finitely many s_t values belonging to some trajectory with positive probability. Further, although the sum over s_{t-1} is technically a sum over an infinite number of terms, auxiliary variable u_t truncates this summation to the finitely many s_{t-1} 's that satisfy both constraints $\pi_{s_{t-1}s_t} \geq u_t$ and $p(s_{t-1} | o_{1:t-1}, u_{1:t-1}) > 0$. Finally, to sample the whole trajectory s , we sample s_T from $p(s_T | o_{1:T}, u_{1:T})$ and perform a backward pass where we sample s_t given the sample for s_{t+1} : $p(s_t | s_{t+1}, o_{1:T}, u_{1:T}) \propto p(s_t | o_{1:t}, u_{1:t}) p(s_{t+1} | s_t, u_{t+1})$.

Sampling π , ϕ and β : about these sampling, we may refer the theory of HDPs [12].

Finally, each ϕ_k is independent of others conditional on s , o and their prior distribution H , i.e. $p(\phi | s, o, H) = \prod_k p(\phi_k | s, o, H)$. When the base distribution H is conjugate to the data distribution F each ϕ_k can be sampled efficiently.

B. The training for behavior patterns with with definite event number

By now then, for the problems of parameters selection and optimization of HMM, usual method is EM method, which is a typically iterative algorithm. At initial stage, experiential estimation values are set by managers, and parameters tend to more reasonable values by many times iteration. The training process of behavior patterns with definite event number is showed in Fig. 2. Firstly, event sequences are constructed from training data, then event sequences are trained with EM method, finally, trained behavior patterns are added to repository.

C. The analyzing and predicting of behavior patterns

The diagram of behavior analyzing and trend predicting is showed in Fig. 4. The Viterbi algorithm is adopted to analyze optimal sequence of interactive events, which help to determine good and evil of current behaviors. Instant alarm is sent for hostile behavior; subsequent events are actively predicted for non-hostile behavior, if subsequent events make current behavior change from good to evil, early warning is sent. Viterbi algorithm shows outstanding capability in behavior analyzing and subsequently possible trend predicting. We will analyze its performance in Section 5.

V. SYSTEM SIMULATION EXPERIMENTS

For simulation experiments, we take digital resource services of campus for instance. It is assumed there are four monitored events (i.e., Login, LookupPaper, DownloadPaper and Logout) in the system. Now, the online service of ordering paper is started, so two additional events (i.e., OrderPaper and OnlinePay) are

added. HDP-iHMM is used to show how two new events are recognized. The state transition diagram with six events is showed in Fig. 5.

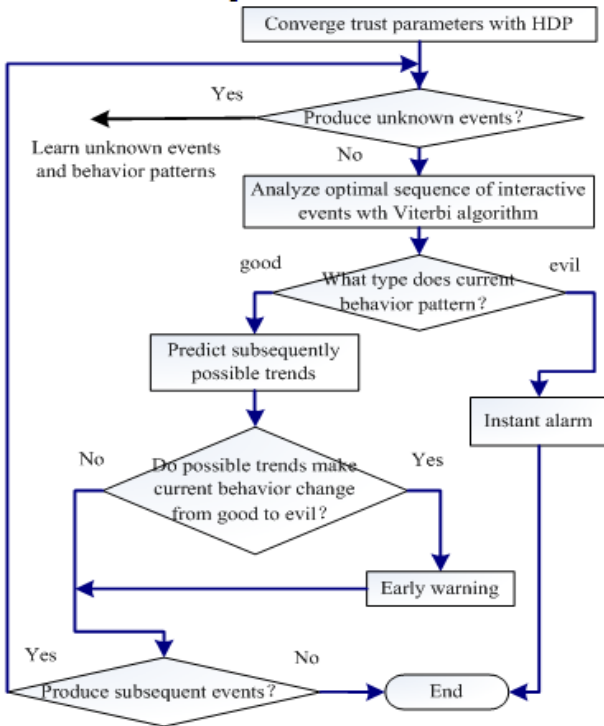


Figure 4. The process of behavior analyzing and trend predicting

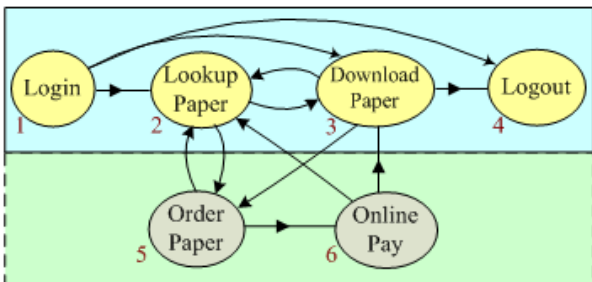


Figure 5. The state transition diagram

A. The simulation for learning unknown events

We separately use Beam sampling and Gibbs sampling algorithm to train the same input parameters information, which helps to determine the number of interactive events. In the course of training, we execute many times iteration until Joint Log Likelihood (*JLL*) reach setting threshold. Here, *K* value is congregated at 5, which show that HDP-iHMM can accurately recognize unknown events. The training process of recognizing unknown events is showed in Fig. 6. From Fig. 6, we distinctly see that Beam algorithm surpasses Gibbs algorithm in training efficiency. In this experiment, the number of initial state is 4, and the number of observation is 8. By the training with iHMM, the number of state becomes 5. In this course, *K* and *JLL* have very small fluctuation and drive to stabilization with Beam algorithm, however, *K* and *JLL* have very large fluctuation and do not converge to stable value with Gibbs algorithm.

Firstly, four initial states (1, 2, 3, 4) are set as parent population, samples are drawn for training HDP-iHMM model, this model is trained and defined as normal behavior pattern. At the same time, samples are converged to get cluster diagram (left one of top row in Fig. 7), model state-transition matrix (left one of second row) and model observation-transition matrix (left one of last row).

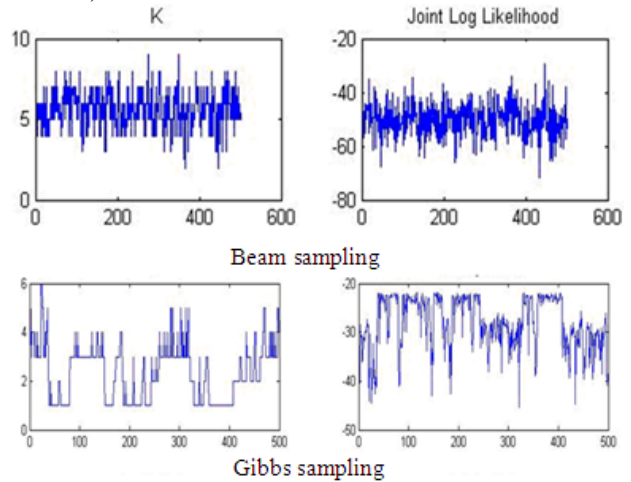


Figure 6. Training process of K and JLL

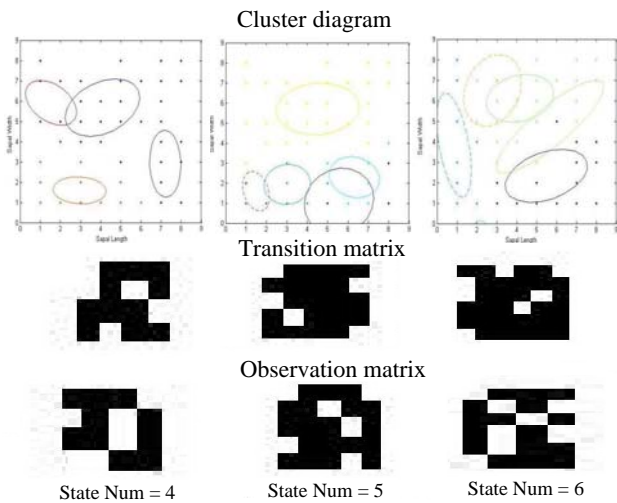


Figure 7. The course of detecting exceptional behaviors: cluster diagram (top row), model state-transition matrix (second row), model observation-transition matrix (last row).

Secondly, a detecting sample is tested with well-trained model to validate the effectiveness of Beam algorithm. When detecting the sample with state five, an exceptional behavior is found, the system will prompted to update repository. New event (state) and behavior pattern consisting of this event are added into repository. The model corresponding to this behavior pattern is trained with definite samples, training outcome is saved into repository. At the same time, samples are converged to get cluster diagram (middle one of top row), model state-transition matrix (middle one of second row) and model observation-transition matrix (middle one of last row). From Fig. 7, we see that a new state is added, namely from four to five.

Lastly, when detecting the sample with state five and six, an exceptional behavior is also found. Repeating above process, we also get cluster diagram (last one of top row), model state-transition matrix (last one of second row) and model observation-transition matrix (last one of last row). From Fig. 7, we see that a new state is added again, namely from five to six.

Above simulation experiment shows the performance of HDP-iHMM in recognizing unknown events. Unknown events and behavior patterns can be determined with iHMM, which is ready for behavior patterns learning, recognizing and predicting.

B. The simulation for learning behavior patterns

After behavior patterns are determined, HMM is adopted to implement behavior learning, recognizing and predicting. Behavior patterns of this simulation experiments are divided into four types: security, middle-security, poor-security and insecurity, which are represented with P1, P2, P3 and P4 separately. Twenty samples are selected in every type to learn, recognize and predict behavior patterns. Part of training samples of behavior patterns is showed in table 1.

TABLE I. TRAINING SAMPLES OF BEHAVIOR PATTERNS

Types of behavior	Event sequence	No
security (P1)	1 2 5 6 3 4 2 5 6 3 4 2 5 2 5 6 3	a1
middle-security (P2)	1 2 5 2 5 2 5 3 3 3 5 2 2 5 4 2 2	b1
poor- security (P3)	1 1 1 5 6 5 6 4 4 4 3 3 3 3 5 4 4	c1
insecurity (P4)	1 1 1 4 4 4 5 5 5 1 1 4 4 5 5 1 1	d1

It is assumed that initial state-transition matrix and observation-transition matrix are even. Initial probability distribution vector PI is [1 0 0 0]. In training process, with the increase of iteration, maximum likelihood estimation Log is also continuously increasing until it reaches to setting threshold. The diagram of all kinds of trained behavior types is showed in Fig. 8.

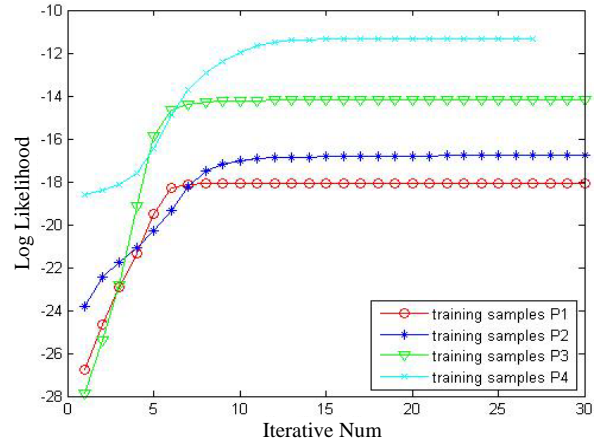


Figure 8. The diagram of all kinds of trained behavior types

C. The simulation for recognizing and predicting behavior types

The Viterbi algorithm is adopted to analyze recognizing capacity of our model for trained samples. Above four types of training samples is used to testify this kind of capacity. Statistical outcomes for recognizing four trained samples (a1, b1, c1 and d1) are showed in table 2. The first nine of trained samples (a1, b1, c1 and d1) are also selected as samples, corresponding statistical outcomes are showed in table 3.

Above twice experiments show that our model has very high recognizing capacity for events sequences conditioned on multi-training samples (namely, multi-behavior patterns, here are 20). In the same way, different samples acquired are recognized with above model, iHMM_HMM always shows very good performance.

Lastly, we testify behavior predicting capacity of iHMM_HMM. For digital resource services of campus, transition probability between behavior types is acquired by managers after a considerable length of time, corresponding experiential matrix is showed in table 4.

TABLE II. RECOGNIZING EFFICIENCY OF BEHAVIOR PATTERNS (1)

Observation No.	P1	P2	P3	P4	Current type	Recognizing probability(%)	Recognizing outcome
a1	0.62868	0.14802	0.18722	0.03606	P1	62.87	TRUE
b1	0.11852	0.54745	0.21962	0.11439	P2	54.74	TRUE
c1	0	0	0.93570	0.06429	P3	93.57	TRUE
d1	0	0	0.20263	0.79736	P4	79.74	TRUE

TABLE III. RECOGNIZING EFFICIENCY OF BEHAVIOR PATTERNS (2)

Observation No.	P1	P2	P3	P4	Current type	Recognizing probability(%)	Recognizing outcome
a1_1	0.63031	0.16099	0.16993	0.03875	P1	63.03	TRUE
b1_1	0.22055	0.61095	0.12916	0.03931	P2	61.10	TRUE
c1_1	0	0.16883	0.74009	0.09106	P3	74.01	TRUE
d1_1	0	0	0.10080	0.89919	P4	89.92	TRUE

TABLE IV. TRANSITION MATRIX OF BEHAVIOR TYPES

	P1	P2	P3	P4
P1	0.8	0.1	0.1	0
P2	0.4	0.2	0.3	0.1
P3	0.1	0.2	0.3	0.4
P4	0	0.1	0.3	0.3

Table 4 shows that the model may effectively predict subsequently possible behavior types on condition that current one is known. For example, if the model has recognized that current behavior type is middle-security, subsequently possible behavior type is security, whose expected probability is 40%. The (a1, b1, c1 and d1) are served as samples to predict subsequently possible behavior types, predicting outcomes are showed in table 5.

By recognizing current behavior patterns and types and predicting subsequently possible behavior types, the

model can determine current security level and predict future behavior trends.

TABLE V. THE PREDICTING OF BEHAVIOR TYPES

Observation No.	P1	P2	P3	P4	Current type	probability	Recognizing outcome
a1	0.62868	0.14802	0.18722	0.03606	P1	P1	80%
b1	0.11852	0.54745	0.21962	0.11439	P2	P1	40%
c1	0	0	0.93570	0.06429	P3	P4	40%
d1	0	0	0.20263	0.79736	P4	P4	60%

VI. CONCLUSION

We research on behavior trust at running time in open, dynamic and complicated MDSS. SML tools are adopted to solve the problems of behavior patterns learning and recognizing and trends predicting. HDP-iHMM is used to determine the number of interactive events, HMM is used to learn, recognize and predict behavior patterns. Effective integration of iHMM and HMM solves the technical difficulties of software behavior learning, recognizing and predicting on condition that interactive events and behavior patterns are unknown, which enhances adaptability of MDSS.

In subsequent researches, the efficiency of the model in behavior patterns learning, recognizing and predicting will be improved. Some prototypical systems such as online business system, campus digital resource services system are designed to test the performance of our model.

ACKNOWLEDGMENT

The financial supports from the national natural science fund of China under the grant No. 60773110, post-doctoral science fund of China under the grant No. 20080440216 and the education department fund of Hunan province under the grant No. 08C286 and 08C284 is gratefully acknowledged.

REFERENCES

- [1] C. Lin, L. Q. Tian and Y. Z. Wang, "Research on user behavior trust in trustworthy network", *Journal of Computer Research and Development*, vol. 12, No.12, Dec. 2008, pp. 2033-2043.
- [2] M. Nielsen and K. Krukow, "A Bayesian model for event-based trust", *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 172, No.4, 2007, pp. 499-521.
- [3] R. Mello, L. Senger, and L. Yang, "Automatic text classification using an artificial neural network", *High Performance Computational Science and Engineering*, vol. 17, No.9, 2005, pp. 1-21.
- [4] N. Bouguila, J. H. Wang and A. B. Hamza. "A Bayesian approach for software quality prediction. 2008 4th International IEEE Conference "Intelligent Systems", 2008, pp. 49-54.
- [5] J. Haller, "A stochastic approach for trust management", *Proceedings of the 22nd International Conference on Data Engineering Workshops*, 2006, pp. 19-27
- [6] W. T. L. Teacy, "Agent-based trust and reputation in the context of inaccurate information sources. PhD thesis, Electronics and Computer Science, University of Southampton, 2006.

- [7] J. Q. Shi, "A trust model with statistical foundation", MS. Thesis, Ottawa: University of Ottawa, 2005.
- [8] Y. B. Lu, L. Zhao and B. Wang, "Exploring factors affecting trust and purchase behavior in virtual communities", *Advanced Management of Information for Globalized Enterprises 2008*. Tianjin, China. 2008, pp. 1-5
- [9] K. Oyama, H. Jaygarl and J. Xia, "A human-machine dimensional inference ontology that weaves human intentions and requirements of context awareness systems", *The 32nd Annual IEEE International Computer Software and Applications Conference*. 2008. pp. 287-294
- [10] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, 1989, pp. 257-286.
- [11] M. J. Beal, Z. Ghahramani and C.E. Rasmussen, "The infinite hidden Markov model", In *Advances in Neural Information Processing Systems*, MIT Press, 2002, pp. 577-584
- [12] Y. W. Teh, M. I. Jordan and M. J. Beal, "Hierarchical dirichlet processes", *Journal of the American Statistical Association*, 2006, pp. 1566-1581.
- [13] V. G. Jurgen, Y. Saatici and Y. W. The, "Beam sampling for the infinite hidden Markov model", *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 102-109

Junfeng Man Born in Suihua, China, on January, 8, 1976. He received the master degree in Computer Software and Theory from Yanshan University in 2003. He is presently working on his PhD in College of Information Science and Technology of Central South University. He is associate professor in College of Computer and Communication of Hunan University of Technology. His research interests include trust software, pervasive computing. Contact: mjfok@tom.com.

Zhicheng Wen received the PhD degree in Computer Software and Theory from Shanghai University in 2007. He is associate professor in College of Computer and Communication of Hunan University of Technology. His research interests include trust software, software architecture.

Changyun Li received the PhD in Computer Software and Theory from Zhejiang University in 2006. He is professor in College of Computer and Communication of Hunan University of Technology. His research interests include trust software, software architecture.