Dynamic Syslog Mining for Network Failure Monitoring

Kenji Yamanishi
NEC Corporation
1753,Shimonumabe,Nakahara-ku,
Kawasaki,Kanagawa 216-8666,JAPAN
k-yamanishi@cw.jp.nec.com

Yuko Maruyama [*]
NEC Corporation
1753,Shimonumabe,Nakahara-ku,
Kawasaki,Kanagawa 216-8666,JAPAN
y-maruyama@bp.jp.nec.com

ABSTRACT

Syslog monitoring technologies have recently received vast attentions in the areas of network management and network monitoring. They are used to address a wide range of important issues including network failure symptom detection and event correlation discovery. Syslogs are intrinsically *dynamic* in the sense that they form a time series and that their behavior may change over time. This paper proposes a new methodology of *dynamic syslog mining* in order to detect failure symptoms with higher confidence and to discover sequential alarm patterns among computer devices. The key ideas of dynamic syslog mining are 1) to represent syslog behavior using a mixture of Hidden Markov Models, 2) to adaptively learn the model using an on-line discounting learning algorithm in combination with dynamic selection of the optimal number of mixture components, and 3) to give anomaly scores using universal test statistics with a dynamically optimized threshold. Using real syslog data we demonstrate the validity of our methodology in the scenarios of failure symptom detection, emerging pattern identification, and correlation discovery.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Learning

General Terms

Experimentation, Theory

Keywords

Syslog mining, Failure detection, Correlation analysis, Probabilistic modeling, Model selection

[*]The current address is
Nomura Securities, 2-2-2 Otemachi, Chiyoda-ku, Tokyo 100-8130, Japan. E-mail: y1-maruyama@frc.nomura.co.jp

1. INTRODUCTION

1.1 Problem Setting

Syslogs are a sequence of events which are collected using the BSD syslog protocol [8]. Syslog monitoring technologies have recently been taking vast attentions in the areas of network risk management, network monitoring, autonomic computing, etc. They are used to address a wide range of important issues including network failure symptom detection and event correlation discovery.

We are mainly concerned with the following three issues of syslog mining:
1)*Failure symptom detection:* It is to detect anomalous events which can be thought of as symptoms of failures as early as possible.
2) *Emerging pattern identification:* It is to identify a syslog pattern which has emerged when an unknown type of network/computer failure has occurred.
3)*Dynamic correlation discovery:* It is to detect event correlations among syslogs for different devices. It is important to recognize how computer devices are dynamically correlated when anomalous events occur.

We require that 1) and 2) be processed in real-time.

The notion of *dynamics* is critical to addressing these three issues from the following two reasons. One reason is that syslogs form a time series. Hence we must consider time correlation of events, which we call *syslog behavior* throughout this paper. This implies that syslog behavior must be represented using a dynamical model and be learned in an on-line fashion. The other reason is that patterns of syslog behavior may dynamically change over time, because computer environments are intrinsically non-stationary. This implies that syslog behavior must be learned under the non-stationarity assumption for syslog sources.

The purpose of this paper is to introduce a new methodology of *dynamic syslog mining* to address the three issues 1)-3) as above and to demonstrate its effectiveness through experiments using real data. The key ideas of our dynamic syslog mining approach are summarized as follows:
I) A dynamic syslog behavior is represented using a finite mixture of HMMs (Hidden Markov Models), which we abbreviate as an *HMM mixture.*
II) The parameters in the HMM mixture are dynamically learned using the *on-line discounting learning algorithm* (see [21],[20]), which learns the model by gradually forgetting out-of-date statistics. This makes the model adaptive to the non-stationary environment.
III) The optimal number of mixture components in the HMM

mixture is dynamically selected on the basis of the theory of *dynamic model selection* (see [10]).
IV) An anomaly score is calculated for a series of messages on the basis of *universal test statistics* and an alert is raised when the anomaly score exceeds a threshold, which is dynamically optimized over time.

Functions II)-IV) are implemented in an on-line fashion.

We expect the following effects of the dynamic syslog mining for the three issues 1)-3) as above: As for the issue 1), it enables us to detect symptoms for failures with less false alarms than any static method. As for the issue 2), it enables us to detect the emergence of a new syslog behavior pattern by dynamically tracking a new component in the HMM mixture. As for the issue 3), it enables us to discover a dynamic correlation among anomalous events for a number of computer devices, e.g., "event message X for device A $\longrightarrow$ event message Y for device B" where X and Y are both related to anomalous events. We empirically demonstrate these effects using data sets collected for a real network.

1.2 Related Work

The technologies of analyzing event log files, including syslog mining, have extensively been explored in the areas of dependable computing, network management, network monitoring, etc. Vaarandi [17] addressed the issue of failure detection using a clustering technique. It first constructs clusters by grouping event logs on the basis of their message characters and then detects failures by tracking anomalous events which do not belong to any existing cluster. Smyth [14] has developed Markov monitoring method for mining sequential patterns to apply it to deep space network monitoring and failure detection.

The issue of event correlation discovery has been addressed in the scenario of fault localization. Conventional approaches to it include alarm correlation method using model-based reasoning systems [4], fault propagation modeling such as belief network or Bayesian networks [16], code-based techniques [22], and AI techniques such as expert systems (see the review paper [15]). Other related works include temporal correlation mining, i.e., episode rule induction [6],[9], and historical event log mining, i.e., periodic pattern/similarity pattern discovery and its visualization [3]. Perng et.al. [12] took a hybrid approach of data-driven and knowledge based methods to event relationship network analysis. A number of log file monitoring tools for rule-based correlation analysis have also been developed (see e.g., [5],[18]).

It has been pointed out in [3] that the technique of sequential pattern mining [1] could be applied to failure detection and time correlation discovery. However, little of previous works addressed the issue of adaptively tracking dynamics of syslog behavior in the non-stationary environment, i.e., how to adaptively track syslog behavior patterns even when they change over time. Hence it is our primary challenge to build a new methodology of syslog mining in terms of not only "dynamic modeling" but also of "adaptive tracking of the dynamics," with applications to failure symptom detection, emerging pattern identification and correlation discovery.

The rest of this paper is organized as follows: Section 2 introduces our methodology of dynamic syslog mining. Section 3 shows experimental results on its applications to failure symptom detection and emerging pattern identification. Section 4 shows its application to dynamic correlation discovery. Section 5 yields concluding remarks.

2. METHODOLOGY

Our approach to dynamic syslog mining consists of the following key components: I) probabilistic modeling using an HMM mixture, II) on-line discounting learning of parameters in the model, III) dynamic model selection for determining optimal mixture components, and IV) scoring using universal test statistics with a dynamically optimized threshold. We describe their details below.

2.1 Probabilistic Modeling

Syslogs form a time series of events. Table 1 shows an example of syslogs. We call each row in Table 1 an *event*. Here "Event Severity" indicates the severity level of the message, "Att1" and "Att2" are fields for processes that generated the message, and "Message" contains free-form text which gives detailed information of the event [8]. In this paper we ignore the time stamp and keep the ordering of events only. Although each event may be multi-dimensional, we consider it as a symbol belonging to a finite alphabet.

In modeling syslogs, we divide syslogs into a number of sessions to get a session stream where each session is a subsequence of events forming a time series. We employ an HMM mixture to represent a probabilistic model of session generation. An HMM mixture is a linear combination of hidden Markov models where each HMM component corresponds to a syslog behavior pattern and a mixture of K components represents that K different patterns exist. Details are described below.

Let $\mathcal{S}$ be a finite set of states such that $|\mathcal{S}| = N_1$ and $\mathcal{Y}$ be a finite set of different event symbols such that $|\mathcal{Y}| = N_2$ where N_1 and N_2 are given positive integers. Let $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$ denote a session stream consisting of M sessions where $\boldsymbol{y}_j = (y_1, \ldots, y_t, \ldots, y_{T_j}) \in \mathcal{Y}^{T_j}$ denotes the j-th session of length T_j $(j = 1, \cdots, M)$. An element y_t denotes the t-th observed event in the j-th session.

For a given positive integer K, we assume that each session is independently drawn according to an *HMM mixture* with K components of the following form:

$$P(\boldsymbol{y}_j \mid \theta) = \sum_{k=1}^{K} \pi_k P_k(\boldsymbol{y}_j \mid \theta_k), \qquad (1)$$

where π_k is a mixture coefficient satisfying $\pi_k > 0$ and $\sum_{k=1}^{K} \pi_k = 1$ $(k = 1, \cdots, K)$, and $P_k(\cdot \mid \theta_k)$ denotes an HMM corresponding to the k-th component specified by the parameter θ_k. We set $\theta = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$.

Let n be a given positive integer. A dynamic structure of each session is represented using an *n-th-order HMM* of the following form: Letting $(s_1, \cdots, s_{T_j}) \in \mathcal{S}^{T_j}$ $(j = 1, \ldots, M)$ be a vector of *hidden states*, we set

$$P_k(\boldsymbol{y}_j \mid \theta_k) = \sum_{(s_1,\ldots,s_{T_j})} \gamma_k(s_1, \ldots, s_n)$$
$$\times \prod_{t=n+1}^{T_j} a_k(s_t \mid s_{t-1}, \ldots, s_{t-n}) \prod_{t=1}^{T_j} b_k(y_t \mid s_t), \qquad (2)$$

where the summation in (2) is taken over the set of all possible combinations of $(s_1, \ldots, s_{T_j})$. Here $\gamma_k(\cdot)$ is an initial probability distribution of a tuple of hidden states $(s_1, \ldots, s_n) \in \mathcal{S}^n$, and all of $a_k(\cdot|\cdot)$s and $b_k(\cdot|\cdot)$s are transition probabilities. We set $\theta_k = (\gamma_k(\cdot), a_k(\cdot|\cdot), b_k(\cdot|\cdot))$.

Table 1: An Example of Syslogs

ID:	Timestamp:	Event Severity:	Att1	Att2	Message
##:	Nov 13 00:06:23:	ERR:	bridge:	!brdgursrv:	queue is full. discarding a message.
##:	Nov 13 10:15:00:	WARN:	INTR:	ether2atm:	Ethernet Slot 2L/1 Lock-Up!!
##:	Nov 13 10:15:00:	WARN:	INTR:	ether2atm:	Ethernet Slot 2L/2 Lock-Up!!
##:	Nov 13 10:15:00:	WARN:	INTR:	ether2atm:	Ethernet Slot 2L/3 Lock-Up!!

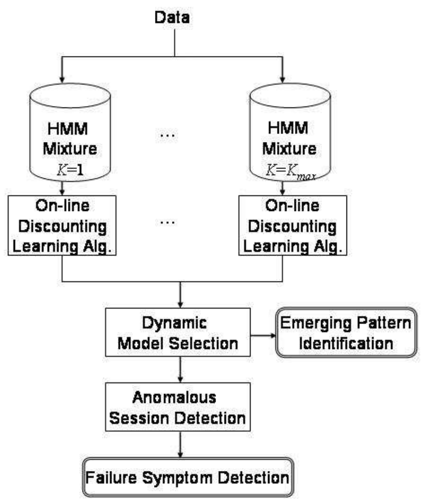

Data
HMM Mixture K=1
...
HMM Mixture K=K_max
On-line Discounting Learning Alg.
...
On-line Discounting Learning Alg.
Dynamic Model Selection
Emerging Pattern Identification
Anomalous Session Detection
Failure Symptom Detection

Figure 1: Flow of Dynamic Syslog Mining

Naive Bayes model (NB) is reduced to the special case where $n = 1$, $K = 1$, $N_1 = 1$ and $\mathcal{S}$ consists of a single state $\{s\}$. Then (2) is written as

$$P_k(\boldsymbol{y}_j \mid \theta_k) = \prod_{t=1}^{T_j} b_k(y_t \mid s).$$

The overall flow of tasks in dynamic syslog mining is illustrated in Figure 1. Events in syslogs are sequentially input to the system. We prepare a number of HMM mixtures, for each of which we learn statistical parameters using the on-line discounting learning algorithm. These tasks are performed in parallel. On the basis of the input data and learned models, we conduct dynamic model selection for choosing the optimal HMM mixture. We then conduct emerging pattern identification using the optimal model. We then conduct anomalous session detection, which leads to failure symptom detection. All of the tasks are implemented in an on-line fashion.

2.2 On-line Discounting Learning Algorithm

We introduce here an algorithm for learning an HMM mixture in an on-line fashion, where the number of components in the HMM mixture is fixed. This algorithm can be thought of as a hybrid of Baum-Welch algorithm for learning HMMs [2] and an *on-line discounting type of EM algorithm for learning mixtures* (see [11],[20]). Below we describe a new variant of the latter for learning HMM mixtures.

Every time a session $\boldsymbol{y}_j$ is given, this algorithm conducts E-step and M-step only once for $k = 1, \ldots, K$ to output estimates of the parameters $\theta = (\pi_k(\cdot), \gamma_k(\cdot), a_k(\cdot|\cdot), b_k(\cdot|\cdot))$. The details of the algorithm are shown in Figure 2. The E-step updates the *membership probability* c_{jk}, which is defined as the probability that the j-th session is generated according to the the k-th component's distribution $P_k(\cdot|\theta_k)$. The M-step updates the parameter θ. For both steps a *discounting parameter* $0 < r < 1$ is introduced in order to make the effect of past statistics gradually decay at every iteration. A larger value of r indicates that it has a smaller influence of past examples. In the M-step $\gamma_{k,1}$, $a_{k,1}$, and $b_{k,1}$, which are sufficient statistics of HMMs, are updated as a weighted average of new statistics (with weight r) and old sufficient statistics (with weight $1 - r$). The state probability $\tau_{k,s_1,\ldots,s_n,s_{n+1},t}$ is the probability that a state sequence from $t - n$ to t is $s_1, \ldots, s_{n+1}$ while $\tau'_{k,s,t}$ is the probability that a state at t is s. They are calculated by Baum-Welch algorithm [2]. In the E-step a parameter ν is further introduced in order to improve the stability of the estimates of c_{jk}. The notation $\sum_{t=1 \wedge y_t = y}^{T_j}$ means that the summation is taken only in the case of $y_t = y$.

The total computation time for the on-line discounting learning algorithm for a 1-order HMM mixture with K components for sample size M is $O(KM(N_1^2 + N_1N_2))$ where N_1 is the number of states and N_2 is the number of different messages.

2.3 Dynamic Model Selection

We are concerned with the issue of detecting the emergence of a new syslog behavior pattern. We reduce here this issue to that of dynamically selecting the optimal number of components for an HMM mixture and tracking its change. We realize this function on the basis of the theory of dynamic model selection (for short, DMS) [10].

According to [10], DMS is classified into two types: *Sequential DMS* and *batch DMS*. The former sequentially outputs the optimal number of mixture components everytime a session is input. Hence it suites the on-line setting as in this paper. Meanwhile the latter outputs a sequence of optimal numbers of mixture components after seeing all the sessions. It rather suites a retrospective variant of our mining setting. Below we show both types of DMS.

2.3.1 Sequential Dynamic Model Selection

The key idea of sequential DMS is to conduct the following process every time a session is input: Sequentially learn a number of HMM mixtures with different numbers of components in parallel, then select the one with the optimal number of components from among them on the basis of the information-theoretic model selection criterion called Rissanen's predictive stochastic complexity [13], shown below.

For a given number K of components, for a given session stream $\boldsymbol{y}^j = \boldsymbol{y}_1, \cdots, \boldsymbol{y}_j$, we define the *predictive stochastic complexity* of the stream relative to the mixture model $P(\cdot|\theta)$ with K components as

Given:
r: discounting parameter
ν: estimation parameter
K:number of mixture components
n: order of HMM
M: data size
Initialization:
for all $k = 1, \dots, K$
Set $\pi_k^0, \gamma_{k,1}^{(0)}(\cdot), \gamma_k^{(0)}(\cdot), a_{k,1}^{(0)}(\cdot), a_k^{(0)}(\cdot|\cdot), b_{k,1}^{(0)}(\cdot), b_k^{(0)}(\cdot|\cdot)$
Procedure:
for $j = 0, 1, \cdots, M-1$
E-step:
for all $k = 1, \dots, K$

$$c_{jk}^{(j)} = (1-\nu r)\frac{\pi_k^{(j)} P_k(\boldsymbol{y}_j \mid \theta_k^{(j)})}{\sum_k \pi_k^{(j)} P_k(\boldsymbol{y}_j \mid \theta_k^{(j)})} + \frac{\nu r}{K}$$

M-step:
for all $k = 1, \dots, K$,
for all $y \in \mathcal{Y}$,
for all $s \in \mathcal{S}$, for all $(s_1, \cdots, s_n, s_{n+1}) \in \mathcal{S}^{n+1}$,
($\tau_{k,s_1,\dots,s_n,s_{n+1},t}$ and $\tau'_{k,s,t}$ are calculated by Baum-Welch algorithm)

$$\pi_k^{(j+1)} = (1-r)\pi_k^{(j)} + rc_{jk}^{(j)}$$

$$\gamma_{k,1}^{(j+1)}(s_1,...,s_n) = (1-r)\gamma_{k,1}^{(j)}(s_1,...,s_n) + rc_{jk}^{(j)} \sum_{s_n+1} \tau_{k,s_1,\dots,s_{n+1},1}$$

$$\gamma_k^{(j+1)}(s_1,...,s_n) = \gamma_{k,1}^{(j+1)}(s_1,...,s_n) / \sum_{(s_1,\dots,s_n)} \gamma_{k,1}^{(j+1)}(s_1,...,s_n)$$

$$a_{k,1}^{(j+1)}(s_1,\dots,s_n,s_{n+1})$$
$$= (1-r)a_{k,1}^{(j)}(s_1,\dots,s_n,s_{n+1}) + rc_{jk}^{(j)} \sum_{t=1}^{T_j-n} \tau_{k,s_1,\dots,s_{n+1},t}$$

$$a_k^{(j+1)}(s_{n+1} \mid s_n,\dots,s_1)$$
$$= a_{k,1}^{(j+1)}(s_1,\dots,s_n,s_{n+1}) / \sum_{s_{n+1}} a_{k,1}^{(j+1)}(s_1,\dots,s_n,s_{n+1})$$

$$b_{k,1}^{(j+1)}(s,y) = (1-r)b_{k,1}^{(j)}(s,y) + rc_{jk}^{(j)} \sum_{t=1\wedge y_t=y}^{T_j} \tau'_{k,s,t}$$

$$b_k^{(j+1)}(y \mid s) = b_{k,1}^{(j+1)}(s,y) / \sum_y b_{k,1}^{(j+1)}(s,y)$$

Figure 2: On-line Discounting Learning Algorithm

$$I(\boldsymbol{y}^j : K) = \sum_{j'=1}^{j} -\log P(\boldsymbol{y}_{j'} \mid \theta^{(j'-1)}), \qquad (3)$$

where $\theta^{(j'-1)}$ is the parameter value estimated by the on-line discounting learning algorithm from a sequence of past sessions: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_{j'-1}$. From the information-theoretic viewpoint, (3) is interpreted as the total code-length required for encoding the stream $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_j$ into a binary sequence in a predictive way. At each time j, we select $K_j^* = K$ minimizing $I(\boldsymbol{y}^j : K)$ over various values of K.

If the optimal number K_j^* of components has become larger than K_{j-1}^* at some point j, then we can recognize that a new pattern has emerged at that time. We then check a new component to identify what behavior pattern has emerged. Similarly, if K_j^* has become smaller than K_{j-1}^* at some time j, then we can recognize that some behavior pattern has disappeared at that time.

2.3.2 Batch Dynamic Model Selection

Batch DMS defines a map from a session sequence $\boldsymbol{y}^M = \boldsymbol{y}_1, \cdots, \boldsymbol{y}_M$ to a model sequence $K^M = K_1, \cdots, K_M$ for any data size M. Here each model indicates the number of components in the HMM mixture. We denote an HMM mixture with K components and parameter value θ as $P(\boldsymbol{y} : \theta, K)$.

Suppose that K_j is determined by $K^{j-1} = K_1 \cdots K_{j-1}$ for all j. We introduce a *model transition probability* $P_j(K_j = K|K^{j-1} : \alpha)$ as the probability of $K_j = K$ given K^{j-1} specified by the parameter α where α is unknown.

Initially choose K_0 arbitrarily. For a session sequence $\boldsymbol{y}^M$ and a model sequence K^M, we define an information-theoretic criterion for batch DMS as follows:

$$\ell(\boldsymbol{y}^M : K^M) = \sum_{j=1}^{M} -\log P(\boldsymbol{y}_j|\theta^{(j-1)} : K_j) \qquad (4)$$
$$-\sum_{j=1}^{M} \log P(K_j|K^{j-1} : \alpha^{(j-1)}),$$

where $\theta^{(j-1)}$ is the parameter value estimated by the on-line discounting learning algorithm from a past session sequence $\boldsymbol{y}^{j-1}$ and $\alpha^{(j-1)}$ is the parameter value estimated from a past model sequence K^{j-1}. Here the first term in the right-hand side of (4) is the predictive stochastic complexity of $\boldsymbol{y}^M$ relative to K^M and the second term is the predictive stochastic complexity of K^M itself. Hence (4) is considered as the total codelength required for encoding $\boldsymbol{y}^M$ and K^M into a binary sequence in a predictive way. Batch DMS procedure takes $\boldsymbol{y}^M$ as input and outputs $K_1^* \cdots K_M^* = K^M$ minimizing the criterion (4).

Note that the result for batch DMS differs from that for sequential DMS in general. Batch DMS works better than sequential DMS in the sense that the value of criterion (4) for batch DMS tends to be smaller than the total code-length for sequential DMS (see [10]). Meanwhile, sequential DMS can process a session stream in an on-line manner while batch DMS processes it in a retrospective manner only.

We make an assumption that a model can only transit to the neighbouring states at each time, i.e., the transiton probabilities are given as follows:

$$P(K_0) = 1/K, \qquad (5)$$
$$P(K_j|K_{j-1}) = \begin{cases} 1-\alpha, \text{ if } K_j = K_{j-1} \text{ and } K_{j-1} \neq 1, K_{max}, \\ 1-\alpha/2, \text{ if } K_j = K_{j-1} \text{ and } K_{j-1} = 1, K_{max}, \\ \alpha/2, \text{ if } K_j = K_{j-1} \pm 1, \end{cases}$$

where $0 < \alpha < 1$ is unknown and K_{max} is the maximum value of K.

We employ the batch DMS algorithm proposed in [10] for efficiently computing the model sequence minimizing the criterion (4). The details are shown in Figure 3. In this algorithm α as in (5) is estimated using the Krichevsky and Trofimov method [7], while the optimal path is calculated using the dynamic programming method like the Viterbi algorithm [19]. The computation time is $(K_{max}M^2)$.

Once the model sequence is obtained using batch DMS, we can detect change points in the sequence to check the emergence or disappearance of syslog behavior patterns, as with sequential DMS.

Given:
K_{max}: maximum number of K
M: data size
For each K, $\theta^{(0)}$: initial parameter value
Initialization:
$j = 1$
For each K,
$S(K, 0, 1) = \log K_{max} - \log P(\boldsymbol{y}_1 \mid \theta^{(0)} : K)$,
$\boldsymbol{K}(K, 0, 1) = (K)$.
Procedure:
$N_{K,j}$: number of change points in $K_1, \dots, K_j = K$.
$\hat{P}_j(K|K', \alpha(N_{K',j-1}))$: probability value obtained by substituting $\alpha(N_{K',j-1})$ for α of (5).
For each K, $N_{K,j}$, j, ($j = 2, \dots, M$, $N_{K,j} = 0, \dots, j-1$)
Model Selection

$$\begin{aligned}
&S(K, N_{K,j}, j) \\
&= \min_{K', N_{K',j-1}} \{S(K', N_{K',j-1}, j-1) - \log P(\boldsymbol{y}_j | \theta^{(j-1)} : K) \\
&\quad - \log \hat{P}_j(K|K', \alpha(N_{K',j-1}))\},
\end{aligned}$$

$$\begin{aligned}
&(\tilde{K}, \tilde{N}_{\tilde{K},j-1}) \\
&= \arg \min_{K', N_{K',j-1}} \{S(K', N_{K',j-1}, j-1) \\
&\quad - \log P(\boldsymbol{y}_j | \theta^{(j-1)} : K) - \log \hat{P}_j(K|K', \alpha(N_{K',j-1}))\},
\end{aligned}$$

$$\boldsymbol{K}(K, N_{K,j}, j) = \boldsymbol{K}(\tilde{K}, \tilde{N}_{\tilde{K},j-1}, j-1) \oplus K.$$

Estimation of Model Transition Probabilities

$$\alpha(N_{K,j}) = \frac{N_{K,j} + \frac{1}{2}}{(j-1) + 1}.$$

Output the Optimal Path:
$j = M$
$(K_M^*, N^*_{K_M^*,M}) = \arg\min_{K, N_{K,M}} S(K, N_{K,M}, M)$,
Output $(K_1^*, \dots, K_M^*) = \boldsymbol{K}(K_M^*, N^*_{K_M^*,M}, M)$.

Figure 3: Batch Dynamic Model Selection

2.4 Anomalous Session Detection

We give an anomaly score to each session where a higher score indicates a higher possibility that the session is anomalous and thus might be related to failure. Hence the detection of failures or their symptoms from syslogs can be reduced to the issue of anomalous session detection. The details of our scoring method are described below.

For the j-th observed session $\boldsymbol{y}_j$ with length T_j, for the learned HMM mixture P for which the number of components is determined by dynamic model selection, we define its anomaly score by

$$\mathrm{Score}(\boldsymbol{y}_j) = -\frac{1}{T_j} \log P(\boldsymbol{y}_j \mid \theta^{(j-1)}) - \mathrm{compress}(\boldsymbol{y}_j), \quad (6)$$

where $\theta^{(j-1)}$ is the parameter value estimated by our algorithm from a sequence of past sessions: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_{j-1}$. The function compress($\boldsymbol{y}$) denotes a compression rate of $\boldsymbol{y}$ for a noiseless universal data compression scheme such as Lempel-Ziv algorithm [23]. The quantity (6) is known to be a *universal test statistics* developed by Ziv [24]. In this scoring,

Given:
N_H: total number of cells
ρ: parameter for threshold
λ_H: estimation parameter
r_H: discounting parameter
M: data size
Initiallization:
Let $\{q_1^{(1)}(h)\}$ (a weighted sufficient statistics) be a uniform distribution.
for $j = 1, ..., M-1$, $h = 1, ..., N_H$,
Alarm Output:
For the j-th session, make an alarm if and only if $\mathrm{Score}(\boldsymbol{y}^j) \geq \eta(j)$.
Updating Rule:
if $\mathrm{Score}(\boldsymbol{y}^j)$ falls into the cell indexed by h
then $q_1^{(j+1)}(h) = (1 - r_H) q_1^{(j)}(h) + r_H$
otherwise $q_1^{(j+1)}(h) = (1 - r_H) q_1^{(j)}(h)$.
$q^{(j+1)}(h) = (q_1^{(j+1)}(h) + \lambda_H) / (\sum_h q_1^{(j+1)}(h) + N_H \lambda_H)$.
Threshold Optimization:
Let $\eta(j+1)$ be the least index such that
$\sum_{h'=1}^{h} q^{(j+1)}(h') \geq 1 - \rho$.

Figure 4: Dynamic Threshold Optimization

if any two sessions take the same values for the first term (Shannon information), then the one with higher regularity, eventually with a smaller compression rate, would result in a larger anomaly score.

We set a threshold for anomaly scores, then we determine that any session is anomalous if its anomaly score exceeds the threshold. Here the threshold must be optimized adaptively to the score distribution, which may dynamically change over time.

The fundamental idea of threshold optimization is as follows: We use a 1-dimensional histogram for the representation of the score distribution. We learn it in an on-line and discounting way as with the technique of Section 2.2, then, for a specified value ρ, to determine the threshold to be the largest score value such that the tail probability beyond the value does not exceed ρ.

We sequentially learn a histogram of anomaly score values every time a session is input. Let N_H be a given positive integer. Let $\{q(h)(h = 1, ..., N_H) : \sum_{h=1}^{N_H} q(h) = 1\}$ be a 1-dimensional histogram with N_H bins where h is an index of bins, with a smaller index indicating a bin having a smaller score. For given a, b such that $a < b$, N_H bins in the histogram are set as: $(-\infty, a)$, $[a + \{(b-a)/(N_H - 2)\}\ell, a + \{(b-a)/(N_H - 2)\}(\ell + 1))$ $(\ell = 0, 1, ..., N_H - 3)$ and $[b, \infty)$. Let $\{q^{(j)}(h)\}$ be a histogram updated after seeing the j-th session $\boldsymbol{y}_j$. The procedures of updating the histogram and threshold optimization are given in Figure 4.

We usually set $N_H = 20$, $\rho = 0.05$, $\lambda_H = 0.5$, and $r_H = 0.001$.

3. EXPERIMENTAL RESULTS

3.1 Data Set

We used four syslog data sets each of which was collected for a server in an ATM (Asynchronous Transfer Mode) net-

Table 2: **Failure Symptom Detection Result**

server	“lock-up” time	lead time	alarm/total (event)	computation (sec)
HMM mixture				
A	11/13/01 10:15:00	11/11/01 10:14:50	188/17859	12.31
	11/20/01 03:10:07	11/20/01 02:44:00		
	01/15/02 15:01:42	01/10/02 12:35:30		
B	11/26/01 15:02:56	11/26/01 15:03:00	1473/14533	16.76
C	01/24/02 09:34:18	01/18/02 11:41:30	167/15273	15.92
D	11/16/01 21:01:37	11/14/01 17:47:50	202/26147	12.67
	11/21/01 18:31:29	-		
	12/10/01 20:21:15	-		
	01/28/02 21:56:23	01/28/02 12:32:10		
	01/30/02 18:53:37	01/29/02 18:22:20		
NB				
A	11/13/01 10:15:00	11/11/01 10:14:50	293/17859	2.29
	11/20/01 03:10:07	11/20/01 02:44:00		
	01/15/02 15:01:42	01/10/02 06:36:40		
B	11/26/01 15:02:56	11/26/01 15:03:00	1494/14533	2.60
C	01/24/02 09:34:18	01/24/02 09:32:30	1387/15273	2.12
D	11/16/01 21:01:37	11/14/01 17:47:50	1490/26147	2.61
	11/21/01 18:31:29	-		
	12/10/01 20:21:15	-		
	01/28/02 21:56:23	01/28/02 12:32:10		
	01/30/02 18:53:37	01/29/02 18:22:20		

work from Nov. 2001 to Jan. 2002. Event severity of all of the events were under 4. Note that severity score ranges from 0 to 7, with a lower score indicating higher severity [8]. The numbers of events for these sets were 17859, 14533, 15273, and 26147, respectively.

Each data set took the form as in Table 1. As a preprocessing step, we deleted all of the numerical information in messages and identified each message by its character information only. An event represented by (Event Severity, Att1, Att2, Message) was transformed into one symbol. Thus every event was dealt with as a symbol ranging over a finite alphabet. We utilized time stamps only for the purpose of constructing sessions, as shown below.

For each data set, we constructed a session stream by grouping events of identical occurrence time in tens of seconds (hh:mm:s) into one session. For example, the event which occurred at time 10:12:40 and the one at time 10:12:48 were grouped together into one session. The numbers of events for the four data sets were 17859, 14533, 15273, and 26147, respectively, while the numbers of sessions for them were 4334, 4111, 5040, and 2215, respectively. The total number of different events (N_2) for the four data sets were 34, 22, 23, and 36, respectively.

3.2 Failure Symptom Detection

We applied our methodology to failure symptom detection for the data set as above. Message “Ethernet Slot 2L1/1 Lock-up” as shown in Table 1, which we abbreviate as “lock-up,” is a critical failure message in syslogs. As for our data sets, the “lock-up” was the failure which a network operator taking care of the network system thought most critical and was mainly concerned with. The numbers of “lock-up” event series in the four data sets were 3, 1, 1, and 5, respectively.

We are interested in detecting anomalous sessions related to “lock-up” earlier than it actually occurs. We call such anomalous sessions *failure symptoms*. Note that it is practically hard to decide whether any alarm of an anomalous session is truly related to the failure or not. Meanwhile, in the real network management system, an operator must check the network status every time an alarm is raised regardless of whether it is truly a symptom or not. Hence in the evaluation of this experiment we formally define a failure symptom as any alarm that is raised within one week before “lock-up.” This definition seems reasonable from the standpoint of network operation. We are thus concerned with the issue of how early our method is able to detect failure symptoms and how many alarms it gives in total. Earlier detection with less alarms would be a better solution to this issue.

We employed an HMM mixture as a dynamic probabilistic model of syslog behavior patterns. The parameter values were set as follows: $N_1 = 3$, $r = 0.1$, $\nu = 0.5$, $n = 1$, $N_H = 20$, $\rho = 0.05$, $\lambda_H = 0.5$, and $r_H = 0.001$. As for a dynamic model selection procedure, we employed sequential DMS. For the sake of comparison, we also employed a Naive Bayes model, which we abbreviate as NB, as a static probabilistic model. For NB, the number of mixture components was set 1 and $r = 1/j$. Hence this variant performs neither discounting learning nor dynamic model selection. In both models the first 10 sessions were fed to learning algorithms without scoring.

As a post-processing step, we applied the following rule: Once an alarm is raised, ignore all of the alarms that were raised within 60 minutes after that time.

Table 2 shows the results on failure symptom detection of our method for HMM mixture and NB. “Lock-up” time is the time when “lock-up” failure actually occurred. Lead time is the earliest time when any alarm was raised within one week before the lock-up time. The symbol “$-$” shows that the method was not able to detect anything. Alarm/total (event) means the ratio of the total number of event alarms over the total number of events. Notice here that one session consists of a number of events, and that an alarm was made per session. Hence for an alarmed session, all of the events included in the session were considered as alarmed events. Computation time was measured for PC with Pentium IV 2GHz, 768MB.

We observe from Table 2 that as for the lead time HMM mixture is almost comparable to NB except for two cases (servers A and C). For example, for server A, both methods detected failure symptoms 30 min $\sim$ 2 days earlier than "lock-up." Most of the lead times were early enough for operators. However, alarm/total for HMM mixture is significantly smaller than NB. For example, alarm/total for HMM mixture is around 1/100 while that for NB is around 1/50 $\sim$ 1/10. This implies that the dynamic approach makes more reliable alarms than the static one.

3.3 Emerging Pattern Identification

Next we are concerned with the issue of identifying emerging behavior patterns related to "lock-up" failures. This is conducted by looking into a new emerging component in the HMM mixture when the optimal number of components has changed.

A component in an HMM mixture is not necessarily easy to understand the syslog behavior pattern of messages because it includes a number of hidden variables. Hence we transform an HMM into a *Markovian expression* in order to make it more readable. This is constructed in the following way: We classify each data into a cluster in the mixture with the largest posterior probability. Then for each cluster we learn a Markov model from a data set assigned to it, to obtain a Markovian expression, i.e., a list of transitions in a descendent order with respect to their probabilities.

Figure 5 shows how the number of mixture components (clusters) changed over time. The horizontal axis shows the session number while the vertical one shows the optimal number of components in the HMM mixture calculated on the basis of dynamic model selection, as in Section 2.3.

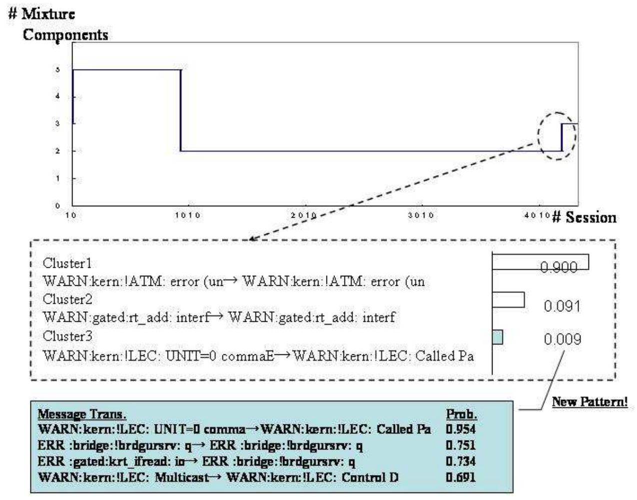
We observe that the change occurred at the 4211-st session (time Jan.15th 15:04:10), immediately after "lock-up" failure. At that time the number of mixture components increased from 2 to 3. Figure 5 shows Markovian representations of the three components in the HMM mixture where the occurrence probabilities for the components were 0.90, 0.091, and 0.009, respectively. A correlation rule with the highest transition probability is shown for each cluster.

The component with the smallest occurrence probability turned out to be a pattern which newly emerged at the change point. Figure 5 further gives a Markovian representation of this component. In comparison with other clusters, we found that this component was characterized by the following patterns:

"ERR:bridge:!bridgursrv: q → ERR:bridge:!!bridgursrv: q"
with transition probability 0.751.

"ERR:gated:krt_ifread:in → ERR:bridge:!!bridgursrv: q."
with transition probability 0.734.

"WARN:kern:!LEC:Multicast → WARN:kern:!LEC:Control D
with transition probability 0.691.



Cluster1
WARN:kern:!ATM: error (un→ WARN:kern:!ATM: error (un 0.900
Cluster2
WARN:gated:rt_add: interf→ WARN:gated:rt_add: interf 0.091
Cluster3
WARN:kern:!LEC: UNIT=0 commaE→WARN:kern:!LEC: Called Pa 0.009

Message Trans.	Prob.
WARN:kern:!LEC: UNIT=0 comma→WARN:kern:!LEC: Called Pa	0.954
ERR :bridge:!brdgursrv: q→ ERR :bridge:!brdgursrv: q	0.751
ERR :gated:krt_ifread: io→ ERR :bridge:!brdgursrv: q	0.734
WARN:kern:!LEC: Multicast→ WARN:kern:!LEC: Control D	0.691

Figure 5: Identified Syslog Behavior Patterns

The first pattern means the repetition of the message that "queue is full." The second one means that "interface error propagates to queueing error." The third one means that "control direct VCC down follows multicast forward VCC down."

These patterns actually characterize the syslog behavior associated with "lock-up" failure for this server. Thus we were able to successfully identify the syslog behavior patterns related to a critical failure. This led to knowledge discovery for the network system of concern.

4. DYNAMIC CORRELATION DISCOVERY

4.1 Methodology

Suppose that a number of computer devices are connected one another within a network, where syslogs are observed for each computer device. We are interested in the issue of discovering how these devices are dynamically correlated when anomalous events which might be related to failures occur. A straightforward solution to it is first to synchronize and merge log files for all of the computers to obtain a total log file, and then to discover correlations such as belief networks from among them. However, in such a solution we may suffer from the problem that many redundant rules which are not necessarily related to failures are produced, and thus significant failure-related information is buried in them. Below we give another solution to overcome this problem.

The basic flow of our approach is illustrated in Figure 6. The key idea is to conduct two stage learning of syslog dynamics; one is for *syslog quantization* for individual syslogs while the other is for *correlation discovery* for merged syslogs. Specifically syslog quantization is an important step because it makes our attention focus on the correlations of anomalous events. The fundamental steps of the two stage learning are summarized as follows:

1st Step: Quantization. For each computer device, we dynamically learn an HMM mixture from syslogs and then conduct anomalous session detection using the techniques as in Sections 2.2-2.4. We then *quantize* sessions included

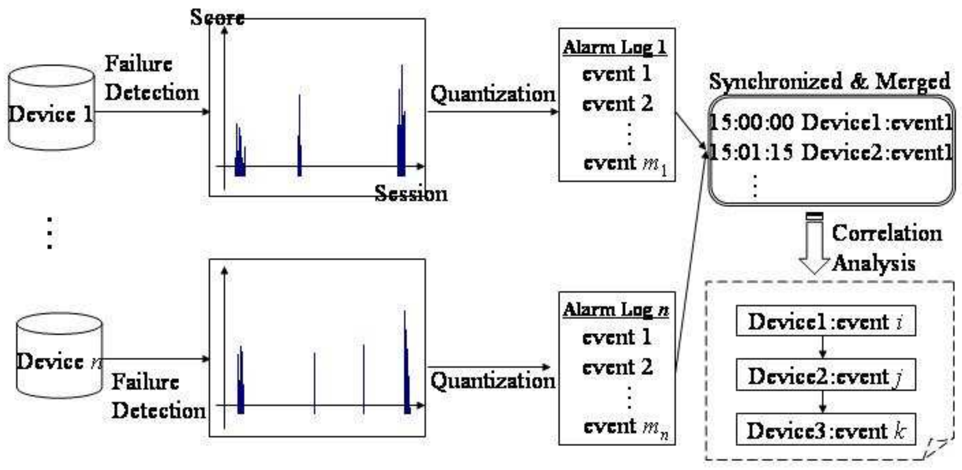


Figure 6: Flow of Correlation Discovery

in original syslogs by classifying them into anomalous ones and normal ones where any session is determined to be an anomalous one if and only if its score exceeds the threshold with $\rho = 0.05$, otherwise a normal one. Here original event messages are preserved only for anomalous sessions, while those for normal ones are ignored and are uniformly transformed into a single message "others."

2nd Step: Correlation Discovery. We synchronize and merge the quantized sessions for all the servers to get a total log file. We dynamically learn from it an HMM mixture again using the techniques as in Sections 2.2 and 2.3. Components in the resulting HMM mixture show dynamic correlations among different computer devices.

4.2 Experimental Results

We applied the above methodology to conduct correlation analysis using syslogs for 21 computer devices within a network. The total number of events included for the merged session stream was 181363. For each device a session stream was constructed in the same way as in Section 3.1 while sessions in the merged sequence were constructed by sliding a window of length 10. The total number of sessions in the merged session stream was 181354. The total number of different event messages which appeared in the merged syslogs was 132. We employed the same parameter values for learning HMM mixtures as in Section 3.2 and sequential DMS for a dynamic model selection procedure.

Table 3 shows examples of discovered correlation rules. For example, the first line shows that when anomalous sessions emerge,

Message "WARN:kern:!LEC: Called Pa"

for server B1 appears, then

Message "WARN:kern:!LEC: UNIT=0 commaE"

follows for server A10 with probability 4.03999e-01.

Figure 7 shows a macro correlation map which we obtained. Servers specified by an identical alphabet were located at the same node of ATM network. All of the servers which were not correlated with other servers were dropped off from this figure. A width of a connection line is proportional to the transition probability, i.e., a wider line indicates a higher correlation. The numerical value associated with each line shows the occurrence number of transitions between the servers connected by the line.

We may observe from Figure 7 that there were strong connections between C1 and C9 and between C3 and C4. This fact is not surprising because servers located at the same node may usually be correlated. Meanwhile, it should be noticed that there were found some strong correlations between different nodes: e.g., between A10 and B1 and between B9 and D8. This seems quite interesting because it indicates that different nodes were highly correlated each other when anomalous sessions occurred. It suggests that for example, any failure may propagate from the node A to the node B with high probability. This correlation discovery has actually been appreciated by operators and network designers taking care of this network management system.

Table 3: Examples of Correlations

Message Trans.	Prob.
B-1: WARN:kern:!LEC: Called Pa → A-10: WARN:kern:!LEC: UNIT=0 commaE	4.03999e-01
A-10: WARN:kern:!LEC: Called Pa → B-1: WARN:kern:!LEC: UNIT=0 commaE	2.75850e-01
E-1: CRIT:gated:KRT SEND DELET → C-1: WARN:kern:!LEC: Control D	2.69846e-01
C-4: WARN:gated:OSPF RECV Area → C-1: WARN:kern:!LEC: UNIT=0 commaE	2.69220e-01

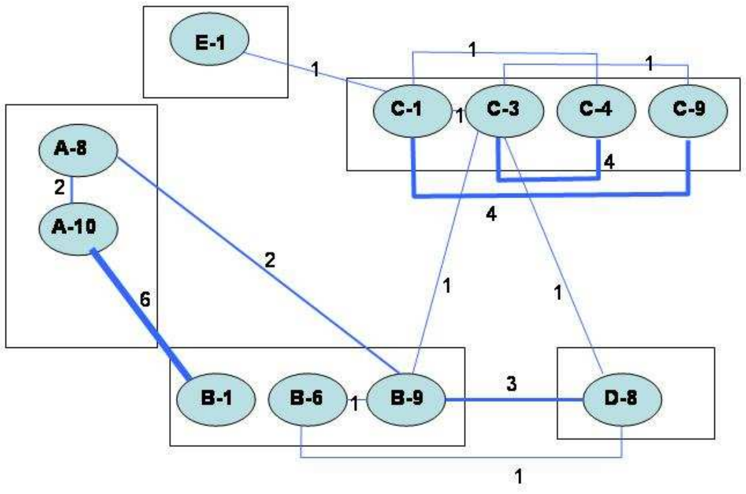


Figure 7: Correlation Map

5. CONCLUSIONS

We have introduced a new methodology of dynamic syslog mining for network failure monitoring. This includes four key techniques; I) probabilistic modeling using an HMM mixture, II) on-line discounting learning of parameters in the model, III) dynamic model selection for determining the optimal number of mixture components, and IV) scoring sessions using universal test statistics with a dynamically optimized threshold. We have demonstrated the validity of our methodology using real syslog data in the scenarios of failure symptom detection, emerging pattern identification and dynamic correlation discovery. Our methodology can be straightforwardly applied to the analysis of a wide range of event log files, including system calls, command lines, Web access logs, etc. In this paper we have focused on mining symbolic data. It is left for future study how to extend our work in order to discover richer knowledge from syslogs by mining numeric data in combination with symbolic data.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Eleventh International Conference on Data Engineering (ICDE95)*, pages 3-14, 1995.

[2] L. E. Baum and T. Petrie and G. Soules and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Statistics*, 41(1):164-171,1970.

[3] L. Burns and J. L. Hellerstein and S. Ma and C. S. Perng and D. A. Rabenhorst and D. Taylor. A systematic approach to discovering correlation rules for event management. In *Proc. of IEEE/IFIP International Sysmposium on Integrated Network Management*, 2001.

[4] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Networks*, 37:52-59, 1993.

[5] S. E. Hansen and E. T. Atkins. Automated system monitoring and notification with swatch. In *Proc. of USENIX Seventh System Administration Conference (LISA93)*, 1993.

[6] M. Klemettinen and H. Mannila and H. Toivonen. Rule discovery in telecommunication alarm data. *Journal of Network and Systems Management*, 7(4): 395-423, 1999.

[7] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Trans. on Inform. Theory*, 27:199-207, 1981.

[8] C. Lonvick. The BSD syslog protocol, RFC, 3164, 2001.

[9] H. Mannila and H. Toivonen and A. I. Vernamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259-289, 1997.

[10] Y. Maruyama and K. Yamanishi. Dynamic model selection with its applications to computer security. In *Proc. of 2004 IEEE International Workshop on Information Theory*, 2004.

[11] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. Learning in Graphical Models, M. Jordan (editor), *MIT Press*, Cambridge, MA, USA, pages 355-368, 1999.

[12] C-S. Perng and D. Thoenen and G. Grabarnik and S. Ma and J. Hellerstein. Data-driven validation, completion and construction of event relationship networks. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD2003)*, pages 729-734, 2003.

[13] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. on Inform. Theory*, 30:629-636, 1984.

[14] P. Smyth. Markov monitoring with unknown states. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Intelligent Signal Processing for Communications*, 1994.

[15] M. Steinder and A. Sethi. The present and future of event correlation: A need for end-to-end service fault localization. In *Proc. of 2001 World Multi-Conference on Systemics, Cybernetics and Informatics*, 2001.

[16] M. Steinder and A. Sethi. Probabilistic fault localization in communication systems using belief networks. *IEEE Trans. on Networking*, 12(5):809-822, 2004.

[17] R. Vaarandi. A data clustering algorithm for mining patterns from event logs. In *Proc. of 2003 IEEE Workshop on IP Operations & Management (IPOM2003)*, 2003.

[18] R. Vaarandi. Sec - a lightweight event correlation tool. In *Proc. of 2002 IEEE Workshop on IP Operations & Management (IPOM2002)*, 2002.

[19] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Inform. Theory*, IT-13:260-267, 1967.

[20] K. Yamanishi and J. Takeuchi and G. Williams and P. Milne. On-line unsupervised oultlier detection using finite mixtures with discounting learning algorithms. In *Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD2000)*, pages 320-324, ACM Press, 2000.

[21] K. Yamanishi and J. Takeuchi. A unifying framework for detecting outliers and change-points from non-stationary time series data. In *Proc. of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD2002)*, pages 676-681, ACM Press, 2002.

[22] S. A. Yemini and S. Kliger and E. Mozes and Y. Yemini and D. Ohsie. High speed and robust event correlation. *IEEE Communications Magazine*, 34(5):82-90, 1996.

[23] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Trans. on Inform. Theory*, IT-24:530-536, 1978.

[24] J. Ziv. On classification with empirically observed statistics and universal data compression. *IEEE Trans. on Inform. Theory*, IT-34:278-286, 1988.