Learning Baseline Models of Log Sources

Ashot N. Harutyunyan, Arnak V. Poghosyan, Nicholas Kushmerick,
and Naira M. Grigoryan

Office of CTO of Cloud Management, VMware
{aharutyunyan;apoghosyan;nicholask;ngrigoryan}@vmware.com

Abstract. Leveraging cloud management products to effectively control performance of IT applications and infrastructures inevitably leads to the issue of automatically identifying *baseline structures* (typical behavioral patterns) of measured data sets including log sources. Those structures can be utilized for a variety of purposes from anomaly and change detection, to characterization of the application or infrastructure state in large (for instance, high/low stress levels, sickness, overprovisioning, security threats, etc.). Particularly, VMware vRealize Operations Manager performs such an analysis for any time series metric of an IT object through its basic dynamic thresholding analytics, while building a similar capability for log analytics is challenging - the very high volume of log data makes machine learning extremely expensive. To overcome the learning complexity, we propose *random sampling* techniques. Our method allows for controlling the confidence of the learned model by tuning the sampling rate. In this paper, we focus on learning the baseline model of log sources in terms of the distribution of log event types generated by vRealize Log Insight. Moreover, our algorithms identify the *expected normal discrepancy* from such a baseline that the log source exhibits. We demonstrate the proposed approach by applying our prototype algorithms to different data sets.

Keywords: Automated log management, baseline model/structure, sampling with confidence control, binomial distribution, anomaly detection, state characterization, clustering, machine learning.

1 Introduction

With VMware's growing interest in application-aware cloud management and analytics, the log intelligence becomes especially import. VMware's cloud management solutions vRealize Operations (vR Ops) [1] and vRealize Log Insight (vR LI) [2] are the main platforms to empower the modern Software-Defined Data Center management with automated machine learning capabilities, self-tuning and optimization, and move forward into an AI-enabled autonomous management in the cloud computing market.

For vR Ops, the current state-of-the-art in terms of data analytics is based on dynamic thresholding [3] and capacity forecasting analysis for time series data of the environment objects.

vR LI supports two important machine learning features of 1) Event Types as similarity clusters of raw log data and the 2) Event Trends allowing to compare to selected time windows by their differences of corresponding event types. While vR Ops performs pattern detection for any metric data from data center objects and derives expected ranges of processes based on a complex time series analysis, accounting for change, trends, and periodicity, LI is lacking a similar capability to automatically identify the main behavioral patterns of the log source. It makes troubleshooting and pattern detection in log data mostly a query-based task with intensive user efforts to find problem root causes or track the application state in general.

In this regard, to have a much intelligent characterization of the application, identification of its *baseline model* or behavioral fingerprint from logs history is of exceptional importance. Intelligent proactive management of data centers and applications from logs perspective with building an accurate expert baseline requires a huge knowledgebase and extensive efforts. At the same time, it cannot be easily generalizable because of many factors coming from conditions of the IT ecosystem. Hence, machine identification of application baselines can be a powerful addition to any log analytics. Evidently, with such a fundamental structure then, the real-time anomaly detection and many other core tasks will be easily automated. By comparing the current log stream against its historically typical model, we'll be able to effectively describe the state of the application in real-time and efficiently identify issues and incidents (a new software bug, sickness, hardware failure, software upgrade, configuration changes, change in workload) related to various aspects of data center management (troubleshooting, performance monitoring, capacity planning, provisioning and configuration, compliance auditing, policy enforcement, etc.). This means that those structures should be enough informative to reveal the whole complexity of the log stream with sophisticated relationships between events.

Any method dealing with extracting and continuously updating baseline structures along the log stream requires an expensive unsupervised learning plan. Although alternative approaches applying meta-data analysis or quantification of log information bypass such a complexity (discussed in Section 2), however they address only a specific problem without structural characterization of the log source in general.

In this paper, we propose a random sampling technique to overcome the learning complexity of baselines subject to confidence level of the learned model based on binomial distributions. We focus on learning the baseline in terms of the probability distribution of event types that LI builds from the stream with similarity analysis of log messages. Moreover, our algorithms identify the *expected normal discrepancy* from such a baseline that the log source historically exhibits. Our prototype was applied to different data sets to validate and demonstrate the approach.

In Section 2 we motivate our research and discuss the related work. Section 3 describes our methods to identifying baselines of log sources and also demonstrateing their application to log data sets. In Section 4 we specify larger experimental plans and conclude in Section 5.

2 Motivation and Related Work

As we mentioned in the introduction, one approach to overcome the complex machine learning tasks for log data is to extract different meta-data properties from those sets and proceed with numeric data (time series) analysis (e.g. [4]) or build event correlation models (e.g. [5]). In particular, in our earlier work [4] we applied quantification of log data with information theory [6]. The quantified/extracted time series metric representing stream's Jensen-Shannon divergence over time was analyzed for change detection purposes. Although this kind of metric plus log analytics framework empowers the log intelligence with highly effective toolset of low complexity, but it remains an indirect method for behavioral analysis (without revealing the complete characteristics of the log source and hiding much of the content in logs).

In [4] we intensively utilized distributions of event types generated by LI in change point detection for a single source, sickness detection of a source within population of similar sources, as well as for an application topology discovery using hierarchical clustering. Below we are going to utilize LI's event types further to identify the sought baseline models for log sources.

Importance of baseline models for VMware's log analytics was first realized in [7], where authors applied information divergence measures to detect anomalies subject to a known/assumed baseline distribution of event types. The work was largely motivational for us to address the problem of automatic discovery of baseline distributions of log events.

For a short overview on Event Types by LI, let us mention that they are the main machine learning constructs of the product that represent abstract clusters of raw log events into similarity groups. With such a similarity grouping the product performs a dramatic data reduction, mapping thousands or millions of log messages into a manageable number of groups/types. Fig. 1 illustrates log data of a source for a 10-minute period as a bar chart of events of distinct types (in different colors). It highlights those distinct groups in a fractional view within each bar of the chart. Those fractions/rates in each bar (10 seconds) of the chart can be converted into relative frequencies or probability distributions of event types within the window. Then if we want to compare two log portions in term of their content we can apply information measures (or other similarity distances like *cosine*) to estimate their "difference".

Particularly, taking relative frequencies of event types observed in two log portions as probability distributions

$$P = (p_1, p_2, \dots p_n) \text{ and } Q = (q_1, q_2, \dots q_n)$$

of n different event types, we applied [4] Jensen-Shannon divergence varying between 0 and 1:

$$JSD(P,Q) = \frac{1}{2}D(P,M) + \frac{1}{2}D(Q,M),$$

where $M = \frac{P+Q}{2}$ and $D(P,Q)$ is the Kullback-Leibler divergence [6] between P and Q:

$$D(P,Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}.$$

Respectively, the cosine similarity is based on the angle between two vectors:

$$\cos(\theta) = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2} \sqrt{\sum_{i=1}^{n} {q_i}^2}}.$$

For more details regarding event types, their probabilistic representation and application of information measures to anomaly, change, and sickness detection we refer the reader to the same papers [4,7].

Based on the above review, the proposed machine learning identification of baseline structures for log sources is a novel formulation.

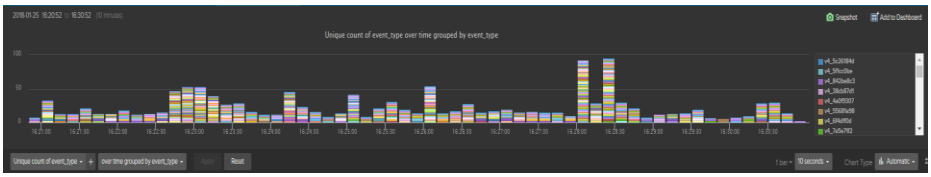

Fig. 1. Bar chart representation of fractions of distinct event types by LI for each 10 seconds within a 10-minute window.

3 Identifying Baselines with ML

Our algorithms are based on random sampling employing the binomial probability distribution. This allows us to tune the confidence of our ML algorithms. Below we give a brief information about the binomial distributions.

3.1 Binomial Distribution and Sampling

The binomial distribution [8] with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments, widely used in probability theory and statistics. A single success/failure experiment is also called a Bernoulli trial or Bernoulli experiment and a sequence of outcomes is called a Bernoulli process. The binomial distribution is the basis for the popular binomial test of statistical significance. It is frequently used to model the number of successes in a sample of size n drawn with replacement.

The number of success (Yes) k versus failure (No) $n-k$ probability in n trials is given by the formula:

$$Prob(k\ success\ in\ n) = C_n^k p^k (1-p)^{n-k},$$

where p is the probability of success in a single trial.

Let us assume that the log source stays at its normal operational state most (99%) of the time. Instead of 99% can be another prior probability. This means that if we randomly sample some log portions during the progress of the stream, we'll get mostly normal (i.e., the success outcome) behavioral patterns of the log source (let us say in terms of its event types).

How many randomly sampled event type distributions are "enough" to verify if 99% of those distributions describe the normal mode of the log stream? The answer of

the question can be given with the help of binomial distributions. Namely, if normal samples occur with probability 0.99 versus 0.01, then applying the binomial distribution (see the online calculator http://stattrek.com/online-calculator/binomial.aspx) we can measure how many sampled probability distributions of event types are "enough" to identify the "normal" (success) ones. Fig. 2 shows the calculator in action.

Probability of success on a single trial	0.99
Number of trials	5
Number of success	4
Binomial Probability: P(X = 4)	0.0480298005
Cumulative Probability: P(X < 4)	0.0009801496
Cumulative Probability: P(X <= 4)	0.0490099501
Cumulative Probability: P(X > 4)	0.9509900499
Cumulative Probability: P(X >= 4)	0.9990198504

Fig. 2. Results of the binomial distribution calculation for 5 trials with 4 "success".

It illustrates that from 5 distributions at least 4 are the normal patterns with confidence (cumulative probability) = 0.999. The number of necessary samples will evidently grow if we assume lesser probability of success, while guaranteeing the same level of confidence in our experiment/trial. In general, for a large number of samples collected, say, 10,000, we need to calculate the relevant quantiles of the binomial distribution https://keisan.casio.com/exec/system/1180573200. So, for the cumulative distribution equal to 0.999, with number of trials equal to 10,000 and the probability of success to be 0.99, we expect at least 9,868 sampled distributions representing the normal state of the log source which should be identified.

3.2 Algorithms and Experiments

As we mentioned in Section 1 and 2, when performing anomaly detection and other important tasks for a log source using LI, we face the problem of having a baseline model for the source as a typical characteristic of its historically normal behavior. More specifically, if the streams' current distribution of event types is largely deviating (as a matter of a distance measure) from the baseline, we can automatically raise an alert to the system administrator. We have already shown in [4] that in tasks such as anomaly, change, and sickness detection, the event types are invaluable "signatures" or "fingerprints" of log sources to rely on.

In this subsection, we describe our ML algorithms implemented in Python for identification of that baseline structure using LI's event types with random sampling. We describe two methods to perform such a learning task. The first method applies random sampling of log messages with confidence control of the inference. The second algorithm indicates the most generic and sophisticated solution to the problem although with much higher complexity.

Method I (with random sampling). How to identify the baseline distribution with the sampled 5 distributions in the example in subsection 3.1? The next question is then how to identify those 4 dominant (in terms of characterizing the state of the source) distributions out of 5?

Our solutions below indicate how to choose the baseline event type distribution and the related *normal discrepancy radius* of the stream that quantifies the tolerable "distances" of the observed event type distributions from this baseline as still within the expected behavior:

1. compute cosine similarity distances between all pairs of event type distributions (histograms) derived for each of sampled log portion;
2. compute average cosine similarity distance (ACSD) for each sample histogram from the rest;
3. rank sampled histograms in decreasing order of their ACSDs and pick up the top 4 (tries to identify the most similar subset of 4 distributions.);
4. pick up the histogram with minimum ACSD as the baseline (centroid) distribution;
5. if there are several histograms with min ACSD, compute Shannon entropy of those and pick up the one with maximum entropy value as a baseline distribution.

In an alternative implementation, the step 4 can be replaced with the maximum entropy principle applied to the top distributions directly to identify the most unbiased baseline distribution.

Shannon entropy [6] measures the uncertainty in a random variable defined by

$$H(P, a) = -\sum p_i \log_a p_i \leq \log_a n$$

and its binary version's plot is depicted in Fig. 3.

For a demonstration purposes, we performed a small experiment on an Apache server (consisting of web, email, and ftp services), a similar experiment described in [4] for sickness detection task within a population of peers. We emulated a stress or security attack (using ApacheBench test tool) on the web host with a high-rate service requests for a 5-minute duration, after observing it in a "normal" operational workload for half an hour. Then we sampled 5 different five log portions of 5-minute length that captured the stressed window (Sample 2) as well. For each of log portions (Samples 1-5 shortened to S1-S5) we computed the probability distributions of observed 25 event types within, which are shown in Table 1.

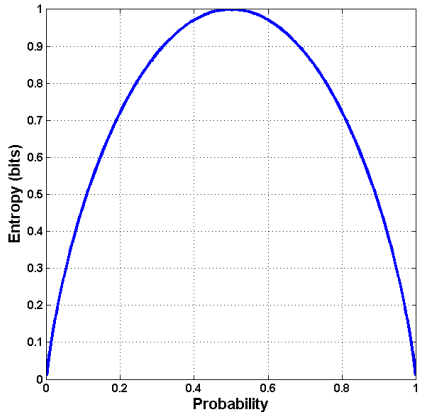

Fig. 3. Shannon entropy function for binary distribution and log base $a = 2$ with the maximum uncertainty (1) at probability = 0.5.

Then computing the ACSDs for each distribution from the rest, we get:

$$ACSD(S1) = 0.97,\ ACSD(S2) = 0.89,$$
$$ACSD(S3) = 0.96,\ ACSD(S4) = 0.97,\ ACSD(S5) = 0.97.$$

Table I. Five samples of log event types taken from a host for a 5-minute time range each.

	Probabilities				
LI's Event Types	**S1**	**S2**	**S3**	**S4**	**S5**
v4_18ca9254	0.03	**0.02**	**0.03**	0.03	0.03
v4_1a6ac047	0	**0**	**0**	0	0
v4_28077ade	0	**0**	**0**	0	0
v4_2f6e41a2	0.03	**0.02**	**0.03**	0.03	0.03
v4_36c81ef6	0.05	**0.04**	**0.07**	0.05	0.06
v4_393c8071	0.2	**0.17**	**0.18**	0.21	0.19
v4_59cd0174	0.03	**0.02**	**0.03**	0.03	0.03
v4_681f6046	0.05	**0.04**	**0.07**	0.05	0.06
v4_69475cc1	0	**0.08**	**0**	0	0
v4_6dd466a5	0.03	**0.02**	**0.03**	0.03	0.03
v4_71183f87	0	**0**	**0**	0	0
v4_802bd0d4	0.11	**0.1**	**0.1**	0.12	0.11
v4_87e0ca23	0.03	**0.02**	**0.03**	0.03	0.03
v4_88de5e12	0.03	**0.02**	**0.03**	0.03	0.03
v4_8ebbb638	0.03	**0.02**	**0.03**	0.03	0.03
v4_94680e71	0.03	**0.02**	**0.03**	0.03	0.03
v4_9d3e7bdd	0.03	**0.02**	**0.03**	0.03	0.03
v4_9fd2eafd	0.04	**0.11**	**0.04**	0.03	0.04
v4_a7f56e13	0.06	**0.05**	**0.05**	0.06	0.06
v4_a8a71825	0.03	**0.02**	**0.03**	0.03	0.03
v4_b610f232	0.03	**0.02**	**0.03**	0.03	0.03
v4_b9100c8f	0.05	**0.04**	**0.07**	0.05	0.06
v4_bafd4270	0.03	**0.02**	**0.03**	0.03	0.03
v4_bfebb8d	0.05	**0.04**	**0.07**	0.05	0.06
v4_f0533255	0.03	**0.02**	**0.03**	0.03	0.03

By ranking (step 4) samples/distributions in decreasing order of the distance measure, we pick up the following four having highest average similarity (this is the dominant similarity set representing the normal workload mode of the host):

$$ACSD(S1) = 0.97,\ ACSD(S3) = 0.96,\ ACSD(S4) = 0.97,\ ACSD(S5) = 0.97.$$

The chosen four distributions are the baseline "candidates". With this ranking, the anomalous Sample 2 indicated in red in Table I dropped from the "candidates" list.

Since there are three samples with the same score, we are going to identify the one with maximum uncertainty as the "safest" unbiased model for the baseline distribution. The entropies (in *nats*, $a = 10$) of sample distributions are:

$$H(S1) = 2.83,$$
$$H(S2) = 2.78,$$
$$H(S3) = 2.85,$$
$$H(S4) = 2.80,$$
$$H(S5) = 2.84.$$

And what is the best "candidate" to be the baseline? Applying the maximum entropy principle [10], it is the distribution that has the highest information uncertainty. Sample 3 (green column in Table I) has the maximum entropy distribution (Fig. 4) and is chosen to be the baseline for the log source.

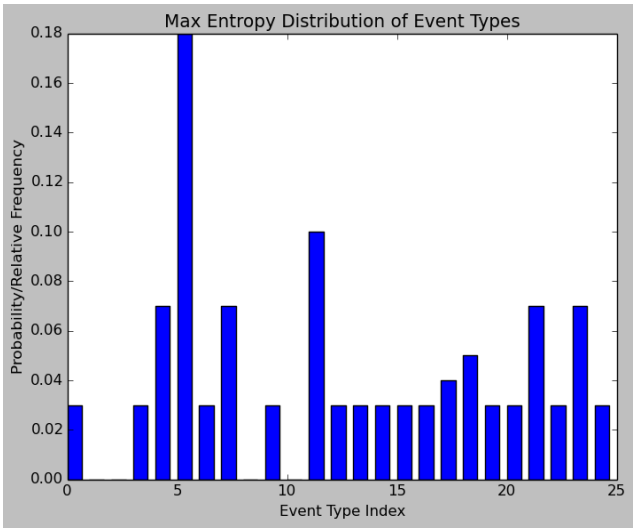

Fig. 4. Maximum entropy distribution as a learned baseline.

The final step is to derive the "normal discrepancy radius" of the baseline/source. This is the variance in ACSD that we observe in the top similarity subset of 4.

The *normal discrepancy radius* (R) then can be defined as the range of ACSD's in the "dominant" similarity set. In our experimental example, it is the following difference:

$$R = 0.97 - 0.96 = 0.01.$$

Method II (clustering of event type distributions without random sampling). In an alternative, highly complex implementation, without randomized sampling, with continuously measured and stored probability distributions/histograms of event types

that contain all normal and abnormal patterns in the log stream, our algorithm performs the following steps:

1. calculate local outlier factors of all histograms using LOF algorithm [9];
2. pick up the “centroid” event type distribution as the histogram having the smallest LOF;
3. if those are several, the one that has the highest entropy;
4. compute the average and standard deviation of distances of all histograms from the centroid;
5. define the “normal discrepancy radius” of the log source from its baseline with 3-sigma rule, assuming that distances are distributed normally [11].

LOF is based on a concept of a local density (or similarity distance in our case), locality defined by k nearest (similar) neighbors. Those neighbors are employed to estimate the density. Based on this evaluation, outliers are detected as those distributions that have substantially lower density than their neighbors. The local density is estimated by the distance at which a point can be "reached" from its neighbors [9]. In a simpler implementation, the centroid can be the histogram having the minimum average distance from the rest of the histograms. The normalcy radius can be also linked to Chebyshev’s inequality [12] without making the assumption of normality.

Fig. 5 pictorially supports the main ideas of Methods I and II.

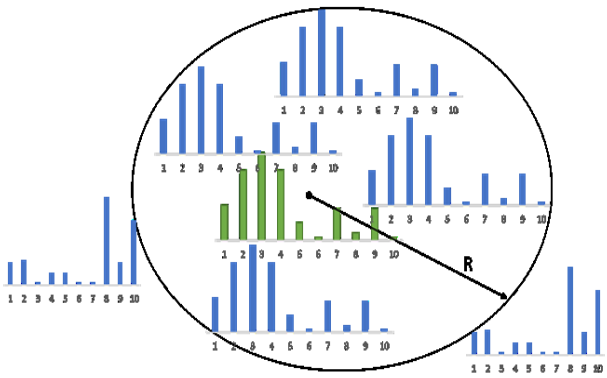

Fig. 5. Illustrating Baseline as a *Centroid Distribution* of the cluster and its *Normal Discrepancy Radius*.

Abnormality degree of baseline violation. We can measure the abnormality degree or criticality of alerts raised using the baseline. It is defined by how far is the run-time event type distribution from the centroid minus the discrepancy radius. So, if the normal operations are within 0.1-radius, a run-time distribution having distance=0.4 exceeds the tolerance by 0.3. Then we can use a scale to map those over-tolerance distances into a score range from “lowest” to “highest” (continuous or discrete). Alternatively, measure and communicate to the user how many times or by which percent the tolerance radius is exceeded.

It can happen that the event source operates in different modes and have different baseline characteristics accordingly (for example, the corresponding IT resource has

high and low utilization modes). In such cases, k-means clustering [13] can be applied to derive relevant clusters and their centroids, and apply the above-mentioned normality assumption or Chebyshev inequality to extract the normal discrepancy radius for each of the clusters. This means that for the corresponding anomaly detection we identify in which mode the system operates and apply the relevant centroid baseline.

We conducted another controlled experiment as in [4], again choosing vR LI as our proof-of-concept application. We monitored LI's logs in INFO and DEBUG modes using another LI instance. This is a way to observe the application in two different stress modes (high and low). In those logging modes, our algorithm implemented in Python observes different event type distributions. Representative distributions are depicted in Figs 6 and 7, respectively.

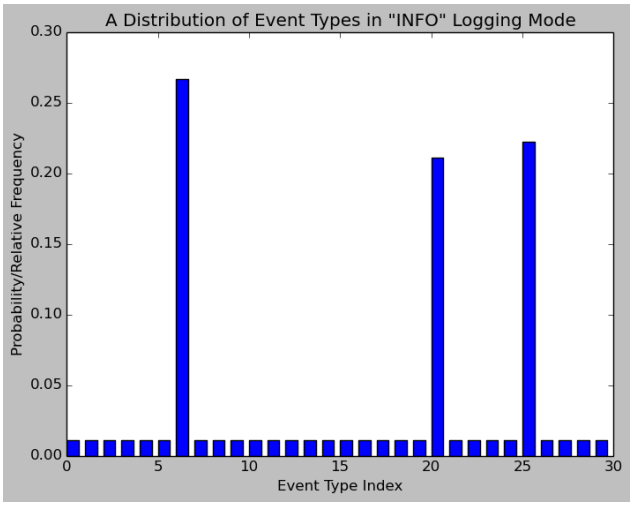

Fig. 6. Representative distribution in INFO logging mode of LI.

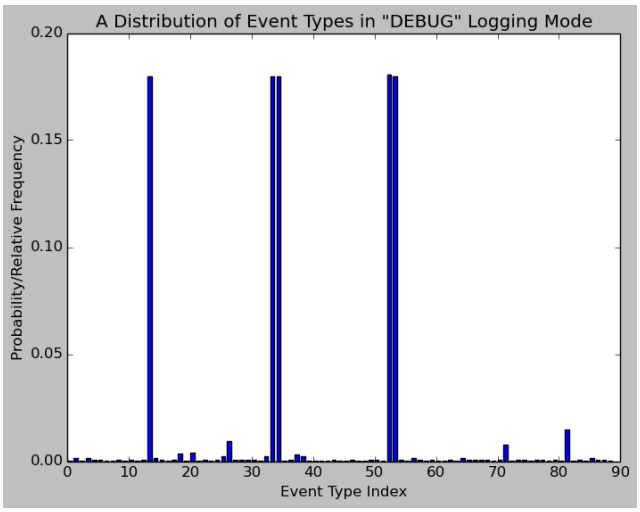

Fig. 7. Representative distribution in DEBUG logging mode of LI.

The graphs illustrate different number and rates of event types. Therefore, having detected possible workload modes of an application with complex clustering methods, then the sampling technique of Method I can be applied to derive the baseline for each mode individually.

4 Future Work

We plan to experiment with much bigger data sets. Particularly, within a large experimental setup we are going to learn baselines of ESXi hosts in various environments by applying our sampling concepts with confidence control in Method I and employ the learned structures in real-time anomaly detection. Then we'll be able to validate observed anomalies with the incident data regularly being reported by operations teams.

The live never-ending stream of ESXi events in large cloud infrastructures make a big volume. This means that even we observe a large number of different event types and hence deal with storage of large-size histograms, with the sampling techniques based on binomial distributions, the number of those histograms will be moderate. Therefore, feasibility of our implementation is not under risk because of algorithmic complexity.

5 Conclusion

We introduced the concept of *baseline models* for log sources employing LI's native constructs of Event Types and described algorithms to efficiently identify those structures using statistical sampling and machine learning. Baseline models are comprehensive for various analytical tasks in log management from real-time anomaly and change detection to other pattern detection problems. With controlled experiments, we demonstrated how the baselines are learned.

References

1. VMware vRealize Operations Manager: http://www.vmware.com/products/vrealize-operations.html.
2. VMware vRealize Log Insight: https://www.vmware.com/products/vrealize-log-insight.
3. Marvasti, M.A., Poghosyan, A.V., Harutyunyan, A.N., and Grigoryan, N.M.: An enterprise dynamic thresholding system. In: Proceedings of USENIX International Conference on Autonomic Computing (ICAC), pp. 129-135, June 18-20, Philadelphia, PA, USA (2014).
4. Harutyunyan, A.N., Poghosyan, A.V., Grigoryan, N.M., Kushmerick, N., and Beybutyan, H.: Identifying changed or sick resources from logs. Submitted to 4th International Workshop on Data-Driven Self-Regulating Systems (DSS 2018), September 7, Trento, Italy (2018).

5. Harutyunyan, A.N., Poghosyan, A.V., Grigoryan, N.M., and Marvasti, M.A.: Abnormality analysis of streamed log data. In: Proceedings of IEEE Network Operations and Management Symposium (NOMS), 7p., May 5-9, Krakow, Poland (2014).
6. Cover, T. and Thomas J.: Elements of Information Theory. Wiley, (1991).
7. Brown, D. and Kushmerick, N.: Anomaly detection using log summary divergence. A method for computing the similarity of two log message queries and its application in anomaly detection in distributed environments. Technical paper, VMware (2015).
8. Binomial Distribution: https://en.wikipedia.org/wiki/Binomial_distribution.
9. Breunig, M.M., Kriegel, H.-P., Ng, R.T., and Sander, J.: LOF: Identifying density-based local outliers. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 93–104, May 15-18, Dallas, TX, USA (2000).
10. Maximum Entropy Principle: https://en.wikipedia.org/wiki/Principle_of_maximum_entropy.
11. Normal Distribution: https://en.wikipedia.org/wiki/Normal_distribution.
12. Chebyshev inequality: https://en.wikipedia.org/wiki/Chebyshev's_inequality.
13. K-means clustering: https://en.wikipedia.org/wiki/K-means_clustering.