Spatio-temporal Factorization of Log Data for Understanding Network Events

Tatsuaki Kimura*, Keisuke Ishibashi*§, Tatsuya Mori†, Hiroshi Sawada‡§, Tsuyoshi Toyono*,
Ken Nishimatsu*, Akio Watanabe*, Akihiro Shimoda* and Kohei Shiomoto*
*NTT Network Technology Laboratories, NTT Corporation, Tokyo 180-8585, Japan
†School of Fundamental Science and Engineering, Waseda University, Tokyo 169-8555, Japan
‡NTT Service Evolution Laboratories, NTT Corporation, Kanagawa 239-0847, Japan
§NTT Machine Learning and Data Science Center, NTT Corporation, Kyoto 610-0237, Japan

***Abstract*—Understanding the impacts and patterns of network events such as link flaps or hardware errors is crucial for diagnosing network anomalies. In large production networks, analyzing the log messages that record network events has become a challenging task due to the following two reasons. First, the log messages are composed of unstructured text messages generated by vendor-specific rules. Second, network equipment such as routers, switches, and RADIUS severs generate various log messages induced by network events that span across several geographical locations, network layers, protocols, and services. In this paper, we have tackled these obstacles by building two novel techniques: statistical template extraction (STE) and log tensor factorization (LTF). STE leverages a statistical clustering technique to automatically extract primary *templates* from unstructured log messages. LTF aims to build a *statistical model* that captures spatial-temporal patterns of log messages. Such spatial-temporal patterns provide useful insights into understanding the impacts and root cause of *hidden network events*. This paper first formulates our problem in a mathematical way. We then validate our techniques using massive amount of network log messages collected from a large operating network. We also demonstrate several case studies that validate the usefulness of our technique.**

I. Introduction

With the increase in the number of network elements and services (e.g. IPTV, VoIP), it has become extremely difficult to understand the impacts and patterns of network events such as link flaps or hardware errors. The diversity of services complicates the interactions of events, a common link down event leads not only to neighboring nodes going down, but may cause virtual path disconnections or cause related services to go down. Typically, a Network Management System (NMS), e.g. [1]–[4], monitors network health to deal with the enormous network events that occur on a daily basis by using Simple Network Management Protocol (SNMP) traps or syslog triggers. Predefined rules in NMSs are such that alarms represent apparently critical states. In today's networks, on the other hand, operators often care about temporally abnormal events, e.g. whether customers can access the service even though its quality is low. In addition, NMS alarms are so obscure that they fail to capture the impacts or structures of events such as the network layer, protocol, service dependencies. However, understanding such complicated structures of network events is crucial for caring out a detailed diagnosis of network anomalies.

978-1-4799-3360-0/14/$31.00 © 2014 IEEE

Network logs, including syslogs and alarm messages reported by the NMS, contain some of the most important information for diagnosing network anomalies. Network operators usually start their troubleshooting processes with alarm messages. Syslog messages are often used for more complex drill-down processes, in which operators check the root cause of problems and decide what they should do to recover from the failure. However, despite the usefulness of network logs, they are not fully utilized in large production networks for two reasons: (i) log messages are composed of *unstructured* text messages generated by vendor-specific rules. Since a large network consists of multi-vendor elements, formats of logs are highly diverse. (ii) Log data is very *complex* because network equipment such as routers, switches, and RADIUS severs generate various log messages induced by underlying network events that span across several geographical locations, network layers, protocols, and services.

A simple example that illustrates the complexity of log messages is shown in Fig. 1. In this example, a set of link down/up log messages is generated by a link flap event, which can be differentiated from the original link down event. Note that the same log message pattern appears in different hosts with the layer 2 relationship. In addition to this, during the link flap event, other events (e.g. a cron job in the example of Fig. 1) may happen in the network. Thus, log data can be considered to be a *superposition* of the spatial and temporal patterns caused by underlying network events. Since various events occur from hour to hour throughout the entire network, extracting these spatial-temporal patterns of log messages from unstructured and complex log data is far more challenging.

Understanding the spatial-temporal patterns of log messages provides useful insights into the network anomaly diagnosis. For instance, it enables us to correlate multiple link flap events that occur at different equipment each working in different network layers. Such analysis is crucial in understanding the root cause and impacts of the network anomaly. It also enables us to extract *hidden network events* that could lead to network failures. The extracted information can be used for preventive maintenance operation.

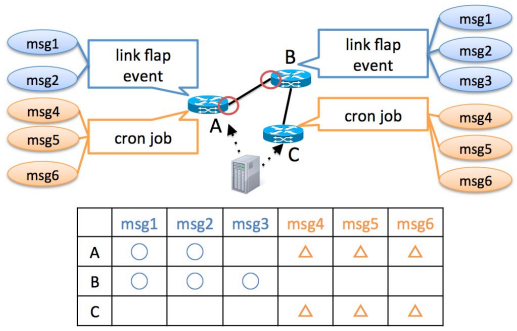

Fig. 1. Log data can be considered as a *superposition* of log message patterns induced by network events; in this example, we observe messages associated with both link flap (msg1–3) event and cron job (msg4–6).

This paper presents a novel *spatio-temporal* factorization technique, which automatically learns and identifies spatial and temporal patterns of log messages from unstructured and complex log data without relying on any domain knowledge. Here, *spatio* refers to relationships among hosts in the network. To achieve our goal, we first develop a **Statistical Template Extraction (STE)** method that automatically converts unformatted log data to primary *templates* using a statistical clustering algorithm. We then mathematically formulate the pattern of log messages by regarding it as a rank-3 tensor (a *template*, a host, and a timestamp). To build a *statistical model* that captures spatial-temporal patterns of log messages, we develop **Log Tensor Factorization (LTF)** based on Nonnegative Matrix Factorization (NMF) [14] and Nonnegative Tensor Factorization (NTF) [8], which are powerful techniques for decomposing a mixture of nonnegative elements into raw features in unsupervised manner. By using LTF, we can obtain spatial and temporal patterns of log message groups, which can be considered to be caused by the *underlying network events* according to our observation. We successfully model both the spatial and temporal relationships, which cannot be obtained by existing approaches because they are insufficient to model these multiple relationships. To the best of our knowledge, this formulation is the first attempt to analyze log data. Further, despite the special features of LTF, we derive a simple algorithm for obtaining the solution. Finally, we describe our validation of the proposed method using a massive number of data sets collected from a large scale operating network. Experimental results show that our techniques successfully fit the spatial-temporal patterns of network events. We also demonstrate several case studies that validate the usefulness of our technique.

We briefly summarize our contributions as follows:

- We developed a statistical template extraction method, STE, that automatically extracts primary templates from massive and unstructured network logs without using any prior information about log format.
- We developed a novel tensor factorization method, LTF, that enables us to model spatial-temporal patterns of log messages from the templates extracted by STE. We also derived a simple algorithm for LTF.
- We validated LTF fits well the spatial-temporal patterns of network events through the extensive analysis of measurement data collected at a large operating network. We also demonstrated case studies that clearly show the usefulness of LTF.

The rest of this paper is organized as follows. Section II summarizes the works related to our research. In Section III, we explain network logs, log templates and the definition of network events to mathematically formulate our problem. Section IV provides our main result, the log tensor factorization. In Section V, we conduct several validations for the proposed method and present case studies in Section VI. Finally, Section VII concludes our paper.

II. Related Work

There are many commercial products that aim to increase the efficiency of network management and operations [1]–[5]. Specifically, Splunk [5] is a log analytics platform that collects machine log data in real time and helps achieve fast search and analysis. Although NMSs [1]–[4] have a root cause analysis (RCA) function, they cannot identify detailed network events and require domain knowledge.

A lot of research has also been done in the areas of network fault diagnosis and fault localization by using various network data in enterprise network. Yamanishi et al. [26] proposed a technique to detect system failure from server syslog using a mixture of Hidden Markov Models. SCORE [13] and Shrink [10] are both fault localization systems using a bipartite graph that models relationships among network elements on the same SRLG (shared risk link group). Sherlock [6] is an RCA system that learns a dependency graph between multilevel resources in enterprise networks. Orion [7] extracts the causal relationship among application traffic by analyzing each traffic delay. eXpose [11] reveals flow generation rules by learning the association rules of flows from packet trace data while our objective is modeling and learning the patterns of network log messages. NetMedic [12] monitors acts of applications and conducts detailed diagnoses of fault events occurring in enterprise. Meta [21] and TAR [22] are both fault localization systems, which automatically learn fault events from the event data sets and find the root causes of faults rapidly by indexing the network events. However, they used the network events data collected by the NMS, whereas our scope is to extract such events from raw primitive log data.

Additionally, various studies that use router syslog and SNMP messages have been done for large ISP networks. NICE [16] extracts the statistical correlation between temporal network events with time lags. G-RCA [27] is an RCA system that identifies the root cause of the problem by matching a current event to a predefined decision tree, which represents the causal relationships of the network events. MERCURY [17] and PRISM [18] detect maintenance induced performance changes using change point analysis and PCA-based subspace algorithm. SyslogDigest [19] is the most closely related work to our research. It constructs digest information from router

TABLE I
EXAMPLES OF LOG MESSAGES

#	syslog
1	%LINK-3-UPDOWN: Interface FastEthernet 0/9, changed state to down
2	%SYS-5-CONFIG_I: Configured from console by vty2 (10.11.XXX.YYY)
3	System: Interface ethernet XXX, state down
#	alarm
1	Interface down: IF: XXX IFIndex: YYY IFStatus: ZZZ
2	Node down: ping timeout (0 / 5)

TABLE II
EXAMPLES OF LOG TEMPLATES

#	syslog
1	%LINK-3-UPDOWN: Interface FastEthernet *, changed state to down
2	%SYS-5-CONFIG_I: Configured from console by * (*)
3	System: Interface ethernet *, state down
#	alarm
1	Interface down: IF: * IFIndex: * IFStatus: *
2	Node down: ping timeout (* / *)

syslogs by extracting the templates of logs and grouping them within relevant routers. Since they do not take into account the fact that log data is a superposition of message patterns, they require domain knowledge for extracting the spatial relationship.

III. NETWORK LOG, TEMPLATE AND NETWORK EVENT

In this section, we explain an overview of network log data and a preprocessing technique for modeling the patterns of log messages, called STE. We then define the network events to mathematically formulate the LTF problem.

Network logs such as router syslog and alarms captured by NMS include various information: network faults, security issues, and console logs. In general, they consist of three parts: a timestamp, a host name (or IP address) and a message. Examples of the message parts of network logs are given in Table I. Although their format depends on the service type or the vendor of each instrument, log messages consist of short readable texts. Therefore, common structures can be found among them; there are parts that represent event types or entities (`%LINK-3-UPDOWN` and `System`), parts that show changes of states (down/up), and parameter parts (IP address, host name, and process ID).

A. Statistical Template Extraction

Because of the variety and large volume of logs, it is unrealistic to extract patterns from raw log messages. For example, we can easily find a strong correlation between `%LINK-3-UPDOWN` events and `%LINEPROTO-5-UPDOWN` events because layer 2 down/up events often induce layer 1 down/up events. However, to extract such a correlation, we need a sufficient number of samples of layer 1 and 2 down/up event co-occurrence at the same interface and link. Therefore, our objective here is to obtain correlation among not *raw log messages*, but *log templates*, i.e. messages without parameters. Examples of log templates are listed in Table. II, where parameters are replaced by "*" and each log template corresponds to the log message with the same ID in Table. I.

Although log templates may be obtained from vendor support pages or manuals, these templates may change due to software updates or operational issues. In addition, some of them may not be opened publicly. Therefore, we developed a method of statistical template extraction (STE) from large scale log messages without relying on any prior information about log format. Log parsing technique has been reported in other studies [19], [25]. Specifically, Qiu et al. [19] presented the template extraction from router syslogs. They selected a similar approach to the spam detection by constructing a signature tree whose root is the message type of the log. However, it is difficult to determine the root of the tree in the case of general logs. Thus, STE takes another approach using the following features of log messages: parameters appear less frequently than template words; and messages have similar structures with the *positions* of words, among those generated from the same templates. STE consists of two parts: (1) statistical word scoring and (2) score clustering.

1) Word scoring: We assume words are separated by white space in each message. In general, template words tend to appear in the same position in messages that have the same words length. From this observation, we score the tendency of words to be a template word as follows: if a $word$ appears in the p-th position in a log message that contains Len words, then the score for the $word$ is defined as

$$Score(word, p, Len) = \mathsf{P}(word \mid p, Len).$$

From the above definition, the score for the template words is greater than the parameters.

2) Score clustering: Next, we need to determine whether word is a parameter or template word. A simple approach is to set a threshold for the score; however, such an approach would fail because the range of scores depends on the feature space with the same (p, Len). To avoid this, we take a clustering approach for scores and divide all the words in each message into template words and parameters. The clustering technique we use here is DBSCAN (Density Based Spatial Clustering) [9], which makes clusters so that the distance between clusters is greater than some threshold δ. We then finally obtain a template by choosing the top clusters so that the number of words is greater than $\beta \times Len$, where $0 < \beta < 1$.

B. Definition of Network Events

We now give the definition of network events using log templates. This definition will be used for LTF formulation. Observations of log data indicate that there is a log template group that is likely to co-occur, and it represents the state of individual network elements. Further, underlying network events may cause log template groups to influence multiple hosts (see Fig 1). Therefore, we consider such correlated log messages with spatial impacts as network events. We first define the log template group and then define the network events. In what follows, we assume log templates for all log messages are appropriately extracted and that all host names

and time stamps of log messages are also known. In addition, we partition input log data by time-windows. Further, we write $i \in \mathcal{I}$, $h \in \mathcal{H}$, and $j \in \mathcal{J}$ to represent templates, hosts, and time-windows, where $\mathcal{I}$, $\mathcal{H}$, and $\mathcal{J}$ are the total sets of i, h, and j, respectively.

Definition 1 (Template Group) A log template group $l \in \mathcal{L}$ is defined as a set of templates that tends to co-occur at a single host, where $\mathcal{L}$ represents the total set of template groups.

A template group represents an event at an individual host, e.g., router reboot, link flap, or configuration change. Note that for a fixed template, its multiple allocation to different template groups is allowed. The reason for this is that some template groups may be a subset of other template groups. For instance, although link flap is a combination of link down and up templates occurring at the same time-window, each link down/up message can appear separately.

Definition 2 (Network Event) A network event e is defined as tuple sets of a host and a template group that tend to co-occur, i.e. $e = \{(h_1, l_1), (h_2, l_2), \dots\}$ $(h_1, h_2, \cdots \in \mathcal{H},\ l_1, l_2, \cdots \in \mathcal{L})$.

From the definition, network events are considered to be a spatial extension of template groups. For example, with a link flap, if a link flap occurs at some host, then similar log groups appear at its neighbor hosts. Network events are described as a mixture of such relationships and the template groups.

IV. Log Tensor Factorization

Our main contribution in this paper is modeling the the patterns of network log messages and mathematically formulating the extraction of network events from logs. The first key idea for the event extraction is to regard network log data as a rank-3 tensor (log templates, hosts, and time-windows), and to extract template groups and host relationships that are likely to co-occur in log data. On the basis of this idea, we take a nonnegative tensor factorization approach, in which we *directly* model and learn the template group and the network events defined in Section III-B. Our tensor factorization problem is closely related to NMF [14] and NTF [8], which are powerful machine learning techniques for decomposing a mixture component to nonnegative features. In addition, they are widely applied to various areas such as audio processing [20], text mining [23], image analysis [14] and brain signal analysis [8]. However, these classical approaches do not fit well to our problem because they cannot model both the template groups and the spatial relationships (discussed in detail in Section IV-B). Therefore, we developed a novel tensor factorization, LTF, and derived simple update rules for it. We first introduce a definition of tensor expression of logs.

Definition 3 (Log Tensor) For a log template $i \in \mathcal{I}$, a host $h \in \mathcal{H}$, and a time-window $j \in \mathcal{J}$, let $x_{ihj} \in \{0, 1, \dots\}$ denote the number of occurrences of a log message. A *log tensor* $\boldsymbol{X}$ is defined as a rank-3 tensor with the dimension $I \times H \times J$ whose (i, h, j)-th element is x_{ihj}, where $I = |\mathcal{I}|$, $H = |\mathcal{H}|$ and $J = |\mathcal{J}|$, respectively.

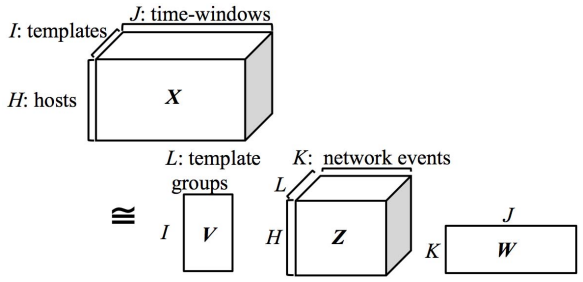

Fig. 2. Concept image of LTF. The target tensor $\boldsymbol{X}$ is factorized into an $I \times L$ matrix $\boldsymbol{V}$, an $L \times K \times H$ rank-3 tensor $\boldsymbol{Z}$, and a $K \times J$ matrix $\boldsymbol{W}$.

Our key insight for modeling the log is to regard log data in each time-windows as a hierarchical superposition, i.e. log data is a mixture of network events, which also consist of template groups. According to this observation, we rewrite the network event extraction problem from log data as the following tensor factorization problem.

Problem (LTF: Log Tensor Factorization) For a given tensor $\boldsymbol{X}$ with the dimension $I \times H \times J$ and integers K and L, the log tensor factorization problem is to find a factorization such that

$$\boldsymbol{X} \simeq \sum_{k=1}^{K} \left[\sum_{l=1}^{L} \boldsymbol{v}_l \otimes \boldsymbol{z}_{lk} \right] \otimes \boldsymbol{w}_k, \tag{1}$$

where for $l = 1, \dots, L$ and $k = 1, \dots, K$, $\boldsymbol{v}_l = [v_{il}]$, $\boldsymbol{z}_{lk} = [z_{lkh}]$ and $\boldsymbol{w}_k = [w_{kj}]$ are nonnegative vectors with the dimensions I, H, and J, respectively.

For convenience, we define an $I \times L$ matrix $\boldsymbol{V}$ as $\boldsymbol{V} = [v_{il}]$, an $L \times K \times H$ tensor $\boldsymbol{Z}$ as $\boldsymbol{Z} = [z_{lkh}]$, and a $K \times J$ matrix $\boldsymbol{W}$ as $\boldsymbol{W} = [w_{kj}]$. A conceptual image of LTF is shown in Fig. 2. In what follows, we give intuitive interpretations for $\boldsymbol{V}$, $\boldsymbol{Z}$, and $\boldsymbol{W}$. First, matrix $\boldsymbol{V}$ can be considered as a *template group matrix*. A visualized image of $\boldsymbol{V}$ is shown in Fig. 3. If a template i belongs to a template group l, then $v_{il} > 0$, and otherwise $v_{il} = 0$, i.e., each vector $\boldsymbol{v}_l$ corresponds to the l-th template group, and L represents the number of template groups. Second, rank-3 tensor $\boldsymbol{Z}$ can be regarded as a *network event tensor*, and we call each $H \times L$ matrix $\boldsymbol{z}_k$ an *event slice*. A visualized image of $\boldsymbol{Z}$ is shown in Fig. 4. In each event slice, if a template group l at host h belongs to the k-th network event, then $z_{lkh} > 0$. In other words, the k-th event slice $\boldsymbol{z}_k$ corresponds to k-th network events and K is equal to the total number of network events. Finally, matrix $\boldsymbol{W}$ can be considered as a *weight matrix*. Each element w_{kj} represents a weight that the k-th network event takes in time-window j. In short, we can understand when the k-th network event has occurred by observing $\boldsymbol{w}_k$.

Summarizing the above, we can interpret the LTF problem as follows: (i) each $\boldsymbol{v}_l$ corresponds to template groups; (ii) the event slice $\boldsymbol{z}_k$ represents the network events, i.e. which hosts

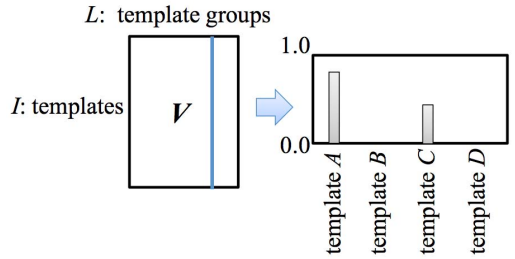

Fig. 3. Visualized image of matrix V. The l-th vector of $\boldsymbol{V}$, $\boldsymbol{v}_l$, represents in each element whether a template i belongs to a template group l or not. In this example, there are spikes at templates A and C, thus, they belong to the same template group.

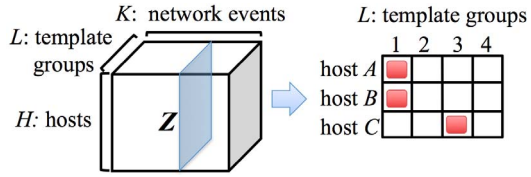

Fig. 4. Visualized image of tensor Z. The k-th event slice of $\boldsymbol{Z}$, $\boldsymbol{z}_k$, represents k-th network events. If template group l at host h is the k-th network event, then we can see $z_{lkh} > 0$.

take a template group l; and (iii) $\boldsymbol{X}$ can be considered as a superposition of event slices with weights $\{\boldsymbol{w}_k\}$.

A. Multiplicative Update Rules

We now develop an algorithm to calculate output tensors of the LTF problem. As mentioned before, the LTF problem is an extension of the NMF or NTF problems, which have simple iteration algorithms for obtaining the solutions; these algorithm are known as *multiplicative update rules* [8], [14]. On the basis of their derivation, despite the special features of our model, we can derive very simple update rules for LTF via an auxiliary function approach.

First, the LTF problem can be mathematically formulated as the following minimization problem:

$$\begin{aligned} &\min_{\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W}} \mathcal{D}(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W}), \\ &\text{s.t.}\ \ \boldsymbol{V},\boldsymbol{Z},\boldsymbol{W} \geq \boldsymbol{O},\ \sum_i v_{il} = 1, \sum_{l,h} z_{lkh} = 1, \end{aligned} \tag{2}$$

where $\mathcal{D}(\cdot)$ represents the KL-divergence function and serves as a cost function. More specifically,

$$\begin{aligned} &\mathcal{D}(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W}) \\ &= \sum_{i,h,j} x_{ihj} \log \frac{x_{ihj}}{\sum_{k,l} v_{il} z_{lkh} w_{kj}} - x_{ihj} + \sum_{k,l} v_{il} z_{lkh} w_{kj}. \end{aligned} \tag{3}$$

Note that the second and third equalities in (2) are normalization conditions. We then have the multiplicative update rules for optimization problem (2) as follows: for $i \in \mathcal{I}$, $h \in \mathcal{H}$ and $j \in \mathcal{J}$,

$$v_{il} := \frac{\sum_{h,j,k} \frac{\breve{z}_{lkh}\breve{w}_{kj}}{\sum_{k',l'} \breve{v}_{il'}\breve{z}_{l'k'h}\breve{w}_{k'j}} \cdot x_{ihj}}{\sum_{h,j,k} z_{lkh} w_{kj}} \breve{v}_{il}, \tag{4}$$

$$z_{lkh} := \frac{\sum_{i,j} \frac{\breve{v}_{il}\breve{w}_{kj}}{\sum_{k',l'} \breve{v}_{il'}\breve{z}_{l'k'h}\breve{w}_{k'j}} \cdot x_{ihj}}{\sum_{i,j} v_{il} w_{kj}} \breve{z}_{lkh}, \tag{5}$$

$$h_{kj} := \frac{\sum_{i,h,l} \frac{\breve{v}_{il}\breve{z}_{lkh}}{\sum_{k',l'} \breve{v}_{il'}\breve{z}_{l'k'h}\breve{w}_{k'j}} \cdot x_{ihj}}{\sum_{i,h,l} v_{il} z_{lkh}} \breve{w}_{kj}, \tag{6}$$

where $\breve{v}_{il}$, $\breve{z}_{lkh}$, and $\breve{w}_{kj}$ represent older values in each iteration. After each update, it is necessary to normalize $\boldsymbol{V}$ and $\boldsymbol{Z}$ with respect to the conditions in (2). We give complete proofs for deriving (4)–(6) in the Appendix.

B. NTF and LTF

We here note the difference between the conventional NTF problem and the LTF problem. Given a rank-3 tensor $\boldsymbol{X}$, the NTF problem is to find a factorization such that

$$\boldsymbol{X} \simeq \sum_{k=1}^{K} \boldsymbol{v}'_k \otimes \boldsymbol{z}'_k \otimes \boldsymbol{w}'_k,$$

where $\boldsymbol{v}'_k$, $\boldsymbol{z}'_k$, and $\boldsymbol{w}'_k$ are rank-1 tensors with the dimensions I, H, and J, respectively. Intuitively, the NTF finds the correlated patterns in the log tensor $\boldsymbol{X}$ and describes it with the sum of the tensor product of K rank-1 tensors. In coordination with the expression of the LTF, each $\boldsymbol{v}'_k \otimes \boldsymbol{z}'_k$ can be considered as an insufficient expression of the network events: when we interpret $\boldsymbol{v}'_k$ as a log template group, the corresponding tensor product $\boldsymbol{v}'_k \otimes \boldsymbol{z}'_k$ represents that the k-th template group occurs at the hosts in $\boldsymbol{z}'_k$; however, this is different from Definition 2. According to our observation, a certain log template group may appear in different host groups; therefore, the number of template groups and of network events should be different ($L < K$, in general). Consequently, we can say that LTF is an extension of the normal NTF in the sense that the NTF can model only groups of a set <log template, host> that tend to co-occur. By contrast, the LTF models the template groups that appear in these groups.

C. Noise Filter

There are long running logs that induce spurious correlations, such as the firewall logs. In addition, these logs increase computational costs. Therefore, we apply a *noise filter* to the data that remove sets of a host and a template that occur with high frequency. We define the frequency for template i and host h as a fraction of the time-windows in which a tuple $(i,\ h)$ is observed. Specifically, we choose 0.01 as the threshold in our experiments described in Section V.

V. Evaluation

In this section, we report our experimental validation of our proposed method using real log data sets.

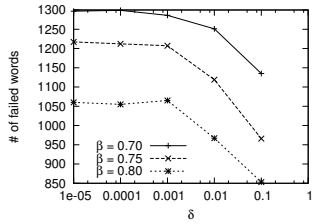

Fig. 5. Comparison of the number of false positive words.

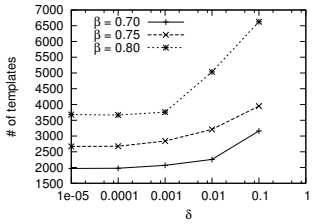

Fig. 6. Comparison of the number of extracted templates.

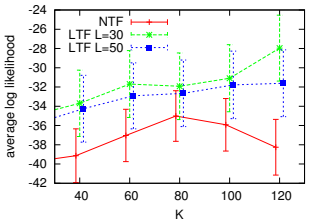

Fig. 7. Comparison of the average of test log likelihood with different K and L values. Error bars represent the 95% confidence interval of a mean.

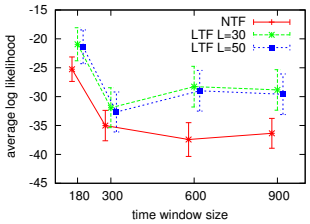

Fig. 8. Comparison of the average of test log likelihood with different L and time-window sizes and $K = 60$. Error bars represent the 95% confidence interval of a mean.

A. STE Evaluation

We first show the evaluation results for the template extraction, STE. We used over 5 million lines of logs captured in a certain small network. In this evaluation, we set two metrics: *accuracy* and *effectiveness*. However, because of a lack of ground truth, it was difficult to measure the accuracy. Therefore, we assumed the following two scenarios: (i) calculate the number of *false positive words*, which are words without numbers removed by STE; and (ii) calculate the number of extracted templates. We observed that words without numbers tend to be template words; however, they may be template words such as a hostname and a path name, e.g. `/usr/local/path`.

The result for (i) is shown in Fig. 5. The vertical axis represents the number of false positive words. It is clear from the graph that if parameter δ increases, STE removed too many words. Since our data set include more than a hundred of thousands of unique words, we can confirm that our method has sufficient accuracy. Next, Fig. 6 shows the result for scenario (ii). The graph shows that our template extraction method can reduce the size of log data to less than 1%. We can see that if δ or β increases, then the total volume of templates also increases. This character of β is obvious, because β determines the threshold for the number of parameter words in each log messages. On the other hand, if δ increases, then DBSCAN algorithm split words into less clusters, hence, the number of templates decreases. Finally, we choose $\beta = 0.75$ and $\delta = 0.01$ as practical values in the following all experiments by observing the output templates.

B. LTF Evaluation

1) Evaluation metrics: We first explain our evaluation metrics. Our primary goal is to model the generative rules of log data and network events and to extract meaningful events for network operations. Therefore we set the following two metrics: (i) the *expressive power* of LTF for real log data; and (ii) the *effectiveness* of the outputs of LTF. To evaluate (i), we selected a well-known quantitative measure, which evaluates how we can predict non-zero elements in unknown log data with a given input tensor. More specifically, we employ the average log likelihood for the test set, which consists of non-zero elements randomly chosen from the target data tensor $\boldsymbol{X}$. The predictions for the test set are calculated by the LTF model that learned from $\boldsymbol{X}$ without chosen test data points. The average log likelihood is then defined as

$$\frac{1}{N}\sum_{n=1}^{N}\log p(x_n;\tilde{x}_n),$$

where $p(\cdot)$ is a Poisson distribution function, N is the number of non-zero elements, and $\tilde{x}_n$ represents the prediction of each element. Note that a higher average log likelihood indicates better modeling of the data. For the validation of (ii), we calculated the *false positive rate* of output template groups by manual inspection. To obtain the false positive rate, we labeled all template groups of LTF as 'related' or not by checking the locations or protocols described in the messages. Note that we do not check network events. This is because if the template groups are accurately extracted, we can find the meanings of the network events.

2) Comparison with NTF: In the first scenario of our experiments, we verify that our model fits well to real data. We choose the NTF as the baseline of our model. As we discussed in Section IV-B, the LTF can be considered as an extension of the NTF because the LTF extracts template groups and network events at the same time. The input data we used here is over 600,000 lines of 1-day log data from a certain network, with the dimension roughly $100\ (I) \times 150\ (H) \times 150\ (J)$. We set the number of masked elements N to 5 and iterate 50 times for both parameter estimation and learning processes. In Fig. 7, we show the average test log likelihood results of the NTF and LTF with different K and L values. We can see that the LTF achieves higher log likelihood than NTF for all K and L values. This result indicates that the LTF model fits better to the real data than the NTF model. We can also see that the NTF achieves the highest value when parameter K is 80, and if K turns away from 80, its log likelihood decreases. The reason for this can be considered as follows: if K is less than the proper value ($K = 80$ in this case), the NTF fails to sufficiently model the input data, and if K is higher than the proper value of K, then they over-fit the test data and can not predict the removed entries. On the other hand, the LTF takes higher values when $K > 80$. This is because the LTF expresses the log data with L template groups, whereas the NTF expresses the log data with K the rank-1 tensors. This fact means that there certainly exist template groups that tend to co-occur, and our modeling is valid for the real data.

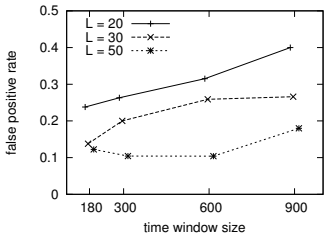

Fig. 9. The FPR with different time-window sizes and L's.

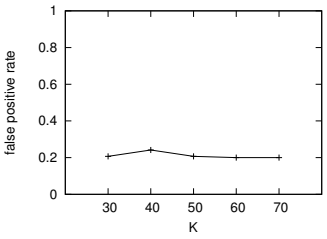

Fig. 10. The FPR with different K's and fixed time-window size and L.

Next, Fig. 8 shows the impact of the time-window size on the average log likelihood. We also can see that the LTF achieves better results than the NTF in all time-windows. In particular, when the time-window size is 180 sec. ($= 3$ min.), the average log likelihood is the highest. This fact means that most of the network events tend to occur within 3 min. The graph also shows that if the time-window size is greater than 180, then the average log likelihood decreases. The reason for this fact is that when the time-window is too large, each time-window may contain irrelevant network events.

3) False positive template groups: In the second experiment, we confirmed that LTF could efficiently output accurate information from the network operators' viewpoints. We used the same input data as in the previous experiment. Due to the lack of ground truth data, we calculated the false positive rates (FPR) of LTF outputs by manually checking all template groups whether they were 'related' or not. Fig. 9 shows FPR with different time-window sizes and L, where K was fixed at 60. Clearly, approximately 80–90% of the outputs of LTF can be considered to be meaningful events. We can also see that time-window sizes larger than 300 sec. result in larger FPR, except for the case of $L = 50$. The reason for this is similar to the case of evaluating the average log likelihood with different time-window sizes. The case in which only $L = 50$ has a different tendency can be explained as follows: when L is too large, the LTF outputs more redundant template groups, i.e. it over-fits the input data. In Fig. 10, we show the FPR with different values of K where L and the time-window size are fixed at 30 and 5 min., respectively. We can observe that the value K does not have much effect on L. This fact means that the number of template groups are selected independently with the number of network events. This character is also seen in Fig. 7.

VI. Case Study

In this section, we present several interesting network events obtained by the LTF as case studies. We conducted an LTF experiment to one week of log data of a certain large scale network where tens of thousands of hosts are working. The input data include daily maintenance, temporally abnormal logs and even critical network faults. Since our dataset was large, we split it into independent host groups that are considered to be related, in order to avoid computational costs. As our primary implementation, we can finish each experiment of a one-week tensor with the dimensions of about $100 \times 100 \times 1000$ roughly within a day. LTF can be easily applied to parallel processing by splitting the inputs, therefore the performance of LTF can be considered to be sufficient for an event extraction purpose. We selected appropriate parameters K and L for each input, and we fixed the time-window size at 5 min.

A. Case: Self-generating Events

Although one of our objectives is to find relationships between multiple hosts, there are *self-generating* events that only appear at a single host and do not have any influence on other hosts. Table III lists the examples of them. The first example is a simple user login event, which is a very common event in the daily network operation. We can confirm that even if a template group contains only one element (template), the proposed method was able to extract it accurately, by observing that the corresponding weight value is 1.0. The second example is a fan speed-up event at a certain host. This type of event can be found on some other hosts and is not related to alarms, thus, operators do not pay much attention to them. By observing the raw log messages, we can see that the time intervals between each messages of this events are approximately 3–5 minutes. As as result, the time-window size should be more than 180 sec. to capture this event. It is understood that the router fan speed-up implies an increase in the temperature, which may cause an unintended shutdown of components. Thus, even if it does not directly relate to a complete fault, we believe that this template group is important for the daily operation. The third example is an administrative link down event, which does not have any influence on other hosts. This is because the neighboring nodes are outside of their monitoring area, and thus log data are not collected. We can also see that even this simple link down event has three log templates because of protocol interaction.

B. Case: Neighboring Nodes Flap Event

One of the interesting outputs of LTF is a network event with a neighbor relationship. An example is presented in Table IV. This example shows that an interface flap has occurred at the CoreRouterA, and then a related interface and link flap has occurred at its neighbor EdgeRouterB. As we mentioned in Section IV-B, it can be seen that these neighboring interface events result in different template groups. Further, in EdgeRouterB's flap event, we can see that the layer 1 protocol flap induces layer 2 protocol flap. We note that link flap events are common in real network, because they are caused by typical operations including adding new ports or reloading modules. Thus, they do not always represent critical faults, which would have an impact on users' experience. However, it is also known that link flaps may occur before nodes completely down. Therefore, we can find importance in extracting and identifying such temporally abnormal events. We also note that the LTF can extract the complete link down event separately with this link flap event.

TABLE III
EXAMPLES OF SELF-GENERATING NETWORK EVENTS (ONLY TEMPLATE GROUPS ARE SHOWN)

#	Weights	Templates
1	1.0	login : LOGIN_INFORMATION : User * logged in from host * on device *
2	0.5	chassisd [*] : CHASSISD_BLOWERS_SPEED : Fans and impellers are now running at normal speed
	0.5	chassisd [*] : CHASSISD_BLOWERS_SPEED_FULL : Fans and impellers being set to full speed [system warm]
3	0.33	* : TIME : %LINK-5-CHANGED : Interface * , changed state to administratively down
	0.33	* : * : %LINK-3-UPDOWN : Interface * , changed state to down
	0.33	* : TIME : %LINK-3-UPDOWN : SIP * : Interface * , changed state to down

TABLE IV
NEIGHBORING LINK FLAP EVENT

Host Name	TG Weights	Weights	Templates
CoreRouterA	0.666	0.4	TIME : ifmgr [*] : %PKT_INFRA-LINK-3-UPDOWN : Interface * , changed state to Up
		0.4	TIME : ifmgr [*] : %PKT_INFRA-LINK-3-UPDOWN : Interface * , changed state to Down
		0.2	TIME : %SYS-3-LOGGER_DROPPED : System dropped * console debug messages.
EdgeRouterB	0.333	0.4	* : * : %LINK-3-UPDOWN : Interface * , changed state to up
		0.17	* : TIME : %LINK-3-UPDOWN : Interface * , changed state to administratively down
		0.17	* : * : %LINEPROTO-5-UPDOWN : Line protocol on Interface * , changed state to up
		0.17	* : * : %LINEPROTO-5-UPDOWN : Line protocol on Interface * , changed state to down
		0.05	* : * : %LINK-3-UPDOWN : Interface * , changed state to down

C. Case: Tunneling Disconnection Event

The final case we present is a network event with tunneling relationships. In this example, network operators did not notice the spacial impact of the network event because of the complexity of it. In general, tunneling or virtual path protocols are widely accepted in existing networks. However, they are difficult to maintain because virtual paths change from hour to hour. In Table V, we show an example of a tunneling network event. First, according to network operators, the template group of CoreRouter indicates a module reload event, due to the module error. Since this reload event includes a lot of templates, we can see that the corresponding weight given to each template in the template group is low. Second, virtual path disconnections are caused at EdgeRouters C and D. Since the CoreRouter has more than a hundred connections with other hosts, the corresponding weight given to the template group is low (less than 0.02). From the observation of the raw data, we found that there was a pattern of the template group that occurred periodically at the CoreRouter, and they were information for debugging generated by a certain periodic job. Since a number of these log messages appeared during the duration of the event (approximately 5 minutes), it was difficult to find this module reload and tunneling disconnection event. Further, the volume of not only the other events, but also of the virtual path disconnection messages was large. Thus, they also became an obstacle to identify the network event. As a result, the LTF can accurately extract a complex network event when it is difficult to identify from the raw log data.

VII. CONCLUSION

In this paper, we proposed a spatio-temporal factorization method, which automatically learns underlying network events from unstructured and complex log data. To overcome the complexity of log messages, we developed a statistical template extraction method, STE. Using log templates, we mathematically modeled the patterns of network logs. To formulate spatial and temporal patterns of log messages, we developed a novel tensor factorization technique, LTF. We verified that our model fits well to real data and also showed several case studies in which the proposed methods can extract various useful network events for the network operation.

Understanding the spatial-temporal patterns of log messages provides useful insights into various network operations. Associating learned network events with NMSs enables us to monitor both normal and abnormal events with their spatial impacts even though such relationships are difficult to maintain as in the example of Table V. The extracted information can be used for preventive maintenance operation, such as a link flap event or a fan speed-up event. In addition, by learning normal network events, LTF can detect the abnormal log group or network events that cannot be expressed as a superposition of learned network events in the past. For example, a router reboot event causes various logs such as process initialization. With the network events learned by the LTF, we can detect that certain process did not start because in such cases, the cost function increases. This is an interesting application, because a typical anomaly detection system with threshold monitoring fails to detect the event.

ACKNOWLEDGMENT

The authors would like to thank to Masayasu Miyazaki for helping in the experiments.

APPENDIX:UPDATE RULE DERIVATION

In what follows, we derive only the update rule for $\boldsymbol{V}$. The cases of $\boldsymbol{Z}$ and $\boldsymbol{W}$ can be derived in the same way. Thus, we

TABLE V
TUNNELING PATH DISCONNECTION EVENT

Host Name	TG Weights	Template Weights	Templates
CoreRouter	0.0253	0.4375	SNMP Trap: a status change for a module. Software image for the module is missing or invalid....
		0.0833	os: loader: * for * is *
		0.0833	id of requester is *
		0.0833	OsCrashDump: invalid crash record skipped, ...
		⋮	⋮
EdgeRouterC	0.01656	0.92805	Tunneling Virtual Path * is disconnected, hardware unavailable.
EdgeRouterD	0.01594	0.98308	Tunneling Virtual Path * is disconnected, hardware unavailable.
⋮	⋮	⋮	⋮

omit them. We define the auxiliary function $\mathcal{D}^+(\cdot)$ as

$$\begin{aligned}&\mathcal{D}^+(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W},\boldsymbol{R})\\&=\sum_{i,h,j}\sum_{k,l}-r_{ihjlk}x_{ihj}\log\frac{v_{li}z_{lkh}w_{kj}}{r_{ihjlk}}\\&+v_{li}z_{lkh}w_{kj}+x_{ihj}\log x_{ihj}-x_{ihj},\end{aligned}\tag{7}$$

where r_{ihjlk} is a nonnegative variable such that $\sum_{i,j,h,l,k} r_{ihjlk}=1$ and $\boldsymbol{R}$ denotes a total set of r_{ihjlk}. Due to Jensen's inequality and the convexity of the log function, we obtain for any r_{ihjlk},

$$\log\sum_{k,l} r_{ihjlk}\frac{v_{li}z_{lkh}w_{kj}}{r_{ihjlk}}\geq\sum_{k,l} r_{ihjlk}\log\frac{v_{li}z_{lkh}w_{kj}}{r_{ihjlk}},$$

from which, (3) and (7), it follows that $\mathcal{D}(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W})\leq\mathcal{D}^+(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W},\boldsymbol{R})$, where the equality attained if and only if $v_{li}z_{lkh}w_{kj}/r_{ihjlk}$ is the same for all l,k, i.e.,

$$r_{ihjlk}=\frac{v_{li}z_{lkh}w_{kj}}{\sum_{k',l'}v_{l'i}z_{l'k'h}w_{k'j}}.\tag{8}$$

Next, we minimize $\mathcal{D}(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W})$ by $\boldsymbol{V}$. Substituting (8) into (7) yields

$$\begin{aligned}&\mathcal{D}^+(\boldsymbol{X}\|\boldsymbol{V},\boldsymbol{Z},\boldsymbol{W},\breve{\boldsymbol{V}},\breve{\boldsymbol{Z}},\breve{\boldsymbol{W}})\\&=\sum_{i,h,j}\sum_{k,l}\frac{\breve{v}_{li}\breve{z}_{lkh}\breve{w}_{kj}}{\sum_{k',l'}\breve{v}_{l'i}\breve{z}_{l'k'm}\breve{w}_{k'j}}\cdot x_{ihj}\\&\times\left[\log\frac{\breve{v}_{li}\breve{z}_{lkh}\breve{w}_{kj}}{\sum_{k',l'}\breve{v}_{l'i}\breve{z}_{l'k'h}\breve{w}_{k'j}}-\log v_{li}z_{lkh}w_{kj}\right]\\&+x_{ihj}\log x_{ihj}-x_{ihj}+v_{li}z_{lkh}w_{kj}.\end{aligned}$$

Finally, differentiating the above equation by v_{li} and substituting 0 into both sides leads to (4). □

REFERENCES

[1] CA Spectrum. http://www.ca.com/us/root-cause-analysis.aspx.
[2] EMC ionix platform. http://www.emc.com/products/family/ionix-family.htm.
[3] HP Software. http://www8.hp.com/us/en/software/enterprise-software.html.
[4] IBM Tivoli. http://www-01.ibm.com/software/tivoli/.
[5] Splunk. http://www.splunk.com/.
[6] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies, In *Proc. of SIGCOMM*, 2007.
[7] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl, Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions, In *Proc. of OSDI*, 2008.
[8] A. Cichocki, R. Zdunek, A. H. Phan, and S. I. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley. 2009.
[9] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, In *Proc. of KDD*, 1996.
[10] S. Kandula, D. Katabi, and J. P. Vasseur, Shrink: a Tool for Failure Diagnosis in IP Networks, In *MineNet*, 2005.
[11] S. Kandula, R. Chandra, and D. Katabi, What's Going on? Learning Communication Rules in Edge Networks, In *Proc. of SIGCOMM*, 2008.
[12] S. Kandura, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, Detailed Diagnosis in Enterprise Networks, In *Proc. of SIGCOMM*, 2009.
[13] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, IP Fault Localization via Risk Modeling, In *Proc. of NSDI*, 2005.
[14] D. D. Lee and H. S. Seung, Learning the Parts of Objects by Non-Negative Matrix Factorization, *Nature*, 401, pp.788–791, 1999.
[15] D. D. Lee and H. S. Seung, Algorithms for Non-negative Matrix Factorization, In *Proc. of NIPS*, 2000.
[16] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee, Troubleshooting Chronic Conditions in Large IP Networks, In *Proc. of CoNEXT*, 2008.
[17] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons Detecting the Performance Impact of Upgrades in Large Operational Networks, In *Proc. of SIGCOMM*, 2010.
[18] A. Mahimkar, Z. Ge, J. Wang, J. Yates, Y. Zhang, J. Emmons, B. Huntley, and M. Stockert, Rapid Detection of Maintenance Induced Changes in Service Performance, In *Proc. of CoNEXT*, 2011.
[19] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, What Happened in my Network? Mining Network Events from Router Syslogs, In *Proc. of IMC*, 2010.
[20] P. Smaragdis and J. C. Brown, Non-negative Matrix Factorization for Polyphonic Music Transcription, In *Proc. of WASPA*, 2003.
[21] T. Wang, M. Srivatsa, D. Agrawal, and L. Liu, Learning, Indexing, and Diagnosing Network Faults, In *Proc. of KDD*, 2009.
[22] T. Wang, M. Srivatsa, D. Agrawal, and L. Liu, Spatio-temporal Patterns in Network Events, In *Proc. of CoNEXT*, 2010.
[23] W. Xu, X. Liu and Y. Gong, Document Clustering Based on Non-negative Matrix Factorization. In *Proc. of SIAM*, 20111.
[24] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, Mining Console Logs for Large-Scale System Problem Detection, In *Proc. of SysML*, 2008.
[25] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, Detecting Large-Scale System Problems by Mining Console Logs, In *Proc. of SOSP*, 2009.
[26] K. Yamanishi and M. Maruyama. Dynamic Syslog Mining for Network Failure Monitoring, In *Proc. of KDD*, 2005.
[27] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates, G-RCA: a Generic Root Cause Analysis Platform for Service Quality Management in Large IP Networks, In *Proc. of CoNEXT*, 2010.