Event Summarization for System Management*

Wei Peng
School of Computer Science
Florida International University
Miami, FL 33199, USA
wpeng002@cs.fiu.edu

Chang-shing Perng
IBM T.J. Waston Research Center
Hawthorne, NY 10532, USA
perng@us.ibm.com

Tao Li
School of Computer Science
Florida International University
Miami, FL 33199 USA
taoli@cs.fiu.edu

Haixun Wang
IBM T.J. Waston Research Center
Hawthorne, NY 10532 USA
haixun@us.ibm.com

ABSTRACT

In system management applications, an overwhelming amount of data are generated and collected in the form of temporal events. While mining temporal event data to discover interesting and frequent patterns has obtained rapidly increasing research efforts, users of the applications are overwhelmed by the mining results. The extracted patterns are generally of large volume and hard to interpret, they may be of no emphasis, intricate and meaningless to non-experts, even to domain experts. While traditional research efforts focus on finding interesting patterns, in this paper, we take a novel approach called event summarization towards the understanding of the seemingly chaotic temporal data. Event summarization aims at providing a concise interpretation of the seemingly chaotic data, so that domain experts may take actions upon the summarized models. Event summarization decomposes the temporal information into many independent subsets and finds well fitted models to describe each subset.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-data mining

General Terms

Algorithms,Experimentation, Human Factors

Keywords

Event summarization, Temporal dependency, ERN, Log

*The work is partially supported by an IBM Faculty Research Award, an IBM SUR award, and NSF IIS-0546280.

KDD'07, August 12–15, 2007, San Jose, California, USA.

1. INTRODUCTION

With advancement in science and technology, computing systems are becoming increasingly more difficult to monitor, manage or maintain. Traditional approaches to manage systems have been largely based on domain experts through a knowledge acquisition process to translate domain knowledge into operating rules and policies. This has been experienced as a cumbersome, labor intensive, and error prone process. Therefore data mining techniques have been applied to discover frequent and interesting patterns from system temporal events to monitor and manage computing systems [1, 2].

In system management applications, temporal event data are generated and collected in the form of temporal events. Data mining approaches for analyzing temporal events generally focuses on discovering frequent or interesting patterns in the data–albeit their occurrences may only account for a small fraction of the entire data. It has attracted intensive research efforts, and as a result, a variety of temporal data mining algorithms have been proposed to speed up the discovery of such patterns, as scalability and efficiency are the major focus of these efforts. However, the extracted temporal patterns tend to be massive, without emphasis, intricate, and uninterpretable. Most of them are meaningless or useless to nonexperts, even analysts.

Hence correctly understanding and interpreting patterns is a big challenge. It is of critical importance to enable temporal relationships between events hierarchical and more user-friendly for monitoring and managing complex systems. In this paper, we take a novel approach and demonstrate that an effective way to understand the seemingly chaotic temporal data is through event summarization, which attempts to use well-fitted models to describe the entire data set. Good summarization leads to models that provide concise and comprehensible interpretation of the data so that domain experts can take actions upon each subset of the data. In this paper, we introduce event summarization as a systemic process that summarizes and interprets datasets.

1.1 A Motivating Example

To illustrate the problem, we present a real-world example. Figure 1 shows a set of system management events taken from a business-to-business gateway facility. After preprocessing, each event contains two attributes: times-

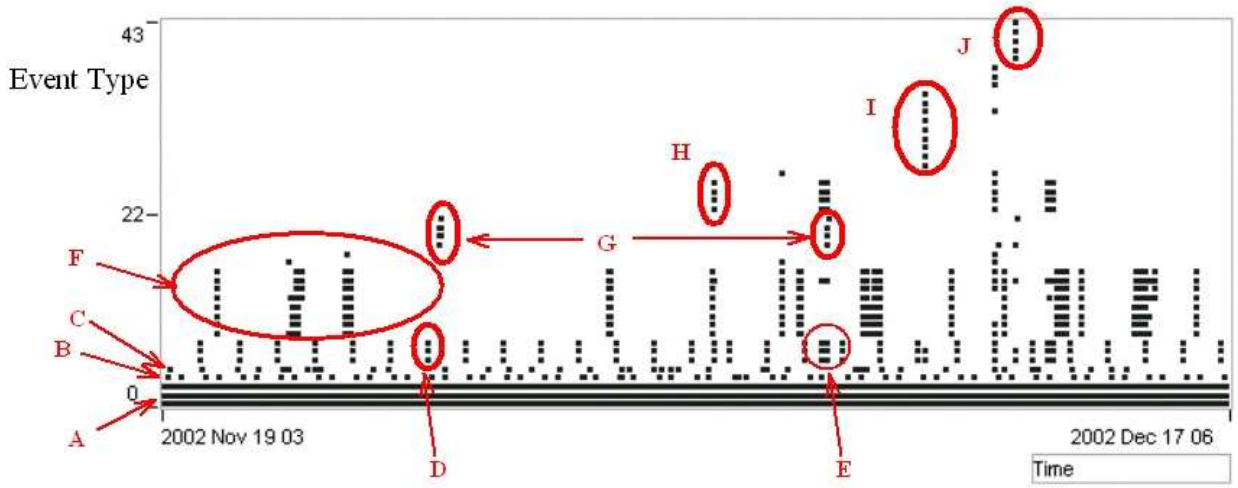


Figure 1: System Management Events

tamp and event type which are mapped to X and Y axis. By visually analyzing the events, it is not difficult to conclude the following:

1. The three event types, labeled by **A** with the lowest Id's are densely populated events. Figure 2 shows a more detailed view of the event and exhibits the bi-periodic pattern with periods 7 minutes and 53 minutes.

2. The event type labeled by **B** forms another bi-periodic pattern with period 8 hours and 16 hours.

3. The event type labeled by **C** appears to be randomly distributed with no apparent period.

4. The three event types indicated by **D** almost always occur at the same time. There is a dominating periodic pattern that governs the event group. However, there are some occurrences such as those indicated by **E** that appear randomly.

5. The 8 events labeled by **F** tend to appear together in a relatively random pace. Sometimes there are another two event types on top that appear along with the group.

6. The event groups labeled by **G** and **H** appear a few times randomly.

7. The event groups labeled by **I** and **J** only occur once.

The above observations are usually useful for decision making. For system management experts, the above summary provides a very good way to itemize patterns and to devise action plans. For example, the **A** section has strong bi-periodicity, so it would make better sense to monitor any violation of such bi-periodicity instead of the normal occurrences; also, 7 of the 8 event types in **F** section can be turned off without losing any information.

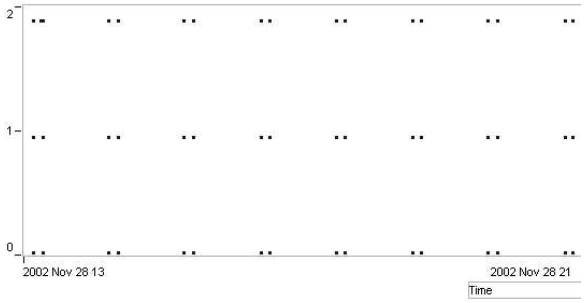


Figure 2: The events labeled A in Figure 1

1.2 Content of the Paper

Currently, such type of data analysis suffers from the lack of a systematic way to discover and describe interesting phenomena and patterns contained in the data. Some analysts pay more attention to the vertical patterns (event spike), some focus on temporal relationship between events, and some regard the event rate as the most important. In this paper, we describe our research efforts on event summarization for system management. We define several measures for describing temporal patterns and use them as an essential foundation for weighting and ranking relationships between temporal events. We then propose a divide-and-conquer progress to summarize the datasets.

2. NOTATIONS AND MEASURES

2.1 Events and Patterns

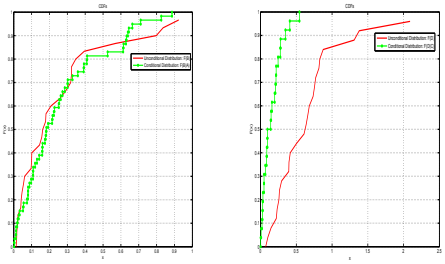
An event has two components: the event type and the time stamp (occurring time or arrival time). Throughout the paper, we assume a finite set E of event types. We use the symbol e, possibly with subscripts to denote event types. An *event* is then a pair $< e, t >$ where $e \in E$ is an event type and t is the timestamp of e. Our data, called an *event sequence*, is then a collection of all events occurred within some time range (say, between 0 and T) $D = \{< e_1, t_1 >, < e_2, t_2 > , \ldots, < e_n, t_n >\}$, where $e_i \in E, t_i \in [0, T]$, and $t_i \leq t_{i+1}$.

2.2 Temporal Dependency

In general, the dependence relationships are essential factors/patterns to be discovered: they assert that an event usually happens after another event. Such patterns of interest appear naturally in the system management applications [4]. In our work, we start with pairwise temporal dependencies. Let T_a and T_b be the sequence of timestamps for event a and b respectively. Let the distance from a point z (or, a timestamp), to an event sequence T_a to be $d(z, T_a) = min_{x \in T_a, x \geq z}(x - z)$, i.e., the distance from the point z to its nearest neighbor in T_a which occurs after z. In order to whether b is dependent on a, we compare the typical distance of random points z to T_b to the typical distance of points $z \in T_a$ to T_b. This technique is motivated by the following idea: *if the occurrence of b is predictable by the occurrence of a, then the conditional distribution which models the waiting time of event type b given event type a's presence would be different from the unconditional one* [5]. In particular, we compare the following two inter-arrival distributions: (1) The unconditional distribution of the waiting time of event b: $F_b(r) = \mathcal{P}\{d(x, T_b) \leq r\}$; (2) The conditional distribution for the waiting time of event b with respect to event a: $F_b^a(r) = \mathcal{P}(d(x, T_b) \leq r | x \in T_a)$. The conditional distribution can be regarded as the conditional probability distribution given there is an event of a at time x. If the two distributions $F_b(r)$ and $F_b^a(r)$ are different, then we say that b is directly dependent on a. This technique is extended from our earlier work [6] and can be used to discover temporal dependencies without using predefined time windows.

After temporal dependencies have been discovered, the time intervals between the two events can be discovered by applying existing lag detection algorithms (see [9], e.g., peak finding, clustering, and density estimation) on the collection of time distances between them. These time intervals can be easily incorporated into our Event Relationship Network.

2.3 Discovering Temporal Dependencies

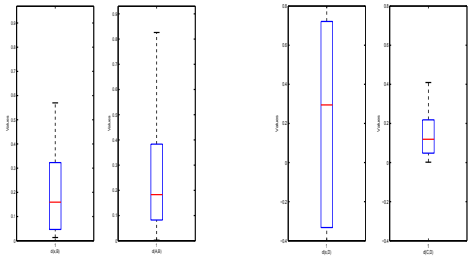


(a) CDFs of two independent events.

(b) CDFs of two dependent events.

Figure 3: Cumulative Distribution Functions (CDFs).

To illustrate the procedure for discovering pairwise temporal dependencies, Figure 3(a) plots the cumulative distribution functions(CDFs) of unconditional F_B and conditional F_B^A for two sample independent events A and B in our dataset, and Figure 3(b) plots the two CDFs for two dependent events C and D. The difference between the two CDFs clearly indicates the dependencies between these two events. In order to compare the differences between the independent relationship and dependent relationship, Figure 4(a) shows the boxplots of $d(x, B)$ where x is a random event and $d(A, B)$. Figure 4(b) shows the boxplots of $d(x, D)$ and $d(C, D)$. The lines in the approximate middle of boxes are medians. By using a robust order statistics method which compares the ranges of confidence intervals, we can also conclude that event B is independent of event A while event D is dependent of event C.



(a) The Boxplots of d(x,B) and d(A,B) of Independent Events A and B.

(b) The Boxplots of d(x,D) and d(C,D) of dependent Events C and D.

Figure 4: Boxplots.

2.4 Ranking Dependencies

In order to summarize temporal datasets, we use *Entropy* to rank the event dependency relationship. Entropy indicates the amount of information contained in the event patterns and measures the uncertainty of conditional probability distributions on a particular event. We use $FCP(e_j \mid e_i)$ to denote the forward conditional probability of event e_i followed by e_j relative to all event pairs starting from e_i. $BCP(e_i \mid e_j)$ is used to denote the backward conditional probability of e_j preceded by e_i relative to all event pairs ending at e_j.

The forward entropy of an event Q can be defined as follows [3]:

$$(1) \qquad FE(Q) = -\sum_{e \in E} FCP(e \mid Q) \times log_N(FCP(e \mid Q)),$$

where E is the set of all event types, N is the number of events in E, and $\sum_{e \in E} FCP(e \mid Q) = 1$. For an instance, if event A is always only followed by B, the forward conditional probability of AB, $FCP(B|A) = 1.0$, and the forward conditional probabilities of the sequences that A is followed by the other events are 0.0. The higher the entropy, the less deterministic the conditional probabilities are distributed. This means that if an event has a lower forward entropy, it has more confidence or liability to select some events to proceed. Similarly the backward entropy can be calculated as follows:

$$(2) \qquad BE(Q) = -\sum_{e \in E} BCP(e \mid Q) \times log_N(BCP(e \mid Q)),$$

where $\sum_{e \in E} BCP(e \mid Q) = 1$. If an event, or an event sequence has a lower backward entropy, it has more deterministic precedents. In summary, entropy aims to capture the temporal dependence relationships among events.

2.5 Concurrent Behaviors

Concurrency indicates that multiple events are reported or performed in parallel rather than sequentially. Suppose an event e_1 has a forward conditional probability- distribution of $(0.9, 0.05, 0.05, 0.0)$ on four events; it is obvious this event has a more deterministic successor. Suppose another event e_2 has a distribution of $(0.45, 0.55, 0.0, 0.0)$. Obviously e_2 has a 2-peak distribution, i.e., there might be concurrent behaviors after e_2. Hence, to detect concurrent behaviors, for each event, we compare its conditional probability distribution with the prototype distributions $Peak_1 = (1, 0, 0, \cdots, 0)$, $Peak_2 = (0.5, 0.5, 0, \cdots, 0)$, $\cdots$, $Peak_N = (1/N, 1/N, \cdots, 1/N)$, where N is the number of event types, allowing all possible permutations.

3. EVENT SUMMARIZATION PROCESS

In this section, we discuss how to summarize an event set to an easy-to-understand plot based on the pair-wise event relationship. We also discuss some typical ways to reduce summarized patterns into action rules.

3.1 Event Relationship Network

Event Relationship Networks (ERNs) [7, 8] is a graphical representation of temporal relationship among events. ERN is widely used in IT system management and business services management area as a product-agnostic format for bridging the knowledge discovery process with the system implementation process. Event summarization discussed in this paper is aiming to render output in a form of ERN that is richer and more expressive than what the industry is using. Here we will explain both the representation and the construction process through an example.

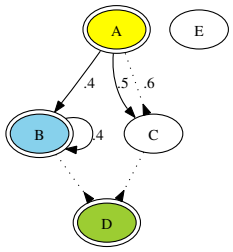


Figure 5: Event Relationship Network

Besides the statistics discussed in the previous section are computed, an additional required input is a threshold tuple for distinguishing whether a particular statistics is significant enough. Threshold tuples are in the form of

$$\langle \Theta_{FE}, \Theta_{BE}, \Theta_{FCP}, \Theta_{BCP} \rangle.$$

Note that Θ_{FE} and Θ_{BE} are upper bounds while Θ_{FCP} and Θ_{BCP} are lower bounds. The intended output, ERN, use nodes to denote events with low Θ_{FE} or Θ_{BE}. These are the events with either more certain leading events or more certain following events. ERNs also need to show significant event relationship as links. The ERN construction process includes the following steps: (1) Plot all the events with $FE \leq \Theta_{FE}$ or $BE \leq \Theta_{BE}$ as nodes in color yellow with two peripheries. As event A in Figure 5. (2) Plot all the events with $FCP \geq \Theta_{FCP}$ or $BCP \geq \Theta_{BCP}$ as nodes in color blue and green respectively with two peripheries. As event B and D. (3) For each event pair X and Y with $FCP(Y|X) \geq \Theta_{FCP}$, plot a forward link as the one from A to B. (4) For each event pair X and Y with $BCP(Y|X) \geq \Theta_{BCP}$, plot a backward link as the one from A to C. The result is an ERN looks like Figure 5.

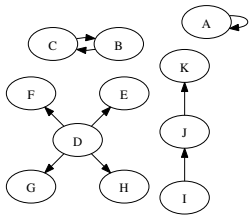


Figure 6: Usual Event Patterns

In Figure 6, we show a few common event patterns: (1) **Periodic patterns**: An event with self-loop like event A with near-constant interval is a periodic event pattern. Typical examples are events results from periodic probing and system heartbeat monitoring. (2) **Circular patterns**: A set of events forming a loop is a circular pattern. (3) **Simultaneous patterns**: A set of events always occur together as **F** in Figure 1 is a simultaneous pattern. If the intervals among D, E, F, G and H in Figure 6 are close to 0, then they form a simultaneous pattern. (4) **Event chain**: Event I, J and K form an event chain. Such patterns are frequently observed from events that represent different stages of a problem and the problem progresses in a predictable sequence.

3.2 Derive Action Rules from Event Summary

As mentioned, the ERN is a product-agnostic and vendor-neutral representation of event dynamics. Consider action rules in the form of

if **condition** is true, take **action**

ERNs contain the information required by the **condition** part of the rule. The **action** part is usually invocation of self-healing scripts or notification for system administrators. Some popular types of rules can be derived from ERNs almost directly. Here are some examples: (1) **Event reduction rules:** A self loop with a nearly constant arrival rate means an event keeps occurring in a constant pace. In system management, such a periodic event pattern is usually a series of probing reports. For those reporting normal, a typical action rule is to invert it – only report the missing of it. For those reporting abnormal, each consecutive "episode" is converted to a start and and an end of an anomaly, those in between can be suppressed. (2) **Event correlation rules:** An event is usually a "symptom" of a problem. Each problem may have many symptoms. For system administrators to correctly take remedial actions, the monitoring system provides root cause analysis (RCA) capabilities. One way to achieve RCA is to use ERNs. An ERN built from a system management event set is usually a set of disconnected subgraphs. Each subgraph contains a set of correlated events and hence represents a problem. (3) **Problem avoidance rules:** Some events indicate a problem as reached a severe stage and immediate actions should be taken to prevent them from occurring. ERNs provides a natural way to predict these severe states.

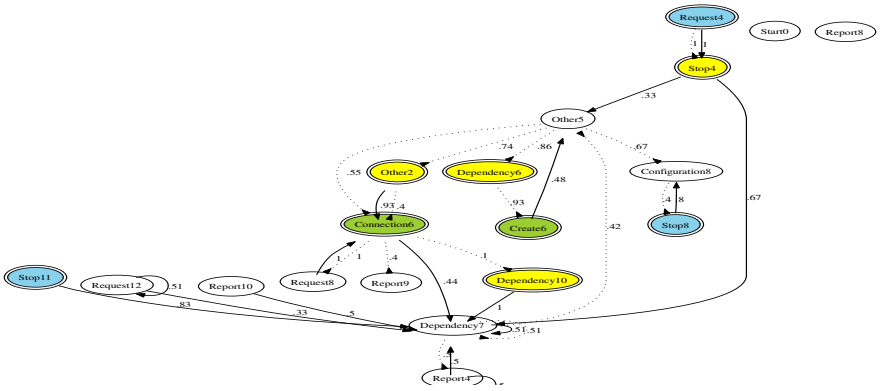


Figure 7: The ERN model summarizing the extracted patterns.

4. A CASE STUDY FOR SYSTEM MANAGEMENT

4.1 System Log Data Description

We collect log files from several Windows machines in the Graduate Lab of the Computer Science School of Florida International University. The detailed information about the log files can be found in [5]. Here each message is represented as a three-tuple $\langle s, c, t\rangle$ where s is the state type, c is the source component of this log message, and t is the timestamp. In order to discover event relationships among components, we map each distinct pair of $\langle s, c\rangle$ into a unique event type e. This representation enable us to analyze the historical event data across multiple components and to discover interesting patterns embedded in the data. In our dataset, we have 9 different states including *start*, *stop*, *dependency*, *create*, *connection*,*report*, *request*, *configuration*, and *other*. We have 13 different event sources, denoted by $0 \cdots 12$.

4.2 Experiment Results

We perform a case study on a real log dataset using event summarization. The dataset contains the following event types: *start0, dependency7, other5, other2, dependency6, create6, report9, configuration8, connection6, report0, report8, stop8, request4, stop4, request8, dependency10, stop11, request12, report4* where the prefix words in indicate the event states and the last number is the component ID (i.e., event sources) that reports the events. Due to the space limit, we omit the detailed decomposition process of the dataset by event summarization. The ERN summarization model constructed from the extracted temporal patterns is given in Figure 7.

The event summarization approach is obviously useful for system management experts to make decisions. First, the dependency patterns are extracted based on the rank. The divide-and-conquer process provides a systematic way to decompose the dataset. Second, the ERN model provides a good guideline for identifying patterns and to devise action plans. For example, the temporal pattern event 5 (i.e., dependency6: event source 6 in dependency situation) followed by event 6 (i.e., create6: event source 6 in create situation) implies the creation requests in component 6 are generally resulted by the lack of resources. So to reduce the creation requests, we need to allocate more resources to the component. As an another example, the temporal pattern event 12 (i.e., stop8: event source 8 in stop situation) followed by event 8 (i.e., configuration8: event source 8 in configuration situation) implies the stop process in component 8 often leads to the configuration procedure. So to speed up the exit process of component 8, we need to either eliminate the configuration procedure when exiting or improve the configuration procedure.

5. CONCLUSIONS

In this paper, we present a novel approach, event summarization, to understand and interpret temporal log data for system management. Event summarization aims at providing a concise interpretation of the seemingly chaotic data, so that domain experts may take actions upon the summarized models. A divide-and-conquer process is employed to extract temporal patterns based on entropy ranking and an ERN is constructed to provide concise representations. We present a case study using the real-world system log data. It should be noted that event summarization is a general concept that can be applied to various types of applications.

6. REFERENCES

[1] Mike Chen, Alice X. Zheng, Jim Lloyd, Michael I. Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *ICAC'04*, 2004.

[2] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *OSDI'04*, pages 231–244, 2004.

[3] J.E. Cook, Z. Du, C. Liu, and A.L. Wolf. Discovering models of behavior for concurrent workflows. *Computers in Industry*, 53(3):297–319, April 2004.

[4] Joseph L. Hellerstein, Sheng Ma, and Chang-Shing Perng. Discovering actionable patterns in event data. *IBM System Journal*, 41(3):475–493, 2002.

[5] Tao Li, Feng Liang, Sheng Ma, and Wei Peng. An integrated framework on mining logs files for computing system management. In *KDD'05*, pages 776–781, 2005.

[6] Tao Li and Sheng Ma. Mining temporal patterns without predefined time windows. In *ICDM'04*, pages 451–454, 2004.

[7] Chang-Shing Perng, David Thoenen, Genady Grabarnik, Sheng Ma, and Joseph Hellerstein. Data-driven validation, completion and construction of event relationship networks. In *KDD'03*, pages 729–734, 2003.

[8] D. Thoenen, J. Riosa, and J. L. Hellerstein. Event relationship networks: A framework for action oriented analysis for event management. In *IM 2001*.

[9] Andrew Webb. *Statistical Pattern Recognition*. Wiley, 2002.