

Mining Temporal Patterns Without Predefined Time Windows

Tao Li

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

Sheng Ma

IBM T.J. Watson Research Center
Hawthorne, NY 10532
shengma@us.ibm.com

Abstract

This paper proposes algorithms for discovering temporal patterns without predefined time windows. The problem of discovering temporal patterns is divided into two sub-tasks: (1) using “cheap statistics” for dependence testing and candidates removal (2) identifying the temporal relationships between dependent event types. The dependence problem is formulated as the problem of comparing two probability distributions and is solved using a technique reminiscent of the distance methods used in spatial point process, while the latter problem is solved using an approach based on Chi-Squared tests. Experiments are conducted to evaluate the effectiveness and scalability of the proposed methods.

1. Introduction

In many real-world applications, an overwhelming amount of data are generated and collected in the form of temporal events. We consider the problem of discovering temporally dependent patterns with some time relationships (e.g., event a followed by b in about 5 seconds) and propose algorithms that discover the temporal patterns without predefined time windows. Specifically, the algorithms enable to discover the following two types of temporal patterns:

Loose temporal patterns. These patterns assert dependency between events. The patterns of this type assert that an event a “usually” precedes another event b or a usually follows by b before time t . Intuitively, this pattern characterizes the situation that a leads to b , but the time is not precise.

Stringent temporal patterns. Patterns of this type specify time distance between two events in loose temporal patterns. Usually, it can be described as “event a happens after event b , say, about 5 minutes”.

In the rest of the paper, we refer this two types of patterns as **t-patterns**. In both patterns, the dependence and the temporal relationships are essential factors to be discovered. Such temporal patterns of our interest appear naturally in all the real-world applications. In specific, a computer system problem may trigger a series of symptom events/activities. Such a sequence of symptom events provides a natural signature for identifying

the root cause [4].

Previous work of temporal mining focuses on frequent itemsets with a predefined time window. It fails to address two important aspects often required by applications. First, the fixed time-window scheme can not explore precise temporal information within a window, and misses the opportunity to mine temporal relationship longer than the window size. To apply association mining techniques a predefined time window has to be used [8, 7] thereby reducing temporal data to basket data. As we shall see later in our experimental results on system management applications, the temporal relationships discovered have time distance ranging from one second to one day. The fixed time-window scheme can not discover all such patterns and a new mechanism is needed to discover temporal relationship specific to an individual pattern. Second, as well-known for transaction data, frequent pattern framework misses significant, but infrequent patterns. In most network management applications, frequent patterns are normal operations and service disruptions are usually infrequent but significant.

In this paper, ideas from spatial statistics are used to characterize the inter-arrival time of events. The dependence problem is formulated as the comparison of two probability distributions and our technique for testing the dependence is reminiscent of the distance methods in spatial point process. Distance methods make use of precise information on the arrival time of events and have the advantage of independent of choices of window size [3, 2]. Also, the statistical properties imposed on the patterns provide meaningful characterizations and they are usually robust against noise. As a first step, we focus on pairwise patterns. Pairwise patterns can be easily interpreted by domain experts, and be easily visually presented. In addition, pairwise patterns can be extended to longer temporal patterns using a level-wise strategy.

2. Notations and Problem Formulation

2.1. Notations

An event has two basic components: the event type and the time stamp (occurrence time or arrival time). Throughout the paper, we assume a finite set E of event

types. Examples of event types include “router up” or “switch down” etc. We use the symbol e , possibly with subscripts to denote event types. An event is then a pair $\langle e, t \rangle$ 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)

$$D = \{ \langle e_1, t_1 \rangle, \langle e_2, t_2 \rangle, \dots, \langle e_n, t_n \rangle \},$$

where $e_i \in E, t_i \in [0, T]$, and $t_i \leq t_{i+1}$. Define the shortest distance from a point z to the point process a by $d(z, T_a) = \inf_{x \in T_a, x \geq z} |x - z|$. Intuitively, the distance is defined to be the distance from the point z to its closest neighbor in T_a . Since each event can be viewed as a point process, to test the dependence between events is to test the dependence of the two point processes. Denote $t_i = b_i - b_{i-1}$ as the inter-arrival time for T_b , where $i = 1, \dots, n_b$ and $b_0 = 0$. Let $n'_a = N_a([0, T'])$ denote the number of points in T_a within the time range $[0, T']$. For each point $a_i, 1 \leq i \leq n'_a$, denote d_i as the distance from a_i to the nearest b occurring after b_i , i.e., $d_i = d(a_i, T_b)$.

2.2. Two Interarrival Distributions

Let T_a and T_b be two point processes for event a and b respectively. We would like to check whether T_b is dependent on T_a . Our approach is motivated from the methods of measuring spatial associations between spatial point patterns developed in statistical community [2, 3]. The idea of the approach is based on the observation: *if the occurrence of event type b could be predictable by event type a , then the conditional distribution which modeling the waiting time of event type b from event type a 's presence would be different from the unconditional one.*

We now formally introduce the concept of two interarrival distributions. These concepts are described in detail in [3, 1] and our description here is adapted to suit our purpose for event data.

Definition 1 *The unconditional distribution of the waiting time of event b is defined as $F_b(r) = \mathcal{P}\{d(x, T_b) \leq r\}$.*

The distribution can be interpreted as the probability of having event type b within $[0, r]$.

Definition 2 *The conditional distribution for the waiting time of event b with respect to event a is defined as: $F_b^a(r) = \mathcal{P}(d(x, T_b) \leq r | x \in T_a) = \mathcal{P}^x(d(x, T_b) \leq r)$ where \mathcal{P}^x denotes the Palm distribution [3] at an arbitrary point x with respect to the process a .*

The conditional distribution can be regarded as the conditional probability distribution given there is an event of a at time x . So, the dependent relationships between event types are then defined based on the two distributions.

Definition 3 *Given two event types a and b , Denote their arrival times as T_a and T_b . We say that b is directly dependent on a if the two distributions $F_b(r)$ and $F_b^a(r)$, defined respectively in Definition 1 and Definition 2, are different.*

3. Two-Stage Approach

Statistical testing for spatial data has been developed in statistical community and has been used for geostatistical data and lattice data etc [3]. Our main contributions are two-fold. First, we extend the existing results to temporal events and propose estimation methods to compute the statistics. Second, we develop a two-stage algorithm that scales up the hypothesis testing. In the first stage, we do a simple statistical test based on low order statistics. As we shall see, such a simple test may generate false positives, but not false negatives. It helps to eliminate candidate space dramatically. Then, the second stage performs more detailed analysis in determining temporal relationships. Due to space limit, we only present the sketch of our idea in the paper. For detailed description, please refer to [5].

3.1. Stage One: Testing the Dependences

Estimating the Two Distributions. Since the true distributions are not available, we have to first estimate them. Under the stationarity assumption, the distribution of $d(x, T_b)$ does not depend on the location of x . The assumption of stationary is a pragmatic simplification which justifies the use of relatively simple estimation methods [3]. The two distributions can be estimated as follows: let T' be the last arrival time of event type b , i.e., $T' = b_{n_b}$. We use the information in $[0, T']$ for estimation. $F_b(r)$ can be estimated by measuring the distance $d(x, b)$ from each of several arbitrary test points to the nearest event occurrence in b and forming an empirical distribution. Similarly, $F_b^a(r)$ can be estimated by measuring the distance from each occurrence a_i of the point process a to the nearest arrival of event type b , and forming an empirical distribution function.

Dependency Tests On First Moments. Under the null hypothesis that there is no dependency between event type a and b , it is expected that $F_b = F_b^a$. For efficient comparison, we have developed efficient computation approaches to first test the differences between the estimated distributions and remove candidates of dependent pairs. Our computation is based on the first moment difference between the distributions. It is easy to see that if $F_b = F_b^a$, so are their first moments. Counterexamples can be constructed in which a and b are dependent but the first moments of two distribution functions are the same. However, stage one only serves as a preprocessing step for reducing the candidate space, the conservative property of the test ensures that we don't

get rid of any true dependent pairs. It can be shown that, under certain assumptions, the first moments can be directly computed from the inter-arrival times.

3.2. Stage Two: Identifying the Relationships

After preprocessing via “cheap statistics” in stage one, the next task is identifying the dependence between the candidate pairs and finding the waiting period (or lag) between two dependent events. Let δ be the time tolerance accounting for factors such as phase shifts and lack of clock synchronization.

Definition 4 Given b is dependent on a , we say that the waiting period of b after a is p if the distance sequence $D_{ab} = (d_1, \dots, d_{n_a})$, has a period p with time tolerance δ .

The discovery of the waiting periods is carried out using the Chi-Squared test based approach first introduced in [7]. Consider an arbitrary element τ in D_{ab} and a fixed δ . Let C_τ be the total number of elements of D_{ab} that fall into the interval $[\tau - \delta, \tau + \delta]$. Intuitively, if τ is not a period p , C_τ should be small; otherwise it should be large. The idea here is to compare C_τ with the number of elements in $[\tau - \delta, \tau + \delta]$ that would be expected from a random sequence. The procedure for identifying the relationship, as described in *Algorithm 1*, is essentially a one dimensional clustering process. $|D_{ab}|$, the possible candidate, is a sequence for a direct dependence between two event types and it is a much shorter sequence than raw data that contains a mixture of event types. Note that $|C - d_i|$ is the distance between C and d_i . It is not hard to see that Step 1 to Step 3 in *Algorithm 1* perform a one-pass clustering. In particular, Step 3.2.1 computes the cluster center (i.e., the mean of the samples in the cluster) in an incremental fashion. P' stores the cluster center and the size of each cluster. The cluster centers obtained from the *Algorithm 1* are candidates for the lags. It can be shown that: the distance from any cluster center to the sample belonging to the cluster is kept within the time tolerance δ .

4. Experiments

4.1. Synthetic Data

All the experiments are performed on a P3 1GHz machine with 256M memory running Linux 2.4.20-19.9. We first test our algorithm on synthetic data. The synthetic datasets are characterized by the duration of time, number of t-patterns, number of event types, total number of events and NSR ratio. Noise events are randomly and uniformly generated. Our first experiment is to test the effectiveness of the algorithm in the presence of noise. We set the number duration of time to $[0, 500]$, the number of t-pattern to 10, the number of event type to 40 and

Algorithm 1 Procedure for Lag Detection

Input: D_{ab} ; δ : Time tolerance

Output: P : set of lags, (initialized to be \emptyset)

1. Randomly pick a point d in D_{ab}
 2. Set $N(d) = 1$, $P' = \{(d, N(d))\}$;
 3. For $d_i \in D_{ab}$ Begin
 - 3.1 $C =$ the point in P “closest” to d_i
 - 3.2 If $|C - d_i| \leq \delta$ Begin
 - 3.2.1 $C = \frac{N(C)}{N(C)+1}C + \frac{1}{N(C)+1}d_i$
 - 3.2.2 $N(c) = N(c) + 1$; End
 - 3.3 Else Begin
 - 3.3.1 Set $N(d_i) = 1$;
 - 3.3.2 Add $(d_i, N(d_i))$ to P' ; End
 4. For each C in P' Begin
 - 4.1 Compute the threshold C'_C as in [7]
 - 4.2 If $N(C) > C'_C$, then insert C into the output P
-

the total number of events to 40000. Figure 1 shows the effectiveness of the algorithm with different NSR ratio in both tabular and graph formats. A run is effective if there is no false negative, i.e., all t-patterns are correctly discovered. The effectiveness is defined as the percentage of the correctly discovered patterns. We observe that the effectiveness is 100% if NSR less than 1 and gradually decreases as NSR increases above 1. Especially when NSR is greater than 4, the effectiveness falls beyond 50%. The

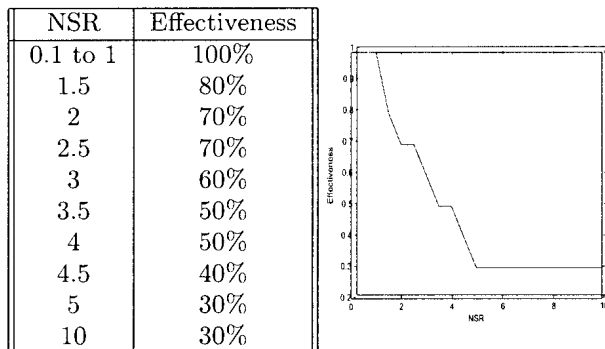


Figure 1. Effectiveness Results with Different NSR.

second experiment is to assessing the scalability results of the algorithm. NSR ratio is controlled between 0.5 – 1 and the number of events per type is controlled between 2000 – 5000. Figure 2 plots the average CPU time in the run against the number of events.

4.2. Product Data

Now, we apply our algorithms to mine t-patterns in real data. Here our evaluation criteria are more subjective in that we must rely on the domain knowledge to

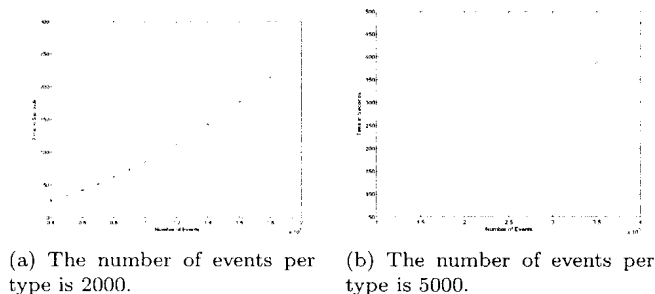


Figure 2. CPU time vs. the number of events

decide whether the results is correct or not. We review the discovered patterns with operation staff. In addition, whenever possible, we also use the Event Browser [6] to visualize and verify the results. Four datasets are considered. These datasets were collected from either an Intranet containing hundreds of network elements (e.g., routers, hubs and servers) or from outsourcing centers that support multiple application servers across a large geographical regions. Events in these datasets are either application-oriented (e.g., the CPU utilization of a server is above threshold) or network-oriented (e.g, the link is down for a router). An event in these datasets consists of three key attributes: *Host ID*, which is the source of event; *Event ID*, which specify what happened (e.g., port up); and the time stamp of when the event occurred. In our preprocessing, we map each distinct pair of host ID and event ID into a unique event type. The datasets and their characteristics are list in Table 1.

Dataset	Number of Events	Number of Event Types	Min:Max Count
Dataset 1	1000	6	6: 480
Dataset 2	10549	312	2:1738
Dataset 3	11204	1496	1:1849
Dataset 4	46069	1784	1:2422

Table 1. Datasets characteristics. Column 4 is the range of the occurrence counts for each event type.

The experimental results are summarized in Table 2. Several factors account for the dependence behaviors in these datasets. The first factor is a result of periodic monitoring that is initiated when a high severity event occurs. The second factor is a sequence of installation policies, such as rebooting some servers every morning. The third factor is the instable network connection or server which often leads a sequence of events. We reviewed the discovered patterns with the operation staff and it turned out that many of the t-patterns related to underlying problems. We found several patterns of interest. For example, the “port up” and “port down” are loose temporal patterns that signify a mobile user’s login and logout; “router link down” followed by “Router

down” in 10 second signifies a typical router down sequence: a router “down” results in its link down. Further, a sequence of four SNMP_requests (5, 10, 15, seconds apart in sequences) raises up a security concern of port_scans in a regular way. Another stringent pattern about a performance measurement crossing critical threshold and resetting, leads to further investigation and final discovery of a wrong threshold setting problem.

Dataset	Possible Pairs	dependent pairs	Min: Max Lags
Dataset 1	36	9	0:100
Dataset 2	97344	257	0:1-day
Dataset 3	2238016	1039	0:1-day
Dataset 4	3182656	1496	0:1-day

Table 2. Experimental Results On The Datasets. Column 2 show the number of possible dependent event type pairs, Column 3 indicates the number of discovered dependent pairs and Column 4 shows the range of the lags.

5. Discussions and Conclusions

Existing temporal mining approaches faces two fundamental problems: the temporal correlation of events and the characteristics of interesting patterns. This paper proposes algorithms to find temporal patterns without predefined time windows. The problem is divided into two stages: testing the dependence and finding the lags. By formulating the dependence problem as the comparison of two probability distributions, our technique for testing the dependence resembles the distance methods in spatial point process and hence overcome the difficulties associated with traditional window techniques. The lag detection is solved with an approach based on Chi-Squared test.

References

- [1] P. Andersen, O. Borgan, R. Gill, and N. Keiding. *Statistical Models based on Counting Processes*. Springer, 1993.
- [2] M. Berman. Testing for spatial association between a point process and another stochastic process. *Applied Statistics*, 35(1):54–62, 1986.
- [3] N. A. Cressie. *Statistics for spatial data*. John Wiley & Sons, 1991.
- [4] J. L. Hellerstein, S. Ma, and C.-S. Perng. Discovering actionable patterns in event data. *IBM System Journal*, 41(3):475–493, 2002.
- [5] T. Li. *Knowledge Discovery from Labeled and Unlabeled Data*. PhD thesis, University of Rochester, 2004.
- [6] S. Ma and J. L. Hellerstein. Eventbrowser: A flexible tool for scalable analysis of event data. In *DSOM’99*.
- [7] S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *ICDE’01*.
- [8] H. Mannila, H. Toivonen, and I. Verkamo. Discovering frequent episodes in sequences. In *KDD’95*.