Correlating Events with Time Series for Incident Diagnosis

Chen Luo*
Jilin University
rackingroll@163.com

Jian-Guang Lou
Microsoft Research
jlou@microsoft.com

Qingwei Lin
Microsoft Research
qlin@microsoft.com

Qiang Fu
Microsoft Research
qifu@microsoft.com

Rui Ding
Microsoft Research
juding@microsoft.com

Dongmei Zhang
Microsoft Research
dongmeiz@microsoft.com

Zhe Wang
Jilin University
wz2000@jlu.edu.cn

ABSTRACT

As online services have more and more popular, incident diagnosis has emerged as a critical task in minimizing the service downtime and ensuring high quality of the services provided. For most online services, incident diagnosis is mainly conducted by analyzing a large amount of telemetry data collected from the services at runtime. Time series data and event sequence data are two major types of telemetry data. Techniques of correlation analysis are important tools that are widely used by engineers for data-driven incident diagnosis. Despite their importance, there has been little previous work addressing the correlation between two types of heterogeneous data for incident diagnosis: continuous time series data and temporal event data. In this paper, we propose an approach to evaluate the correlation between time series data and event data. Our approach is capable of discovering three important aspects of event-timeseries correlation in the context of incident diagnosis: existence of correlation, temporal order, and monotonic effect. Our experimental results on simulation data sets and two real data sets demonstrate the effectiveness of the algorithm.

Keywords

Correlation, Incident Diagnosis, Two-sample Problem

1. INTRODUCTION

The research in this paper is motivated by a real-world application in incident investigation of online services[20]. Unlike traditional shrink-wrapped software, online service systems are designed to run continuously and be available all time(24x7). However, during the operation of an online service, live-site service incidents (unplanned interruption/outage of the service) are often unavoidable, and can lead to significant economic loss or other serious consequences. For example, many reputable online services such as those provided by Amazon, Google, and Citrix have experienced live-site incidents during the past couple of years [1, 14]. In order to minimize service downtime caused by service incidents, much effort has been invested in improving the efficiency of service-incident diagnosis.

Diagnosis of service incidents mainly depends on analyzing the telemetry data collected at the runtime of the service [20]. Unlike a desktop application, the back-end system of an online service is often a large-scale distributed system. It is usually impractical to attach a debugger to the service to investigate service incidents. In most online systems, data analysis is the only way for engineers to diagnose service incidents. Telemetry data such as service-level logs, performance counters, and machine/process/service-level events can often provide enough information for incident diagnosis. Most telemetry data can be grouped into two categories: continuous time series data and temporal event data. A time series is a sequence of real-valued data points, measured typically at successive time points equally spaced with a uniform time interval. A typical example of time series in an online service system is the performance counter of CPU usage. An event sequence in an online service is used to record the occurrences of a specific software message indicating that something has happened in the system, e.g., an event sequence of "Out of memory" contains events of "Out of memory", which occur when there is not enough memory in the system.

Among all data-driven techniques for incident diagnosis, correlation analysis between telemetry data and system states plays a very important role [7] because correlation relationships often provide hints for causality analysis. Although the correlated metrics may not exactly be the root causes of incidents, they could be intermediate useful information that pinpoints the root causes. In practice, engineers usually start their incident investigation by searching for a small set of system metrics that are correlated to the Key Performance Indicators (KPI) of the service (Here, a KPI is an indicator that can be used to manifest system's health state, e.g., service availability or request latency) [9].

*This work is conducted during his internship at Microsoft Research.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623374.

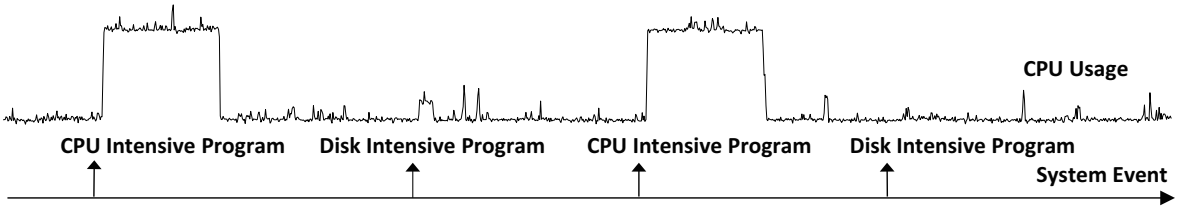


Figure 1: The relationships between CPU usage and two system tasks (*disk intensive task* and *CPU intensive task*)

Because of the importance of correlation analysis, a lot of tools have been built to analyze the correlation between time series [32] and the correlation between system events [18, 17, 2]. However, how to evaluate the correlation between a time series and an event sequence is still not well studied. Because of the heterogeneous property of these two types of data, traditional correlation analysis, such as Pearson correlation and Spearman correlation [8], often cannot provide satisfying results. Furthermore, in a large-scale system, an occurrence of an event may be related to a change of a time series during a time period rather than a point-to-point corresponding relationship in the traditional correlation analysis techniques.

In this paper, we propose an approach to evaluate the correlation between a time series and an event sequence. Motivated by the real requirement of incident management, our correlation technique tries to answer the following three questions: a. Are they correlated? b. What is the delay relationship between them? c. What is the monotonic effect between them (refer to Section 2 for the definition of monotonic effect)? In the context of service incident management, the answers to these questions can provide rich information to engineers for incident diagnosis. We formulate the correlation problem as a two-sample problem [21], and then use the nearest neighbors method to evaluate the existence of the correlation. At the same time, we identify temporal relationships and monotonic effects by analyzing the relationships between sub-series before and after the occurrence of each event(refer to Section 2).

The contributions of this paper are as follows:

1. Motivated by real applications, we articulate the correlation problem as the aforementioned three questions. To the best of our knowledge, this is the first attempt to evaluate the three aspects of correlation between a time series and an event sequence for incident diagnosis.

2. We formulate the correlation problem as a two-sample problem, and propose a novel framework to resolve the problem.

3. The experiments on simulated data and real data from Microsoft services show the effectiveness of our method.

The rest of the paper is organized as follows: The problem statement and formulation are introduced in Section 2. We propose our approach in Section 3. The Empirical evaluation is shown in Section 4. In Section 5, we introduce some related works. Finally, we conclude our work in Section 6.

2. PROBLEM STATEMENT AND FORMULATION

2.1 Problem Statement

In statistics, correlation is any statistical relationship between two random variables in which random variables do not satisfy a mathematical condition of probabilistic independence. Let us give an example first to demonstrate the correlation between event data and the time series data. Fig. 2 shows two types of events (e.g., starting a *disk intensive program and a CPU intensive program*) and a time series (e.g., *a performance counter of CPU Usage*). Every time a *CPU intensive program* starts, the performance counter of *CPU Usage* will increase significantly. On the other hand, the value of *CPU Usage* does not have any specific pattern related to the starting events of *a disk intensive program*. As a result, we can say that the starting events of *the CPU intensive program* and the performance counter of *CPU Usage* have a correlating relationship, while there is no correlation between the starting events of *a disk intensive program* and the *CPU Usage*. Such kind of correlation analysis is an important step for incident diagnosis [9, 6].

In incident diagnosis scenarios, engineers typically study the following three aspects of the correlation among telemetric data when they are investigating service incidents.

- **Existence of Dependency.** Is there a correlation between an event sequence and a time series? Although statistical correlation is not sufficient to demonstrate the presence of a causal relationship, it is still quite useful for incident diagnosis because it can often provide hints to engineers for causal analysis.

- **Temporal Order of Dependency.** Given a pair of dependent data streams, the direction of the correlation is useful information in causal analysis [23]. The temporal order of the occurrence often provides information about the cause-effect direction in the context of incident diagnosis. For example, for a given dependent pair: an event sequence E and a time series S, if

events in E always occur before corresponding significant changes of S, we can often propose a hypothesis that E might be related to a cause of the changes of S, and then take actions to verify this hypothesis based on other aspects of data from the service system.

- **Monotonic Effect of Dependency.** In many cases, some event occurrence is related to a significant value-increase (or value-decrease) of a time series. Such a positive (or negative) monotonic effect between two dependent data streams is also a very important property of correlation for causal inference [30]. For example, during diagnosis, our domain knowledge tells us that significant increase of CPU usage may cause the service to violate its quality SLA (Service Level Agreement) but the decreases of CPU usage should not.

Given an event sequence (or event type) denoted as E, the timestamps of the events are denoted as $T_E = (t_1, t_2, \cdots, t_n)$, where n is the number of events that happened. A time series, denoted as $S = (s_1, s_2, \cdots, s_m)$, where m is the number of points in the time series. Unlike the timestamps of an event sequence, the timestamps of a time series, denoted as $T_S = (t(s_1), t(s_2), \cdots, t(s_n))$, have the relationship of $t(s_i) = t(s_{i-1}) + \tau$, where τ is the sampling interval.

Here, we assume that each time series has an even sampling interval, which is true in many applications. In our analysis, we also assume that the effect of an event on a time series only lasts a certain time interval that is very small compared to the total lasting time of the time series. This assumption is valid in a real-world online service system which is in a normal and healthy state most of the time. Any incident will be resolved soon after its occurrence by the system operation team.

As mentioned before, if an event type E and a time series S have a correlation relationship, every time an event E happens, there is a corresponding change of the time series S. The potential temporal relationships between an occurrence of an event $e_i \in E$ and its corresponding change of the time series are illustrated in Fig. 2. Here, each change is represented as a sub-series of S.

Supposing $\ell_k^{rear}(S, e_i)$ denotes a sub-series of S after e_i happens with length of k, and $\ell_k^{front}(S, e_i)$ denotes a sub-series of S before the e_i happens. Intuitively, if event E does not correlate to time series S, then both $\ell_k^{rear}(S, e_i)$ and $\ell_k^{front}(S, e_i)$ $(i = 1 \cdots n)$ are not related to the occurrences of e_i. In other words, the sub-series $\Gamma^{front} = \{\ell_k^{front}(S, e_i), i = 1 \cdots n\}$ (or $\Gamma^{rear} = \{\ell_k^{rear}(S, e_i), i = 1 \cdots n\}$)are not statistically different from other sub-series with length of k (denoted as Θ) that are randomly sampled from S. On the other hand, if there is a correlation relationship between E and S, the sub-series $\Gamma^{front} = \{\ell_k^{front}(S, e_i), i = 1 \cdots n\}$ (or $\Gamma^{rear} = \{\ell_k^{rear}(S, e_i), i = 1 \cdots n\}$) will be statistically different from the randomly sampled sub-series Θ.

Based on this information, we have the following definitions.

Definition 1. *An event sequence E and a time series S are correlated and E often occurs after the changes of S (denoted as $S \rightarrow E$), if and only if the probabilistic distribution of $\{\ell_k^{front}(S, e_i), i = 1 \cdots n\}$ is statistically different from the randomly sampled sub-series Θ.*

Definition 2. *An event sequence E and a time series S are correlated and E often occurs before the changes of S (denoted as $E \rightarrow S$), if and only if the probabilistic distribution of $\{\ell_k^{rear}(S, e_i), i = 1 \cdots n\}$ is statistically different from the randomly sampled sub-series Θ and the probabilistic distribution of $\{\ell_k^{front}(S, e_i), i = 1 \cdots n\}$ is not statistically different from Θ.*

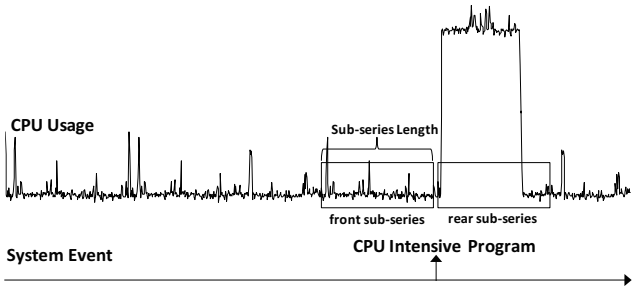


Figure 2: Example of front sub-series and rear-series

Definition 3. *An event sequence E and a time series S are correlated (denoted as $E \sim S$), if there is a relationship of $E \rightarrow S$ or $S \rightarrow E$.*

Definition 4. *If $E \rightarrow S$ (or $S \rightarrow E$) and the event occurrences of E are related to significant value-increases of S, we denote the correlation as $E \overset{+}{\rightarrow} S$ (or $S \overset{+}{\rightarrow} E$).*
If $E \rightarrow S$ (or $S \rightarrow E$) and the event occurrences of E are related to significant value-decrease of S, we denote the correlation as $E \overset{-}{\rightarrow} S$ (or $S \overset{-}{\rightarrow} E$).

2.2 Modeling as a Two-Sample Problem

Based on the above definitions, our correlation analysis problem can be transformed to a multivariate two-sample hypothesis-testing problem. Two-sample tests are commonly used when checking whether two samples come from the same underlying distribution, which is assumed to be unknown. In our context, one sample is the set Γ^{front} or Γ^{rear}, the other sample is Θ that is the set of sub-series randomly sampled from S. Each data point is actually a sub-series, which can be represented as a vector of k dimensions. Therefore, the original problem becomes a multivariate two-sample test.

Here, we use one set Γ^{front} and Θ as our example. Let Γ^{front} and Θ be independent random samples generated from unknown distributions F and G, respectively. The distributions are assumed to be absolutely continuous with respect to Lebesgue measure. Their densities are denoted as f and g, respectively. The hypotheses of the two-sample test can be stated as follows:

$$\begin{cases} H_0 & : & F = G \\ H_1 & : & F \neq G \end{cases} \tag{1}$$

If H_1 is true, that means the probabilistic distribution of Γ^{front} is statistically different from the randomly sampled sub-series Θ. In addition, as Definition 1, time series S and event E are correlated. Otherwise, if H_0 is true, then time series S and event E may not be correlated.

3. THE APPROACH

In this section, we propose a nearest-neighbor based method algorithm to analyze the three aspects of correlation, and then analyze the complexity of the proposed approach.

3.1 Nearest Neighbor Method

Multivariate two-sample tests have been of continuous interest to the statistics community. Several different algorithms of this kind have been proposed [10, 21, 28]. To utilize some structure based distance measures such as DTW [4] or DTW-D [5], we apply the nearest neighbor statistic based method [28] in our scenario. It is worth pointing out that, other two-sample tests algorithms [10, 21] can also be used in our method.

In this subsection, we take Γ^{front} as an example. Given the two samples $\Gamma^{front} = \{\ell_k^{front}(S, e_i), i = 1 \cdots n\}$ and $\Theta = \{\theta_1, \theta_2, \cdots, \theta_{\tilde{n}}\}$, and the pooled sample by $Z = \Gamma \cup \Theta$, we label the pooled sample as $Z_1, \cdots, Z_p$ with $p = n + \tilde{n}$ where

$$Z_i = \begin{cases} \ell_k^{front}(S, e_i), & if\ i = 1, \cdots, n \\ \theta_{i-n}, & if\ i = n+1, \cdots, p. \end{cases} \quad (2)$$

For a finite set of sub-series A and a sub-series $x \in A$, let $NN_r(x, A)$ denote the r-th nearest neighbor of x within the set $\{A \backslash x\}$. For two mutually exclusive subsets A_1, A_2 and a point $x \in A$, we define an indicator function:

$$I_r(x, A_1, A_2) = \begin{cases} 1, & if\ x \in A_i\ \&\&\ NN_r(x, A) \in A_i, \\ 0, & otherwise. \end{cases} \quad (3)$$

The function $I_r(x, A_1, A_2)$ indicates whether x and its r-th nearest neighbor in $A \backslash x$ belong to the same subset.

In our problem, the nearest-neighbor based tests rely on the following quantity and its generalizations:

$$T_{r,p} = \frac{1}{pr} \sum_{i=1}^{p} \sum_{j=1}^{r} I_j(x, A_1, A_2) \quad (4)$$

Here, $p = |n + \tilde{n}|$ is the size of the sample. The test statistic $T_{r,p}$ is the proportion of pairs containing two sub-series from the same sample, among all pairs formed by a sample sub-series and one of its nearest neighbors in the pooled sample Z. Intuitively $T_{r,p}$ is small under the null hypothesis when the two samples are mixed well, while $T_{r,p}$ is large when the two underlying distributions are different.

Then, as in [28], when p is large enough, $(pr)^{1/2}(T_{r,p} - \mu_r)/\sigma_r$ obeys a standard Gaussian distribution with the following parameters:

$$\mu_r = (\lambda_1)^2 + (\lambda_2)^2 \quad (5)$$

and

$$\sigma_r^2 = \lambda_1\lambda_2 + 4\lambda_1^2\lambda_2^2 \quad (6)$$

where $\lambda_1 = n/p$ and $\lambda_2 = \tilde{n}/p$. The traditional Gaussian distribution test, Γ^{front} and Θ are significantly different when: $(pr)^{1/2}(T_{r,p} - \mu_r)/\sigma_r^2 > \alpha$, where $\alpha = 1.96$ for $P = 0.025$ (or $\alpha = 2.58$ for $P = 0.001$). For more about the hypothesis test in a Gaussian distribution, please refer to [15].

3.2 Mining Existence and Temporal Order

According to the definitions in Section 2, we can summarize that if event E and time series S are correlated, the following statement must be true: the front sub-series Γ^{front} and randomly sampled sub-series Θ are statistically different; or the rear sub-series Γ^{rear} and randomly sampled sub-series Θ is statistically different. In our approach, we use the above nearest-neighbor method to determine whether two sets of sub-series have an identical distribution.

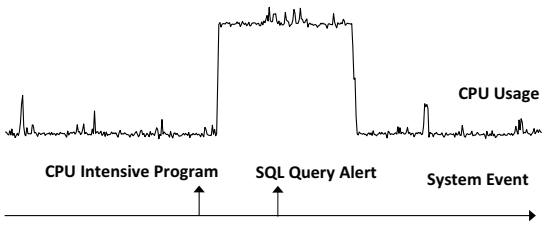


Figure 3: Example of temporal order, *CPU intensive program* → *CPU usage* and *CPU usage* → *query alert*

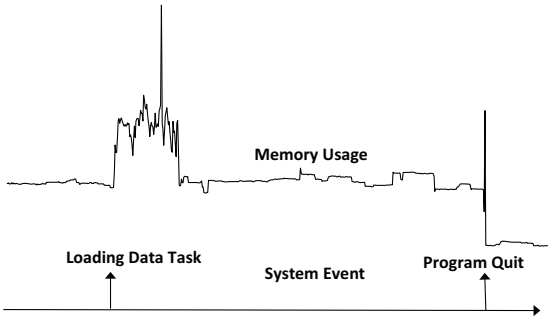


Figure 4: Effect examples: *loading data task* $\stackrel{+}{\rightarrow}$ *memory* and *program exiting* $\stackrel{-}{\rightarrow}$ *memory*

If the front sub-series Γ^{front} and randomly sampled sub-series Θ are statistically different, then $S \rightarrow E$. Otherwise, if the rear sub-series Γ^{rear} and Θ is statistically different, then $E \rightarrow S$. Fig. 3 shows an example of Correlation Temporal Order. From Fig. 3, we can see CPU Intensive Program → CPU Usage, and CPU Usage → SQL Query Alert.

3.3 Mining Effect Type

As explained in Definition 4, in order to determine the monotonic effects such as $E \stackrel{+}{\rightarrow} S$ (or $E \stackrel{-}{\rightarrow} S$) of correlation, we need to check whether there exists a significant value increase (or decrease) on the time series S after an event E happens. Similarly, it can also be formulated as a problem of statistical hypotheses testing. In this research, we use $t - test$ [15] to check whether there is a significant value increase (or decrease) from Γ^{front} to Γ^{rear}.

Here, the t_{score} between Γ^{front} and Γ^{rear} can be calculated by the following equation:

$$t_{score} = \frac{\mu_{\Gamma^{front}} - \mu_{\Gamma^{rear}}}{\sqrt{\frac{(n_1-1)\sigma^2_{\Gamma^{front}} + (n_2-1)\sigma^2_{\Gamma^{rear}}}{n_1+n_2-2}\left(\frac{1}{n_1} - \frac{1}{n_2}\right)}} \quad (7)$$

Algorithm 1: The Overall Algorithm

Input: Event $E = (e_1, e_2, ..., e_n)$, and Time Series $S = (s_1, s_2, ..., s_m)$, and the sub-series length k.
Output: The correlation flag C, the direction D, and the effect type T

Initialize Γ^{front} and Γ^{rear};
Initialize Θ;
Initialize $R = false$, $D = NULL$, $T = NULL$;
Normalize each $\ell_k^{front}(S, e_i)$ and $\ell_k^{rear}(S, e_i)$.;
Test Γ^{front} and Θ using Nearest Neighbors Method. The result is denoted as D_f.;
Test Γ^{rear} and Θ using Nearest Neighbors Method. The result is denoted as D_r.;
if $(D_r == true \&\& D_f == false)$ **then**
    $R = true$.;
    Calculate t_{score} using Equation (8).;
    if $(t_{score} > \alpha)$ **then**
        $T = E \stackrel{-}{\rightarrow} S$.;
    else if $(t_{score} < -\alpha)$ **then**
        $T = E \stackrel{+}{\rightarrow} S$.;
else if $(D_r == false \&\& D_f == true)$ || $(D_r == true \&\& D_f == true)$ **then**
    $R = true$.;
    Calculate t_{score} using Equation (8).;
    if $(t_{score} > \alpha)$ **then**
        $T = S \stackrel{-}{\rightarrow} E$.;
    else if $(t_{score} < -\alpha)$ **then**
        $T = S \stackrel{+}{\rightarrow} E$.;
Out put R, D and T;
Algorithm End.

where $\mu_{\Gamma^{front}}$ and $\mu_{\Gamma^{rear}}$ are the mean values of Γ^{front} and Γ^{rear}. $\sigma_{\Gamma^{front}}$ and $\sigma_{\Gamma^{rear}}$ are their variance values. In our research, $n_1 = n_2 = n$, and n is the event number in E, thus Equation 7 can be reduced to:

$$t_{score} = \frac{\mu_{\Gamma^{front}} - \mu_{\Gamma^{rear}}}{\sqrt{\frac{\sigma^2_{\Gamma^{front}} + \sigma^2_{\Gamma^{rear}}}{n}}} \quad (8)$$

Then, if $t_{score} > \alpha$, we have a negative monotonic effect (e.g, $E \stackrel{-}{\rightarrow} S$ or $S \stackrel{-}{\rightarrow} E$); and if $t_{score} < -\alpha$, we have a positive monotonic effect (e.g., $E \stackrel{+}{\rightarrow} S$ or $S \stackrel{+}{\rightarrow} E$). Here, $\alpha = 1.96$ for $P = 0.025$ (or $\alpha = 2.58$ for $P = 0.001$) [15].

At the same time, for a given α,if $|t_{score}| < \alpha$, we cannot determine the effect type with a high level of confidence. However, such situations are seldom found in the real scenarios of incident diagnosis. Most services have be designed with a large certain margin to fault-tolerate. When a service incident is detected, the corresponding time-series must have a significant difference from its normal behavior (i.e., $t_{score} > \alpha$) [9].

3.4 The Overall Algorithm

The overall algorithm is then summarized as Algorithm 1. This algorithm implements the methods we have introduced in the above subsections. It performs two two-sample tests based on the sub-series Γ^{front}, Γ^{rear}, and Θ. The output of this algorithm contains all three aspects of the correlation between a time series and an event sequence. In this section, we discuss details of the algorithm implementation.

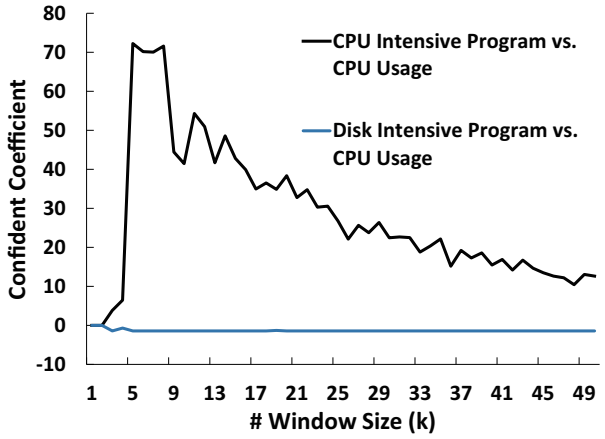
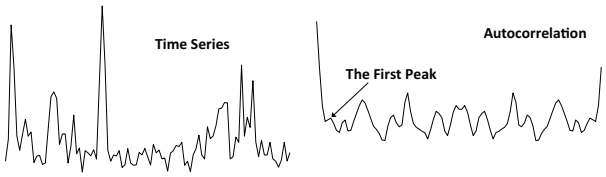


Figure 5: Confidence vs. sub-series length k.

3.4.1 Setting Parameters

The length of sub-series k is an important parameter which can directly influence the performance of our algorithm. We use one example to illustrate the influence of k. Fig. 5 shows two curves, where the vertical axis represents the value of $(Nk)^{1/2}(T_{k,N} - \mu_k)/\sigma_k^2$. For a given α, its value is directly related to the confidence of hypotheses H_1. A larger $(Nk)^{1/2}(T_{k,N} - \mu_k)/\sigma_k^2$ means a high confidence of H_1. We denote it as "Confidence Coefficient". In Fig. 5, the black curve is the confidence coefficient obtained by evaluating the correlation between a pair of "CPU Extensive Program" and "CPU Usage". We know there does exist any correlation between the pair of data. This means the higher confidence of H_1 is better. Because the front sub-series (or the rear sub-series) with a very small length (i.e., $k < 4$) only contains limited information, the confidence in testing is not high enough. As the length increases, we can see a significant increase in confidence in Fig. 5. However, when the length is larger than a certain value (e.g., 8), the confidence decreases after the maximum value. The reason is that as the sub-series length increases thereby the effect of event to time series starts to diminish which makes the subsequence closer to the population Θ. In Fig. 5, the blue curve is the confidence coefficient obtained from the pair of "Disk Transfer Extensive Program" and "CPU Usage". Domain knowledge clearly tells us this pair is not correlated. The result also confirms this fact. The curve of confidence coefficient is flat and very close to 0 for any value of k. It means the rejection of H_1 is not influenced by k if two data sets are not correlated.

In some cases, the value of k can be selected based on domain knowledge and experiments. However, in most real world situations, there are millions of time series and events, and we do not have enough domain knowledge to pre-select the values of all sub-series lengths. In this research, we design a method to auto-select the sub-series length for a time series based on the autocorrelation function [12] of the time series. Given a time series $S = (s_1, s_2, ..., s_n)$, the autocorrelation is showed as follows:



(a) Time Series (b) Autocorrelation

Figure 6: Example of a time series and the corresponding autocorrelation.

$$R(l) = E(s_i * s_{i-l}). \quad (9)$$

where l denotes the lag of the correlation. The autocorrelation function of a time series can be used to represent the energy of signals in the time series with a period of l [12]. Therefore, our length k can be assigned the value of the first peak to include the significant signal of the time series. The experiment (Section 4.5) shows the effectiveness of this method and Fig. 6 shows an example.

Besides k, the number of neighbors r is also a very important parameter. It has been demonstrated that $r = ln(p)$ is a good choice in literatures [28]. In this paper, we directly follow their suggestion.

4. EMPIRICAL EVALUATION

In this section, we make an empirical evaluation of our algorithm by performing a set of experiments on a data set from a controlled environment and two real data sets.

4.1 Baseline

In order to evaluate the effectiveness of our algorithm, we choose two baseline algorithms in our experiment. The first one is the Pearson correlation [8], which is the widely used method for correlation mining in time series. The second baseline is *J-measure* [24], which is a widely used method for correlating event data [16]. In the rest of this section, we brief introduce these two baseline algorithms.

4.1.1 Pearson Correlation

The Pearson correlation method is one of the most widely used methods for measuring the correlation between two time series. The Pearson correlation coefficient, denoted as ρ. is calculated as follows:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

where *cov* is the covariance, σ_X is the standard deviation of X, μ_X is the mean of X and $E[*]$ denotes the expectation.

The Pearson correlation can not be directly used in the correlation evaluation between time series and event data. In order to make it comparable, we transform the event E to a time series, denoted as $S^E = (s_1^E, s_2^E, ..., s_m^E)$, where m is the size of time series S, and s_i^E is calculated as:

$$s_i^E = \begin{cases} 1, & if\, t(s_i) \in T_E \\ 0, & otherwise. \end{cases}$$

where $t(s_i)$ and T_E is the same meaning as before (see section 2.1).

After transforming the event data to the time series form, we directly calculate the Pearson correlation coefficient ρ between the S^E and S. As introduced in [8], due to our application background, we assign the threshold as 0.1 (weak correlation as in [8]). If $\rho > 0.1$, then $E \overset{+}{\sim} S$. And if $\rho < -0.1$, then $E \overset{-}{\sim} S$. Else ($|\rho| < 0.1$), E and S are not correlated.

4.1.2 J-measure Correlation

We choose the widely used event correlation method, *J-measure* [24], as the second baseline. J-measure has been used in many research efforts [16].

However, *J-measure* is proposed for the event correlation and can not be used directly. Therefore, in our experiment, we first transform each time series to an event sequence as follows: Given a time series $S = (s_1, s_2, ..., s_m)$, its corresponding event sequence is $E^S = (e_1^E, e_2^E, ..., e_n^E)$, where each e_i^E is the change point in the time series, and n is the number of changes in the time series. In this research, we use a widely used GLR algorithm [3] to detect changes in time series.

After transforming each time series to its corresponding event sequence, we can obtain a complete event sequence by combining all the event sequences from two sets: (1) event sequences transformed by time series, (2) event sequences from the original data set. Then we can directly use *J-measure* to evaluate the correlation between these complete event sequences. Similar to the first baseline (Pearson correlation), we assign a correlation threshold as 0.1 (weak correlation in [8]). If the *J-Measure* is larger than 0.1, then $E \sim S$. Otherwise, E and S are not correlated. Considering this, the Temporal Order and the Monotonic Effect Type of the correlation can not be determined using this method.

4.2 Effectiveness Study on Simulation Datasets

In this section, we introduce the experiment on the data from a controlled environment.

4.2.1 The Controlled Experiment

In this research, we setup a SQL environment for a controlled experiment to demonstrate the mutual influencing behavior in a system. For example, when a *CPU Intensive program* starts, the performance counter of *CPU usage* may increase. In a SQL server, database operations can be influenced by the increase of *CPU usage* or *memory usage*. In the experiment, three types of programs including *Disk Intensive program*, *CPU Intensive program* and *Memory Intensive program* are launched several times randomly. Each running lasts for an interval of $[5, 15]$ minutes. At the same time, we query a SQL server database every second. If the query latency exceeds the maximum latency limitation ($200ms$), the query will trigger a time-out alert.

We conduct the experiment for 6 hours. The *CPU Intensive program* run 43 times; the *Memory Intensive program* run 41 times; the *Disk Intensive program* run 38 times ; and 285 *Query Alerts* are triggered. At the same time, we collect four system events (e.g., the starting events of each program and query time-out alerts) and three system performance counters (CPU usage, memory usage, and disk I/O usage) as showed in Table 1.

Table 1: Time series and event data in the controlled experiment

Name	Type	Description
CPU Intensive program	Event	A multi-thread process, which will let the CPU Usage achieve nearly 90%
Memory Intensive program	Event	A process that will apply for a nearly 2G memory space
Disk Intensive program	Event	A copy files process which can sharply increase disk transfer rate
Query Alert	Event	When SQL query delay exceed the maximum limit, an alert occurrence.
CPU Usage	Time Series	record the CPU usage every second
Memory Usage	Time Series	record the Memory usage every second
Disk Transfer Rate	Time Series	record the Disk Transfer Rate every second

4.2.2 Result and Analysis

After obtaining the data from the controlled environment, we run the three algorithms on this data. The results produced by the proposed algorithm and two baseline algorithms are showed in Table 2.

The sub-series length k of the three time series (CPU usage, Memory Usage, Disk Transfer Rate) is assigned using the first peak of the autocorrelation (as introduced in section 3.4.1). We choose DTW distance [4] as the distance measure in Nearest neighbors method of our algorithm.

In Table 2, we see that the correlation relationship, produced using our method, can reflect real system behavior: the *CPU Intensive program* has a positive effect on the CPU usage; the *Memory Intensive program* has a positive effect on both Memory usage and CPU usage; the *Disk Intensive program* has a positive effect on the Disk Transfer rate. The *Query Alert* depends on the high usage of CPU and Memory.

On the other hand, in the results of the Pearson Correlation Algorithm, some of the correlation relations are lost (Memory Using Task $\xrightarrow{+}$ CPU usage, and Memory Usage $\xleftarrow{+}$ Query Alert). The result produced by *J-Measure* is very bad, because *J-Measure* can only encode the occurrence between event and changes. In addition, the Temporal Order and the Monotonic Effect of Dependencies can not be calculated using *J-measure*. From this view, our algorithm has great advantages compared to the baseline algorithms.

4.3 Effectiveness Study on Real Datasets

In this section, we will compare the proposed algorithm with the baseline algorithms on two real data sets.

4.3.1 Dataset

The first real world dataset is a System Monitoring Dataset. This dataset is collected from a large scale online service of Microsoft. We collect 24 different kinds of time series as our input time-series from a number of machines including performance counters of Memory usage, CPU usage, Physical Disk usage etc. The sampling interval of each time series is 5 minutes. At the same time, we also collect a set of event sequences. Each event sequence contains the starting events of a specific job that is automatically submitted by a timer in the system. There are 52 different event in this dataset. The ground truth of this data is labeled by a software engineer in the product team.

The second real world dataset is the Custom Support Dataset. In Microsoft, the customer support team is in charge of the support requests from customers about the above online service. Customers may send us support requests when they do not know how to use the service or they encounter some service quality issues. In this paper, we only collect the requests caused by service issues. The requests are grouped to different categories based on their problem phenomena and root causes (also called *support topic*) by the support team. The occurrences of requests in a single category form an event sequence in this study, because we try to understand what kind of service telemetry data is related to a specific request topic. There are 57 different events in this dataset. The time series data in this study is collected from the runtime system. The front end of our system is a Web Server. For every *HTTP* request from an end user, we have a *HTTP* status code. We group the requests from every hour based on their *HTTP* status code into 7 groups: "200 $\sim$ 300" (it means the status code is in the range from 200 to 300.), "300 $\sim$ 400", "400 $\sim$ 500", "500 $\sim$ 600", "$>$ 600", "*Exception*" (it means no *HTTP* response, the program throws an exception.) and "*Sum*" ("*Sum*" means all the status code larger than "300"). Each group will form a time series to record the number of requests hourly.

4.3.2 Evaluation Metric

The basic model of our algorithm is a hypothesis-test model , we use the F_1 score to evaluate our method, which has been used in many research works [26] for analyzing hypothesis-test accuracy. As introduced in [26], F_1 score can be calculated as follow:

$$F_1 = \frac{2 * TruePositive}{2 * TruePositive + FalseNegative + FalsePositive}$$

In the above, $TruePositive = 1 - FalseNegative$.

Given a time series S and an event E, for the test of dependency existence, a $FalseNegative$ is a case where the algorithm result shows $S \sim E$, but actually S and E are not correlated. A $FalsePositive$ is a case where S and E are not correlated, but the result shows that $S \sim E$. For the test of temporal ordering, a $FalsePositive$ is a case where the result shows $S \rightarrow E$, while the actual temporal order is $E \rightarrow S$. A $FalseNegative$ is a case where the result does not show $S \rightarrow E$, while the actual temporal order is $S \rightarrow E$. For the Monotonic Effect Problem, a $FalsePositive$ is a case that the result shows a positive monotonic effect while the actual effect type is not positive. A $FalseNegative$ is a case that the result does not show a positive monotonic effect while the actual effect type is positive.

4.3.3 Result Analysis

We choose three distance measures for the nearest neighbor method of our algorithm, including: DTW [4] , L_1 distance and L_2 distance. The result is shown in Table 3.

For the different distance measures, we can see that there is no significant difference among them. It seems that the DTW distance has a slightly better performances than others in the experiment on Custom Support Data. When a

Table 2: Results of the data from controlled environment

Name	Proposed Method			Pearson Correlation			J-Measure		
	CPU	Memory	Disk	CPU	Memory	Disk	CPU	Memory	Disk
CPU Intensive Program	$\overset{+}{\rightarrow}$	NC	NC	$\overset{+}{\sim}$	NC [1]	NC	NC	$\sim$	$\sim$
Memory Intensive Program	$\overset{+}{\rightarrow}$	$\overset{+}{\rightarrow}$	NC	NC	$\overset{+}{\sim}$	NC	NC	$\sim$	$\sim$
Disk Intensive Program	NC	NC	$\overset{+}{\rightarrow}$	NC	NC	$\overset{+}{\sim}$	NC	$\sim$	$\sim$
Query Alert	$\overset{+}{\leftarrow}$	$\overset{+}{\leftarrow}$	NC	$\overset{+}{\sim}$	NC	NC	NC	$\sim$	$\sim$

Table 3: Result in real data set

Data Set	Methods	Existence	Temporal Order	Effect Type
		F_1 Score	F_1 Score	F_1 Score
System Monitoring Data	Correlation Mining (L1)	0.7916	0.8020	0.8016
	Correlation Mining (L2)	**0.8205**	0.7612	**0.8780**
	Correlation Mining (DTW)	0.7962	**0.8021**	0.8210
	Pearson Correlation	0.6974	N/A [2]	0.6732
	J-Measure	0.6148	N/A	N/A
Custom Support Data	Correlation Mining (L1)	0.7915	0.7659	0.7204
	Correlation Mining (L2)	0.8423	0.7870	0.8334
	Correlation Mining (DTW)	**0.8631**	**0.8205**	**0.8532**
	Pearson Correlation	0.6030	N/A	0.6501
	J-Measure	0.7398	N/A	N/A

customer encounters a problem, s/he may not call Microsoft immediately. Many of them often try several times to make sure they cannot fix by themselves before they call in. Different customers have different delay time before their calls. DTW distance can decrease the influence of latency values [5].

By comparing our algorithm with the baseline algorithms (Pearson correlation and *J-Measure*), we can see that our algorithm has significant advantages.

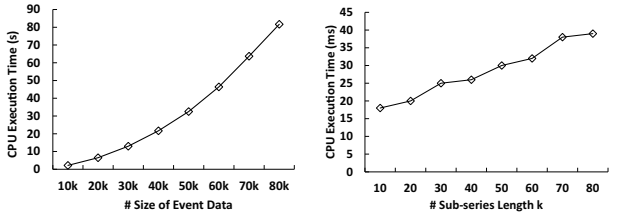
4.4 Efficiency Study

Now, we shall study the efficiency of our algorithm. We use synthetic datasets to evaluate the efficiency of our algorithm, because we can manipulate the size of the dataset flexibly.

In Fig. 7(a), we fix the value of sub-series length k, and vary the event data size n. We can see that the CPU execution time increased by enlarging the data size. Because we need to calculate the pair-wise distance among sub-series, the computational cost is at $O(n^2)$. In Fig. 7(b), we fix the size of event data, and vary the value of length k. Based on the results, we can see that the running time of the program is almost linear to the value of length k (note: we use L_2 as our distance measure in this experiment.).

4.5 Length of Sub-series

In this subsection, we study the influence caused by different length values of sub-series k in the algorithm. We select two event-time series pairs which are really correlated (CPU Extensive Program and CPU usage; Memory Extensive Program and Memory Usage). In this experiment, we use the confidence coefficient as an indicator to evaluate the performance of the analysis (as we have introduced in section 3.4.1). The results are shown in Fig. 8.



(a) Varying event data size (b) Varying sub-series length

Figure 7: Efficiency by varying data size and sub-series length

The first peak of the autocorrelation of CPU usage appears at the index of 5, as shown in Fig. 8(b). In Fig. 8(a), the largest value of the Confidence coefficient appears just at the index of $k = 5$. Similarly, in Fig. 8(d), the first peak of the autocorrelation of Memory Usage appears is at the index of 11. In Fig. 8(c), the value of the Confidence Coefficient is also at a peak when $k = 11$. From these results, we can see that by using the first peak of the autocorrelation, one can obtain a good configuration of the length k.

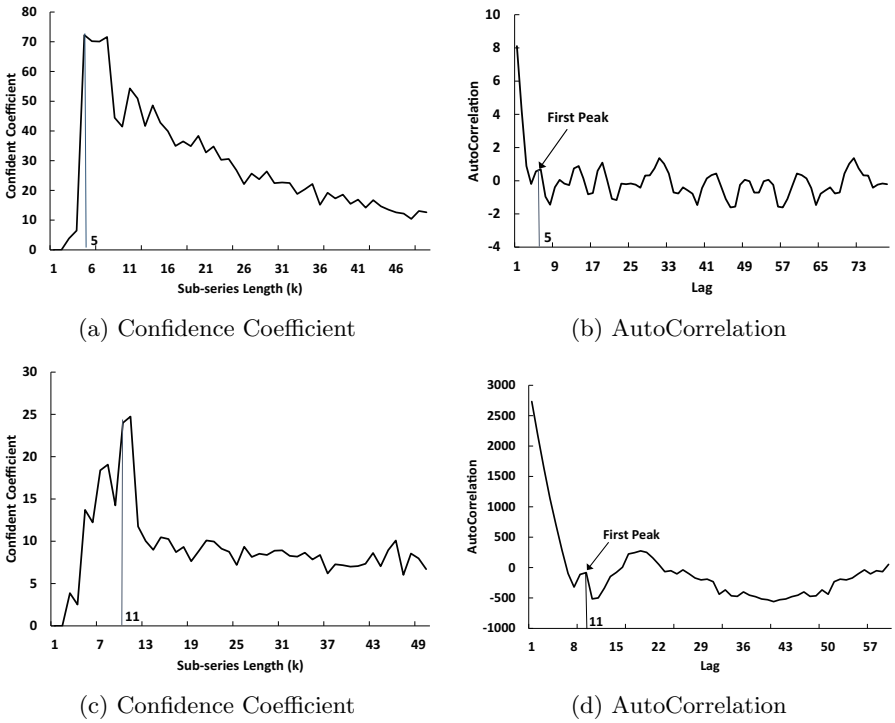
5. RELATED WORK

In this section, we briefly introduce some related research efforts.

5.1 Correlation between Time Series Data

The correlation between two time series has been widely studied, and some of them have been included in text books [15]. The Pearson correlation [8] is a basic correlation measure

[1]"NC" denotes not correlated
[2]"N/A" denotes no result.



(a) Confidence Coefficient (b) AutoCorrelation

(c) Confidence Coefficient (d) AutoCorrelation

Figure 8: The confidence coefficient value vs. sub-series length k, (a) (b):*CPU intensive program* and *CPU usage*, (c) (d): *memory intensive program* and *memory usage*

between time series, which has been widely used in practice [32]. Some extensions of the Pearson correlation are also widely used. For example, lagged correlation is an extension to correlate a lagged dataset with another unlagged dataset using the Pearson product-moment method. In [31], the author uses the lagged-correlation to estimate the lead relationship between a set of time series. Because the Pearson correlation is sensitive outliers in data set, Spearman Rank correlation and Kendall Rank correlation have been used in some scenarios [27] to overcome the drawbacks of Pearson correlation. In the Spearman correlation, data is first sorted and ranked, e.g., rank 1 is assigned to the lowest value. The Spearman Rank correlation is calculated by taking the Pearson product-moment correlation of the ranks in the dataset. The Kendall correlation tries to measure the similarity of the orderings in the dataset. Because there is no ordering relationship among the different events, the above rank based algorithms cannot be directly used in our scenario.

5.2 Correlation between Event Data

Correlation between events have also been studied in many works [13, 19, 25, 22, 11]. Here, correlation mainly means the co-occurrence of different types of events. Because event data generated in online services can naturally be regarded as sequences, association rule mining algorithms [13] for sequence data can be directly used to analyze the correlation among event data. For example, [19], the authors try to construct the dependency relationships between different system components by mining the co-occurrence among the unconstructed log events generated from the system. In [22], the authors add user guided information during the event correlation mining. Therefore, this algorithm allows users to effectively navigate the result. Correlation Mining algorithms have also been used in some natural science. In [25], the authors apply a correlation mining algorithm to mine the correlation between climate events by analyzing the historical climate data of the North Atlantic and China.

5.3 Two-sample Test

Two-sample Test is an important part of our work. For univariate data, the classical two-sample test includes the nonparametric Kolmogorov-Smirnov test and Mann Whitney U test [21]. For multi-variant data, many methods have been proposed in previous works [28, 10, 29]. In [28], the author uses the proportion of all r nearest neighbors in which observations and their neighbors belong to the same sample as a statistic. The algorithm proposed in [10] is a kernel method for the two sample problem, this method maps the distance between two observations into a reproducing kernel Hilbert space (RKHS), and then uses Maximum Mean Discrepancy to solve the problem. They further propose a unifying framework by linking the two classes of statistics (i.e., energy distances in statistics and distances between distributions in RKHS) in [29]. In this paper, we use the nearest neighbor based algorithm in [28] for simplicity. In fact, the kernel based algorithms can also be applied in our scenarios.

6. CONCLUSION AND FUTURE WORKS

In this paper, we have investigated the problem of correlation mining between time series data and event data. We have defined the correlation problem and proposed a novel approach which is able to discover three important

aspects of event-time-series correlation: existence of correlation, temporal ordering, and monotonic effect. In our approach, we first transform the correlation problem to a two-sample problem, use the nearest neighbors method to solve such a two-sample problem and then evaluate the correlation existence. To the best of our knowledge, this is the first work to answer the three correlation questions between event sequences and time series data for incident diagnosis. The experiments on the data from a controlled environment and two real datasets demonstrate the effectiveness and efficiency of our algorithm. The proposed method has already been implemented to an internal tool-set designed for service problem diagnosis.

Our current method does not consider the event combination problem, that only a certain combination of multiple events rather than a single type of event is correlated to a time series. This problem is much more complex, and will be a part of our future work.

7. REFERENCES

[1] Amazon's s3 cloud service turns into a puff of smoke. *Information Week*, Aug 2008.

[2] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enter-prise network services via inference of multi-level dependencies. In *SIGCOMM*, 2007.

[3] M. Basseville, I. V. Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

[4] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Knowledge Discovery and Data Mining*, pages 359–370, 1994.

[5] Y. Chen, B. Hu, E. Keogh, and G. E. Batista. Dtw-d: time series semi-supervised learning from a single example. In *KDD*, pages 383–391. ACM, 2013.

[6] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *OSDI*, pages 231–244, 2004.

[7] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, , and A. Fox. Capturing, indexing, clustering, and retrieving system history. In *Proc. SOSP*, pages 105–118, 2005.

[8] J. Cohen. Statistical power analysis for the behavioral sciences. 1988.

[9] Q. Fu, J.-G. Lou, Q.-W. Lin, R. Ding, Z. Ye, D. Zhang, and T. Xie. Performance issue diagnosis for online service systems. In *SRDS*, October 2012.

[10] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. volume 19, page 513. MIT; 1998, 2007.

[11] B. Gruschke et al. Integrated event management: Event correlation using dependency graphs. In *Proc. DSOM 98*, pages 130–141, 1998.

[12] J. D. Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[13] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[14] J. N. Hoover. Outages force cloud computing users to rethink tactics. *Information Week*, Aug 2008.

[15] R. A. Johnson and D. W. Wichern. *Applied multivariate statistical analysis*. Pearson, 2007.

[16] S. Kandula, R. Chandra, and D. Katabi. What's going on? learning communication rules in edge networks. *SIGCOMM*, 38(4):87–98, 2008.

[17] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *Proc. SIGCOMM*, pages 243–254, 2009.

[18] J.-G. Lou, Q. Fu, Y. Wang, and J. Li. Mining dependency in distributed systems through unstructured logs analysis. *SIGOPS Operating Systems Review*, 41(1):91–96, 2010.

[19] J.-G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu. Mining program workflow from interleaved traces. In *KDD*, pages 613–622. ACM, 2010.

[20] J.-G. Lou, Q. Lin, R. Ding, Q. Fu, D. Zhang, and T. Xie. Software analytics for incident management of online services: An experience report. In *ASE*. ACM, November 2013.

[21] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60, 1947.

[22] H. R. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah. Event correlation for process discovery from web service interaction logs. *VLDBJ*, 20(3):417–444, 2011.

[23] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[24] G. Piateski and W. Frawley. *Knowledge discovery in databases*. MIT press, 1991.

[25] S. C. Porter and A. Zhisheng. Correlation between climate events in the north atlantic and china during the last glaciation. *Nature*, 375:305–308, 1995.

[26] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness & correlation. *JMLT*, 2(1):37–63, 2011.

[27] B. Rosner. *Fundamentals of biostatistics*. Cengage Learning, 2010.

[28] M. F. Schilling. Multivariate two-sample tests based on nearest neighbors. *Journal of the American Statistical Association*, 81(395):799–806, 1986.

[29] D. Sejdinovic, A. Gretton, K. Fukumizu, and B. K. Sriperumbudur. Hypothesis testing using pairwise distances and associated kernels. In *ICML-12*, pages 1111–1118, 2012.

[30] T. J. VanderWeele and J. M. Robins. Signed directed acyclic graphs for causal inference. *Journal of the Royal Statistical Society*, 72(1):111–127, 2010.

[31] D. Wu, Y. Ke, J. X. Yu, S. Y. Philip, and L. Chen. Detecting leaders from correlated time series. In *Database Systems for Advanced Applications*, pages 352–367. Springer, 2010.

[32] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369. VLDB Endowment, 2002.