Journal of the Korean Institute of Industrial Engineers
Vol. 41, No. 6, pp. 540-550, November 2015.
ISSN 1225-0988 | EISSN 2234-6457
http://dx.doi.org/10.7232/JKIIE.2015.41.6.540
© 2015 KIIE
<Application Research>

프로세스 마이닝을 위한 거리 기반의 API(Anomaly Process Instance) 탐지법

전대욱 · 배혜림†

부산대학교 산업공학과

Detection of API(Anomaly Process Instance) Based on Distance for Process Mining

Daeuk Jeon · Hyerim Bae

Department of Industrial Engineering, Pusan National University

There have been many attempts to find knowledge from data using conventional statistics, data mining, artificial intelligence, machine learning and pattern recognition. In those research areas, knowledge is approached in two ways. Firstly, researchers discover knowledge represented in general features for universal recognition, and secondly, they discover exceptional and distinctive features. In process mining, an instance is sequential information bounded by case ID, known as process instance. Here, an exceptional process instance can cause a problem in the analysis and discovery algorithm. Hence, in this paper we develop a method to detect the knowledge of exceptional and distinctive features when performing process mining. We propose a method for anomaly detection named Distance-based Anomaly Process Instance Detection (DAPID) which utilizes distance between process instances. DAPID contributes to a discovery of distinctive characteristic of process instance. For verifying the suggested methodology, we discovered characteristics of exceptional situations from log data. Additionally, we experiment on real data from a domestic port terminal to demonstrate our proposed methodology.

Keywords: Anomaly process instance, Process mining, Arc matrix

1. Introduction

오랜 기간 축적된 데이터로부터 지식을 얻기 위한 노력은 다양한 산업과 영역에서 진행되어 왔으며, 여러 분석방법이 개발되었다. 분석을 위하여 과거에는 잘 정의된, 목적이 분명한 데이터를 사용하였으며, 이러한 데이터로부터 지식을 추출하는 데에는 통계적 방법을 통해 의미를 도출하고 활용하는 사례가 많이 존재하였다. 최근에는 셀 수 없이 많은 디지털 디바이스로부터 발생하는, 복잡한 연관성을 가진 데이터 혹은 측정하기 어려운 형태의 데이터 즉, 비정형 데이터도 분석의 대상이 되었다(Laxhammar, 2014).

비정형 데이터가 쌓이는 정보시스템을 이용하는 것은 대다수의 디지털 디바이스를 이용하는 사용자에게 일반적이다. 정형 데이터의 경우는 일반적으로 필요한 지식을 찾기 위해 수치화, 그래픽화 등의 방법을 활용하는 것이 용이하다. 그러나 비정형 데이터는 일관된 목적이 존재하지 않으므로, 여러 데이터 속성 사이의 연관성 및 의존성을 나타내기 어렵다. 이러한 비정형 데이터로부터 각각의 속성들 사이의 연관성을 통해 지식을 표현하는 것은 방대한 데이터 속에서 다양하게 도출되고 또한 여러 가지 방향으로 해석이 가능하다.

최근에 빅데이터가 각광을 받게 된 것은 비정형 데이터의 방대한 축적으로부터 비롯되었다고 볼 수 있다. 빅데이터가 대

This research was supported by a National Research Foundation of Korea(NRF) grant funded by the Korean government (MEST)(No.NRF-2015R1D1A1A09061331).
† 연락저자 : 배혜림 교수, 46241 부산광역시 금정구 부산대학로63번길 2 (장전동) 제10공학관 10623호, Tel : 051-510-2733, Fax : 051-512-7603, E-mail : hrbae@pusan.ac.kr
2015년 3월 20일 접수; 2015년 9월 23일 수정본 접수; 2015년 9월 30일 게재 확정.

량으로 축적되는 환경은 일반 사용자가 비정형 데이터를 축적하는 영역과 IoT(Internet of Things)를 위한 장비들이 실시간으로 수집하는 영역으로 확대되고 있다. 최근에는 이러한 빅데이터 기술이 생산 현장에서도 중요하게 여겨지고 있는데 이는 글로벌 경제권에서 가열되는 경쟁구도가 더욱 높은 경쟁력을 요구하기 때문이다. 경쟁력을 높이기 위해 기존에는 노동 생산성 향상, 새로운 생산 시스템의 도입, 기술력의 발달 등이 중요했다. 그리고, IT(Information Technology)가 발달함에 따라 데이터의 활용도를 높이면서 생산성을 향상하기 위한 방법을 연구하였고, 그러한 방편에서 로그 데이터를 활용하여 개선점을 찾으려는 시도가 활발하였다(Çiflikli and Kahya-Özyirmidokuz, 2010; Sim *et al*., 2012; van der Aalst *et al*., 2007).`

프로세스 마이닝(Process Mining)은 정보시스템이나, 각종 디바이스가 생성한 이벤트 로그를 수집하여 프로세스 모델 및 패턴분석을 용이하게 하도록 그 방법을 제공하고 있다. 데이터의 속성 중에서 각각의 Case ID, Activity, Timestamp, Resource 등을 지정하여 순차적인 프로세스 모델을 도출하면, 이벤트 로그를 발생시킨 정보시스템의 특성을 해당 프로세스 모델로부터 분석하게 된다. 주된 업무 흐름을 파악하고자 하는 사용자에게 신속하게 프로세스 모델을 제공하여, 쉽게 이해할 수 있도록 하는 방법이다. 물론, 데이터의 전처리 과정에서의 복잡성 및 시간지연 문제는 데이터를 수집하는 과정에서의 시스템 및 데이터베이스 이해도에 따라 편차가 있을 수 있다. 순차적인 이벤트의 정보를 담고 있는 이벤트 로그는 전통적인 통계에서 활용하는 시계열 데이터와는 차이가 있다. 특히, 프로세스 마이닝에서 활용하는 이벤트 로그는 작업이 종료된 데이터를 활용하기 때문에 실시간 운영 지원과 관련해서는 활용에 제약이 있다.

도출된 프로세스 모델이 모든 업무흐름의 경우를 나타낸다면 매우 복잡한 형태의 스파게티 모델을 가지게 된다. 이러한 스파게티 모델은 분석의 의미를 반감시키기 때문에 다수의 마이닝 알고리즘은 추상화 과정을 가지고 있다. 추상화 된 이해도가 높은 프로세스 모델로부터 전체 이벤트 로그의 추가적인 분석을 고려할 수 있다(Rebuge and Ferreira, 2012). 이러한 분석과정에서 일부의 로그 데이터는 결과물에서 제외되는 현상이 발생한다. 이벤트 로그에 존재한다는 것은 실제 발생한 이벤트이며, 동시에 그 이벤트의 중요성이 횟수 혹은 기타 알고리즘에 의해 제한 받지 않을 수 있음을 나타낸다. 수많은 이벤트 중에 단 한 번 발생하였지만 응급한 경우, 외부 침입에 의한 경우, 부당한 처리의 경우 등을 발견할 수 있다. 이러한 발견이 의미적으로 중요한지의 여부는 전문가가 판단할 수 있지만, 전문가가 시스템에서 찾아내기 위해 복잡한 정보시스템을 이해한다는 것은 별개의 문제이다. 특히 여러 시스템이 엮여서 동일 혹은 유사한 데이터를 쏟아내는 현실에서 예외적인 경우를 찾아내는 것은 많은 전문적인 지식을 동원하여야 한다(van der Aalst, 2011).

본 연구는 이벤트 로그로부터 정상적인 프로세스 실행에서 벗어난 API(Anomaly Process Instance)를 판별하는 방법론을 제안하고, API 판별을 위한 거리 측정치로서 Arc matrix(AM)를 제안한다.

2. Related research

2.1 Process Mining

프로세스 마이닝은 인공 지능, 데이터 마이닝, 비즈니스 프로세스, 패턴 인식 등의 연구분야와 연관이 있으며, 기본적인 개념은 정보시스템의 이벤트 로그에서 프로세스 모델을 포함한 패턴과 지식을 추출하는 것이다. 비즈니스 영역에서 유용한 지식이라 할 수 있는 업무 프로세스는 작업자의 머릿속에 있지만, 프로세스 마이닝 분야에서는 데이터로부터 실제 이벤트를 기반으로 업무 프로세스를 도출한다. 도출된 결과로부터 기업이 그럴 것이라고 추측하는 모델을 검증할 수 있으며, 이를 통해서 추가적인 분석을 수행한다(van der Aalst, 2011; Yang and Song, 2015).

프로세스 마이닝은 프로세스를 도출하기 위한 최소 단위의 데이터로 이벤트(Event)를 고려하고 있다. 이벤트는 단일한 사건의 발생을 저장한 데이터이므로 프로세스 모델을 도출하기 위해서는 이벤트가 하나의 인스턴스(Instance) 단위로 순차적으로 묶이게 된다. 시간 순차적으로 이벤트를 묶기 위하여 시간 데이터인 Timestamp를 포함한다.

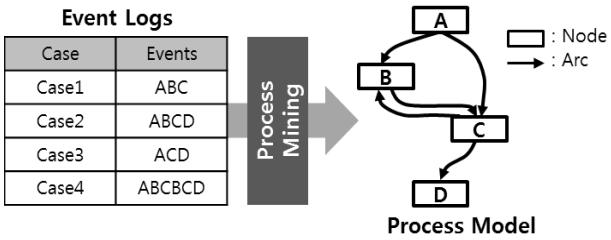

Figure 1. Progress of process mining

프로세스 마이닝을 위해서는 데이터에 포함되어야 하는 기본 속성이 있다. 첫째, 프로세스 모델을 도출하기 위하여 이벤트의 흐름을 추적하여야 한다. 이를 위해서 이벤트 군집의 구분자로서 Case ID가 필요하다. 둘째, 이벤트 군집에서 프로세스 단계별 이벤트를 구분하기 위한 구분자로서 Activity가 필요하다. Activity는 이벤트명이 될 수도 있고, 여러 개의 이벤트를 묶은 그룹의 이름이 될 수도 있다. 따라서 이벤트를 어느 범주에서 고려하는가에 따라 해당 Activity를 더욱 자세한 구분으로 쪼개거나 추상화 할 수도 있다. 이러한 Activity 수준에 따라서 프로세스 모델(Process Model)이 표현하는 정보가 달라진다. 셋째, 프로세스 모델은 순차적인 흐름을 표시하는 것으로서 이벤트의 발생 시간을 고려하기 위하여 Timestamp가 필요하다.

데이터에 포함되어 있는 기본 속성을 활용하여 표현된 프로

세스 모델은 Node와 Arc로 이루어져 있으며, Node 배치의 순서는 절대적인 시간이 아닌 상대적 시간으로서 발생한 순서에 따라 표시한다. 하나 Node에서 다음 Node로 이벤트 발생 순서로 표시 되고, 반복적인 작업이 이루어지는 이벤트의 경우에는 반대방향의 Arc가 표시된다. 프로세스 모델은 결과적으로 해당 이벤트 로그를 남긴 정보시스템에서 각각의 유저들의 작업을 순차적으로 표시한 것이다. 그리고, 분석자의 선택에 의해 데이터 전처리 혹은 클러스터링 기법 등을 사용하면, 일반적인 작업/사용 프로세스 모델과는 상이한 프로세스 모델을 도출할 수 있다.

2.2 Anomaly Detection

기존 연구에서 어떤 관측 개체가 다른 방법으로 발생되어, 나머지 관측 개체로부터 벗어나 있어서 비정상적이거나 특이하다고 판단되는 개체를 이상치(Anomaly)라고 한다(Hawkins, 1980). 이상치를 구분하기 위한 기준은 대부분 전문가의 영역에 속하는 것으로 시스템 혹은 서비스에서 해당 데이터가 의미하는 바를 정확히 이해하고 있어야 가능하다. 따라서 소수의 속성에 근거하여 데이터를 평가하는 것은 판단이 정확하지 않을 수 있으므로 다양한 분석 및 상이한 속성을 고려할 필요가 있다.

이상치는 일반적인 데이터의 집합에서 데이터를 발생시키는 주체의 의도나 특성에 부합하지 못하는 데이터이다. 즉, 데이터가 생성되는 메커니즘이 다르다고 추측되는 데이터를 찾는 것이다. 이상치의 정의는 활용하는 분야에 따라서 다를 수 있으며, 분석을 위한 데이터에서 제거되어야 하는 비정상 데이터로 분류되기도 한다(Han *et al*., 2012).

이상치 탐색(Anomaly Detection)이 최근 중요하게 적용되고 있는 분야로는 <Table 1>과 같이 네트워크 침입 탐지, 금융사기 탐지, 보건 및 의료 서비스 관련, 공공안전 및 보안 관련 등에서 활용되고 있다(Aggarwal, 2013).

빅데이터가 크게 이슈가 되고 있는 시점에서 정상을 찾는 것 이상으로 특이한 데이터, 잘못 입력된 데이터, 부당한 접근을 통해 기록된 데이터, 시스템의 오류 데이터 등을 도출해 내는 것이 중요하다. 중요성이 커진 만큼 이상치를 처리하는 방법이 빅데이터 분야에서 주목 받고 있다. 이상치를 일반적인 분석방법으로 데이터 전처리를 통한 필터링 기법을 활용하기도 하지만, 분석의 대상이 되는 이상치를 제외하는 위험이 따르기도 한다(Witten *et al*., 2011). 이러한 이유로, 이상치를 제외하기 보다는 이상치에서 새로운 의미 있는 정보를 얻기 위한 탐색 방법에 대한 연구가 시작되었다.

전통적인 통계적 기법은 이상치 탐색을 위해 데이터의 발생 확률에 대한 분포의 가정이 필요하고, 분포의 가정이 적절하더라도 모수를 추정하기 어렵기 때문에 빅데이터 분석에 제약이 있다. 분석을 위해 고려되는 데이터 집합에서 개체들 사이의 거리 측정값을 이용하여 서로 근거리에 있음을 평가할 수 있다. 이는 곧 이상치를 찾아내는 효과적인 방법으로서 활용되고 있다. 이러한 방법은 크게 두 가지로 나누어지는데, 거리 기반 이상치 탐색 기법과 밀도기반 이상치 탐색 기법이 있다. 거리 기반 이상치 탐색의 경우는 해당 기준 개체로부터 주어진 반경에 인접 개체가 충분히 존재하지 않으면 기준 개체를 이상치로 판단한다. 밀도 기반 이상치 탐색 기법의 경우는 기준 개체가 이웃 개체들과 이루는 밀도를 구하여, 기준 개체의 밀도가 이웃 개체들의 밀도와 비교하여 상당히 낮은 값을 가질 경우에 이상치로 판단한다(Han *et al*., 2012).

이상치를 찾고자 하는 연구가 프로세스 마이닝 분야에서도 이어지고 있다(Bezerra and Wainer, 2013). 기존 연구에서는 이상치를 판별하기 위해 인스턴스가 발생하는 횟수에 기준하여 도출된 프로세스 모델을 설명하는 정도를 비교하였다. 그러나 로그데이터에서 발생하는 유사 인스턴스 즉, 작업 목적이 같지만 이벤트의 반복적인 발생, 데이터 입력 오류 등을 포함한 인스턴스를 고려하지 못 했다. 본 논문에서는 작업환경의 특

Table 1. Needs of anomaly detection

Field	Description
Intrusion Detection Systems	In many host-based or networked computer systems, network traffic data may show unusual behavior because of malicious activity. The detection of such activity is referred to as intrusion detection(Agarwal and Mittal, 2012).
Credit Card Fraud	In many cases, unauthorized use may show different patterns, such as a buying spree from geographically obscure locations. Such patterns can be used to detect outliers in credit card transaction data(Ngai *et al*., 2011).
Interesting Sensor Events	The sudden changes in the underlying patterns may represent events of interest. Event detection is one of the primary motivating applications in the field of sensor networks(Du *et al*., 2006).
Medical Diagnosis	In many medical applications the data is collected from a variety of devices such as MRI scans, PET scans or ECG time-series. Unusual patterns in such data typically reflect disease conditions(Purarjomandlangrudi *et al*., 2014).
Law Enforcement	Determining fraud in financial transactions, trading activity, or insurance claims typically requires the determination of unusual patterns in the data generated by the actions of the criminal entity(Lin and Brown, 2006).
Earth Science	A significant data about weather patterns, climate changes, or land cover patterns is collected through a variety of mechanisms such as satellites or remote sensing. Anomalies in such data provide significant insights about hidden human or environmental trends, which may have caused such anomalies(Potter *et al*., 2003).

성을 고려한 이벤트 로그에서 이상치를 판단하는 방법을 제시하고자 한다.

3. Anomaly Process Instance(API)

프로세스 마이닝 연구에서 프로세스 모델을 통한 정상적인 인스턴스와 일반적 혹은 정상적이지 않은 인스턴스를 도출하여 데이터의 흐름을 평가할 수 있다.

기존 연구에서의 API를 Noise로서 제거해야 되는 대상으로 분류하였다(van der Aalst, 2011). 그러나 본 연구에서는 API를 시스템의 특수성 혹은 개성을 나타내거나, 이상치가 시스템으로부터 찾아야 되는 정보로서의 가치를 지니고 있다고 본다. 정상적인 범주의 프로세스 인스턴스(Process Instance)에서 도출할 수 있는 정보는 일반적으로 시스템 전문가에게 새로운 정보를 주기 힘들지만 예상치 못한 혹은 시스템을 디자인할 때 고려하지 않은 프로세스 인스턴스를 통해 개별 비즈니스 프로세스의 특징을 정의하거나, 보안상의 문제점을 찾거나, 생산성의 개선점을 찾을 수 있다.

프로세스 마이닝 영역에서 API를 판정하는 데에는 개별 인스턴스를 구성하는 이벤트의 발생 패턴이 일반적인 패턴과 얼마나 차이가 나는지를 측정하여야 한다. 이러한 패턴은 이벤트의 발생순서와 빈도에 의해서 결정되며, 본 논문에서는 이벤트의 발생 선후관계를 표현하는 Arc의 발생 빈도를 활용한 판별방법을 제안한다.

이러한 이상치 검출을 통해서 이벤트 로그(Event logs)를 생성한 시스템을 이해하고 분석/개선하는 활동을 수행할 수 있다. 이상치 검출 이후의 분석/개선 활동에서는 전문가의 판단이 추가적으로 필요할 수도 있으며, 이는 알고리즘적으로는 비정상적인 프로세스 인스턴스를 찾더라도 실제 비즈니스 환경에서는 고려하지 않았던 속성들로 인하여 정상적인 프로세스 인스턴스로 간주될 수 있기 때문이다.

API 탐색의 추가적인 활용 방향으로, 프로세스를 실행함에 있어 시스템에서 제공 또는 추천하는 프로세스 과정이 존재할 경우에, 제공되는 프로세스 경로를 이탈하는 경우만 보안 검사를 실시한다면 전체 프로세스의 보안검사 자원 낭비를 줄일 수 있다. 이러한 API 탐색은 일반적인 시스템에 적용할 수 있으며, 단순히 데이터의 속성을 고려하여 이상치를 탐색하는 것보다 흐름관점에서 접근한 API를 탐색하는 것이 자원관점에서 효율적이다.

이와 같은 이상치 검출 활동을 프로세스 마이닝을 기반으로 진행함으로써, 이벤트 발생의 패턴에 기반해서 비정상적인 실행을 찾아낼 수도 있으며, 프로세스 마이닝의 결과로 도출된 프로세스 모델은 도메인 전문가가 추가적인 분석/개선 활동을 하는데 있어서 직관적인 이해를 하는데 도움을 준다(van der Aalst *et al*., 2012).

본 논문에서 이상치를 판정하고 검출하는 것은 프로세스의 실행 케이스들 간의 거리를 계산하고 이를 척도(Measurement)로 진행하게 된다. 이는 시스템 혹은 비즈니스 모델의 잘못된 업무 흐름, 작업 구조를 의미하는 것과는 다른 의미를 지닌다. 실제로 발생한 데이터를 활용하여 판단하기 때문에, 이미 결과가 문제시 되지 않음에도 불구하고 일부 작업 주체들의 업무 흐름을 이상치로 판단하는 것은 시스템 전문가의 관점에서 문제의 소지가 있다. 인공지능 방법론에서 제시하는 것과 전문가의 판단 사이에 차이가 존재하기 때문에 본 논문에서는 제시하는 방법이 많은 실제 데이터로부터 평가될 필요성이 있다. 따라서 실제 항만 운영 시스템의 데이터를 활용하여 사례연구를 진행하였다.

4. Arc Matrix(AM)

정보시스템에서 발생하는 사건을 시간대별로 로그 형태로 저장한 원본 데이터를 이벤트 로그라고 한다. 따라서 이벤트 로그는 이벤트 정보를 가진 이벤트 집합이다. 분석 대상으로서의 이벤트의 집합에서 개별 이벤트는 여러 속성을 가지고 있으며, 프로세스 마이닝을 적용하기 위해 Case ID, Activity, Timestamp를 개별 속성에 매칭하여 분석한다. 이러한 속성을 활용한 프로세스 모델 도출, 클러스터링, 유사성 분석 등이 활발히 연구되었다. 기존의 유사성을 분석하기 위해 매트릭스(Matrix)를 활용한 연구는 빅데이터 분석 비용을 감소시키며, 그 활용도가 높다(Bae *et al*., 2006). 특히 프로세스를 정량적으로 표현할 수 있어서 인스턴스를 매트릭스로 변환하여 해당 인스턴스 사이의 거리를 측정하는데 장점이 있다.

Definition 1 :

Arc Matrix :

하나의 인스턴스로 하나의 AM을 생성함. 하나의 셀은 선행 이벤트와 후행 이벤트 정보를 가지는 Arc를 나타냄

$\mathcal{L}$: Event Logs(Original data)

ε : Set of events

e : Event, $e \in \varepsilon$

a_i : Activity

c_l : Case ID

M_l : Arc Matrix of c_l

I_l : Process Instance of c_l

AM에서는 하나의 인스턴스 내에서 존재하는 이벤트 사이의 Arc의 빈도를 매트릭스로 나타내었다. 또한, 이상치 척도를 계산하기 위해서는 이벤트 로그에 존재하는 두 인스턴스 사이의 거리값을 사용하며, 거리 측정치로는 Arc가 발생하는 빈도를 고려하였다. Arc는 순차적으로 발생한 이벤트를 나타내는 것으로서 방향성을 가지고 있다. 따라서 기준 Activity와 목표 Activity를 각각 고려하여 매트릭스를 구성하였다. <Table 2>에

서 순차 데이터를 임의로 생성하여 알고리즘으로 탐지하는 방법을 나타내었다. 예제 데이터는 20개 Case ID의 130개 이벤트를 포함하였으며, 하나의 Case ID에 해당하는 트레이스는 < > 괄호로 표기하고, 하나의 이벤트는 알파벳 소문자로 표기하였다. 예제 데이터를 보면 Case ID가 'c_1'인 경우 포함한 이벤트의 Activity는 '*a*', '*b*', '*c*', '*d*' 로서 '*a*', '*b*', '*c*' 는 1번씩 발생하였으며, '*d*'는 2번 발생하였다. 각각의 이벤트는 시간순서를 가지고 있다(Agrawal and Srikant, 1995; Pei *et al*., 2001; Shyur *et al*., 2013).

Table 2. Example of event logs

Case ID	Trace
c_1	$<a, b, c, d, d>$
c_2	$<a, b, c, c, d>$
c_3	$<a, c, b, c, d, d, d>$
c_4	$<a, b, c, b, c, b, c, d, d>$
c_5	$<a, b, c, d, d>$
c_6	$<a, b, c, b, c, d>$
c_7	$<a, b, c, d, d, d>$
c_8	$<a, b, c, b, b, c, d, d>$
c_9	$<a, b, c, d, d>$
c_{10}	$<a, b, c, b, c, d>$
c_{11}	$<a, b, c, d, d>$
c_{12}	$<a, b, c, b, c, b, c, d, d>$
c_{13}	$<a, b, c, d, d>$
c_{14}	$<a, b, c, b, c, d>$
c_{15}	$<c, b, c, d, d, d>$
c_{16}	$<a, b, c, b, c, b, c, d, d>$
c_{17}	$<a, b, c, d, d>$
c_{18}	$<a, b, b, c, d>$
c_{19}	$<a, c, b, c, d, d, d>$
c_{20}	$<a, b, c, b, c, b, c, d>$

M_1		Target Activity			
		a	b	c	d
Base Activity	a		1		
	b			1	
	c				1
	d				1

(a)

M_2		Target Activity			
		a	b	c	d
Base Activity	a		1		
	b			2	
	c		1		1
	d				

(b)

Figure 2. Arc matrix of c_1(a) and c_2(b)

<Table 2>에서 Case ID를 가진 각각의 인스턴스에 해당하는 AM을 <Figure 2>과 같이 변환한다. <Figure 2>(a)는 c_1의 AM으로서 'a' Activity에서 'b' Activity의 순서로 발생하는 경우의 Arc를 'ab'로 표기하였고, 'ab', 'bc', 'cd', 'dd' Arc가 각각 1번씩 발생하였다. <Figure 2>(b)는 c_2의 AM으로서 'ab', 'cb', 'cd' Arc가 각각 1번식 발생하였고, 'bc' Arc가 2번 발생하였다. 두 인스턴스의 Arc가 발생한 빈도 차이를 식 (1)을 사용하여 거리 기반의 AM을 구한다.

$$\text{Distance} : d_{lr} = \sum_{l}\sum_{r}|M_l - M_r| \quad (1)$$

$\lvert M_1 - M_2 \rvert$		Target Activity			
		a	b	c	d
Base Activity	a		0		
	b			1	
	c		1		0
	d				1

Figure 3. Arc matrix of distance between process instances

<Figure 3>에서 $|M_l - M_r|$는 'aa', 'cd' Arc가 0이고, 'bc', 'cb', 'dd' Arc가 1이므로 거리(Distance)는 3이 된다.

5. Detection of API

이벤트 로그가 실제 업무를 적절하게 포함하는 경우에는 구조적으로 Arc가 발생하는 여부와 발생빈도로 이상치를 탐지하지만 데이터가 상세한 업무를 포함하지 못하고, 응급상황 또는 적절치 않은 업무 프로세스를 포함한 경우는 시스템에 관한 전문가의 도움으로 실제 이상치(Real-Anomaly)를 탐지할 수 있다. 여기서 실제 이상치 는 실제 업무영역에서 판단하는 정상적이지 않은 인스턴스를 지칭하는 것으로서 본 논문에서 제시하는 알고리즘에 의해 탐지되는 이상치와는 구분된다. 로그의 적절성 관점에서 이벤트 데이터가 업무의 특성과 흐름을 표현하기에 적절하다면 정상적인 이벤트의 발생횟수가 커진다. 그러나 로그가 업무의 특성과 흐름을 적절하게 표현하지 못하는 경우에는 실제 이상치가 자주 발생하게 된다. 본 논문에서는 알고리즘을 통해 판단하는 이상치를 제시함으로 인해 실제 업무영역에서의 판단과 다를 수 있으며, 전문가의 판단에 의해 탐지하는 영역을 다루지 않는다.

인스턴스사이의 거리를 매트릭스에 표기하면 <Table 3>와 같다. 매트릭스는 거리 기준인 '기준 인스턴스(Base Instance)'와 거리를 측정하기 위해 변동된 인스턴스를 '대상 인스턴스(Target Instance)'로 표기한다. 매트릭스의 대각선은 0값을 가지고, 좌하단과 우상단의 값은 대각선을 기준으로 동일하다.

API를 탐지하기 위해서는 하나의 기준 인스턴스에 대응하는 대상 인스턴스의 거리 값을 r값과 비교한다. 이후에 전체

Table 3. Distance matrix of 20 instances

		Target Instance																			
		c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
Base Instance	c_1	0	3	4	4	0	3	3	3	0	3	4	4	0	3	3	4	0	2	4	5
	c_2	3	0	5	3	3	0	4	2	3	0	5	3	3	0	4	3	3	3	5	2
	c_3	4	5	0	6	4	5	1	5	4	5	0	6	4	5	1	6	4	6	0	7
	c_4	4	3	6	0	4	3	5	3	4	3	6	0	4	3	5	0	4	6	6	1
	c_5	0	3	4	4	0	3	3	3	0	3	4	4	0	3	3	4	0	2	4	5
	c_6	3	0	5	3	3	0	4	2	3	0	5	3	3	0	4	3	3	3	5	2
	c_7	3	4	1	5	3	4	0	4	3	4	1	5	3	4	0	5	3	5	1	6
	c_8	3	2	5	3	3	2	4	0	3	2	5	3	3	2	4	3	3	3	5	4
	c_9	0	3	4	4	0	3	3	3	0	3	4	4	0	3	3	4	0	2	4	5
	c_{10}	3	0	5	3	3	0	4	2	3	0	5	3	3	0	4	3	3	3	5	2
	c_{11}	4	5	0	6	4	5	1	5	4	5	0	6	4	5	1	6	4	6	0	7
	c_{12}	4	3	6	0	4	3	5	3	4	3	6	0	4	3	5	0	4	6	6	1
	c_{13}	0	3	4	4	0	3	3	3	0	3	4	4	0	3	3	4	0	2	4	5
	c_{14}	3	0	5	3	3	0	4	2	3	0	5	3	3	0	4	3	3	3	5	2
	c_{15}	3	4	1	5	3	4	0	4	3	4	1	5	3	4	0	5	3	5	1	6
	c_{16}	4	3	6	0	4	3	5	3	4	3	6	0	4	3	5	0	4	6	6	1
	c_{17}	0	3	4	4	0	3	3	3	0	3	4	4	0	3	3	4	0	2	4	5
	c_{18}	2	3	6	6	2	3	5	3	2	3	6	6	2	3	5	6	2	0	6	5
	c_{19}	4	5	0	6	4	5	1	5	4	5	0	6	4	5	1	6	4	6	0	7
	c_{20}	5	2	7	1	5	2	6	4	5	2	7	1	5	2	6	1	5	5	7	0

인스턴스 수에서 차지하는 비율이 π값보다 작은 경우를 이상치로 판단한다.

Definition 2 :

$\boldsymbol{C}$: Set of cases

$\boldsymbol{C} = \{c_1, \cdots, c_T\}$

T : Total number of cases

r : Distance threshold $(r > 0)$

π : Fraction threshold $(0 < \pi \leq 1)$

API 탐지법을 다음과 같이 의사코드로 작성하였다.

Algorithm : Distance-based API Detection(DAPID)

- Input data :

  A set of Cases $\boldsymbol{C} = \{c_1, \cdots, c_T\}$

  Threshold $r(r > 0)$ and $\pi(0 < \pi \leq 1)$

- Output : Distance-based (r, π) anomaly in $\boldsymbol{C}$

- Method :

  1 : **for** int i = 0; I < T; i++ (Base instance)

  2 : **for** int j = 0; j < T; j++ (Target instance)

  3 : **if** $d_{ij} < r$

  4 : count++;

  5 : **if** count $< (\pi \times T)$

  6 : the case is anomaly;

  7 : **else if** count $\geq (\pi \times T)$

  8 : the case is not anomaly;

$$\frac{|\{c_i \mid d_{ij} \leq r\}|}{T} < \pi \tag{2}$$

DAPID 알고리즘을 사용하여 20개의 인스턴스 예제에서 API를 탐지하였다. 탐지한 결과는 <Table 4>에서와 같이 r과 π 값의 변동에 따른 API를 도출하였다. 값 '0'은 정상적인 인스턴스를 나타내고, 값 '1'은 API임을 나타낸다.

r과 π값의 상호 관계는 <Figure 4>와 같이 우상향하는 등고선으로 나타낼 수 있다. <Figure 4>와 같이 r과 π가 변동할 때, 오른쪽 하단이 API에 가깝다. r = 0이고, $\pi > 0.3$일 때, 모든 인스턴스에서 API로 판단되었다. $\pi = 1.0$이고, r < 3일 때, 모든 인스턴스에서 API로 판단되었다. 인스턴스 중 20%를 API로 판별하고자 하면, r과 π의 조합은 '0.2'선을 따라서 나타난다.

예를 들어 <Figure 4>의 'A'와 같이 r = 5와 $\pi = 0.7$의 조합에서 Case ID가 'c_{18}'인 인스턴스가 유일하게 API로 판단되었다. 해당 인스턴스를 <Figure 5>에서 이벤트 로그로부터 도출된 프로세스 모델과 API를 비교하였다. 프로세스 모델 도출은 오픈 소스 프로그램인 ProM을 활용하였으며, 전체 이벤트 로그로부터 도출된 Arc와 비교할 때, Arc 2개가 차이 있었다.

Table 4. Result of DAPID

r	π	Caseid																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
										≀											
5	0.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0.8	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
5	0.9	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1
5	1	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	1	0	1	1	1
6	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
										≀											

1 : Anomaly, 0 : Non-anomaly.

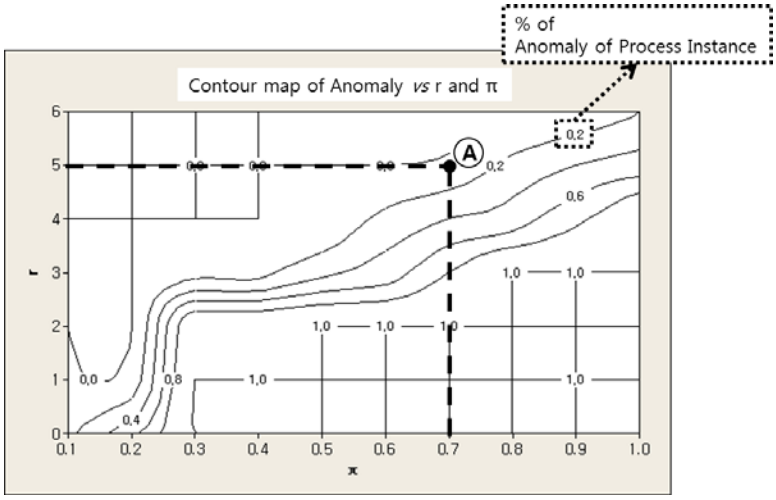

Figure 4. Contour map of API

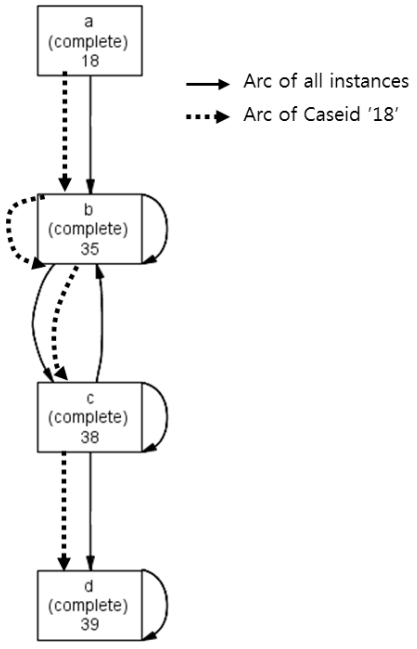

Figure 5. Process Model and API

제안하는 DAPID는 두 개의 한계치를 활용한 API 판별법으로서 r과 π의 조합은 판별에 직접적인 기준이 된다. 그러나 이러한 거리 기반의 한계치는 판별하고자 하는 인스턴스에 따라 다르게 적용될 수 있다. 개별 시스템, 적용 분야 등의 특성, 사용자 환경의 제약조건이 한계치의 조합을 다르게 만들 수 있기 때문이다. DAPID에서 거리기준(r)을 작게 두어서 근거리에 있는 인스턴스만을 고려하여 π보다 작은 비율의 인스턴스가 근거리에 있다면 이상치로 판별하는 것이 용이한 환경과, 거리기준을 크게 두어서 원거리를 기준으로 탐색하는 것이 용이한 환경도 있다. 본 논문에서는 사례 연구를 통해 실제 항만 터미널의 운영 시스템에서 발생하는 이벤트 로그로부터 API를 판별해 내는 것으로 DAPID의 활용성을 검증하고자 한다.

6. Case Study

실제 항만 터미널의 이벤트 로그로부터 DAPID를 사용하여 API를 탐색하였다. 데이터는 국내 P사에서 제공한 환적과 관련된 컨테이너 243개에 대한 998개의 이벤트이다.

<Table 5>은 243개 인스턴스 각각에 대한 거리 매트릭스를 생성한 결과이며, DAPID를 적용하여 <Table 6>에 API 여부를 판별하였다. 판별한 결과로부터 이상치 합(Anomaly Sum)의 크기 순으로 <Table 7>에 나타내었다. 결과적으로, 항만 터미널에서 API로는 이상치 합이 가장 큰 인스턴스 'Container 197'와 'Container 198'이 탐지되었다.

데이터로부터 탐지된 대표 프로세스와 발견된 API를 비교하면, <Figure 6>에서 실선은 데이터 전체에 대한 대표 프로세스를 나타내고, 점선은 API로 탐지된 인스턴스 'Container 197'와 'Container 198'를 나타내었다. 탐지된 API는 'DischargeYard'와 'RemarshallingPickup'을 거치지 않고, 'RemarshallingStack'을 반복적으로 작업한 경우이다. 특히, 'DischargeYard'를 거치지 않고 곧바로 'RemarshallingStack'을 작업한 것은 컨테이너를 하역/운반하는 장비가 컨테이너 없이 적재작업을 곧바로 하였다는 정보를 나타내고 있다. 따라서, DAPID는 순차적인 데이터에서 일반적인 프로세스를 따르지 않는 인스턴스를 찾아낼 수 있다.

'이상치 합', 'Event Count'는 모두 양적 데이터로서 그 영향에 대한 회귀분석을 시행하였고, 분석 결과는 <Table 8>에서와 같이 표준화 잔차의 절대값이 2보다 큰 9개의 인스턴스가 이상치로 판단되었다. 예를 들어 <Figure 7>에서 표준화 잔차값이 큰 두 개의 인스턴스인 'Container 015', 'Container 167'을 평면에 표기하였다. 인스턴스 'Container 015'의 경우는 4개의 이벤트를 포함하고 있지만 특이하게 이상치 합이 크게 나타났고, 인스턴스 'Container 167'의 경우는 5개의 이벤트를 포함하고 있지만 특이하게 이상치 합이 크게 나타났다. 이러한 결과는 DAPID의 활용이 API 판별에 유의미함을 보이고 있다.

<Table 8>에서 표준화 잔차로 탐지한 결과가 제공한 이상치는 본 논문에서 제시한 방법과 일부 유사하게 나타난다. 그러나 회귀분석에서는 변수의 선형성, 잔차의 분포 등에 영향을

Table 5. Distance matrix for 243 instances

	Container 007	Container 008	Container 009	Container 010	Container 011	Container 012	Container 013	Container 014	Container 015	Container 016	…
Container 007	0	0	0	0	0	0	4	4	4	0	
Container 008	0	0	0	0	0	0	4	4	4	0	
Container 009	0	0	0	0	0	0	4	4	4	0	
Container 010	0	0	0	0	0	0	4	4	4	0	
Container 011	0	0	0	0	0	0	4	4	4	0	
Container 012	0	0	0	0	0	0	4	4	4	0	
Container 013	4	4	4	4	4	4	0	0	2	4	
Container 014	4	4	4	4	4	4	0	0	2	4	
Container 015	4	4	4	4	4	4	2	2	0	4	
Container 016	0	0	0	0	0	0	4	4	4	0	
⋮											

Table 6. Result of DAPID for 243 instances

	r	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2	2	…
	π	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	0.1	0.2	0.3	
Case ID	Container 012	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	
	Container 013	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 014	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 015	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 016	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	
	Container 017	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 018	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 019	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 020	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	Container 021	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	
	⋮																								

Table 7. The most likely anomaly instances

Case ID	Anomaly Sum	Event Count
Container 197	71	7
Container 198	71	7
Container 060	51	7
Container 167	51	5
Container 015	44	4
Container 013	43	6
Container 014	43	6
Container 017	43	6
Container 018	43	6
Container 019	43	6
Container 020	43	6
⋮		
Average	9.095	4.107
Standard Deviation	9.213	0.4689
Maximum	71	7
Minimum	7	4

Table 8. Detection anomaly using residual

Case ID	Event Count	Anomaly Sumed	Standardiz Residuals
Container 015	4	44	11.28
Container 038	5	17	-2.62
Container 060	7	51	-3.74
Container 144	5	17	-2.62
Container 167	5	51	7.86
Container 197	7	71	2.92
Container 198	7	71	2.92
Container 201	5	17	-2.62
Container 202	5	17	-2.62

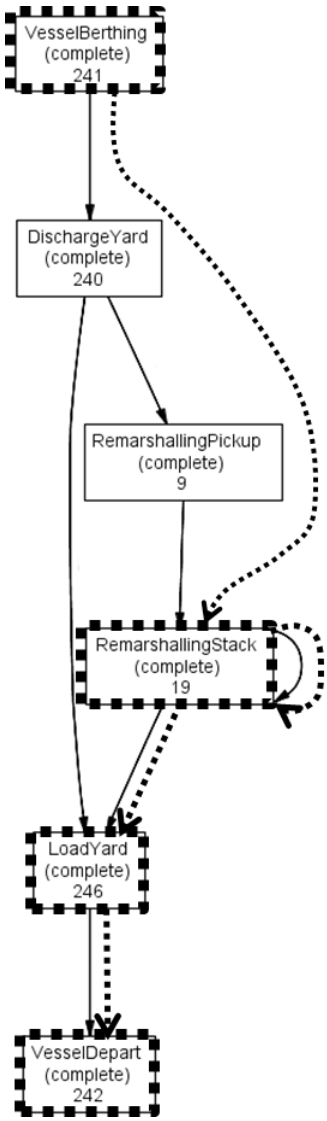

Figure 6. Process of example data and API

Caseid	Activity	Timestamp
Container 015	VesselBerthing	2013-08-06 15:55:00
Container 015	DischargeYard	2013-08-06 20:58:00
Container 015	RemarshallingPickup	2013-08-08 23:24:00
Container 015	RemarshallingStack	2013-08-08 23:31:00

Caseid	Activity	Timestamp
Container 167	RemarshallingPickup	2013-08-04 3:49:00
Container 167	RemarshallingStack	2013-08-04 4:07:00
Container 167	RemarshallingStack	2013-08-04 4:07:00
Container 167	LoadYard	2013-08-05 3:48:00
Container 167	VesselDepart	2013-08-05 10:00:00

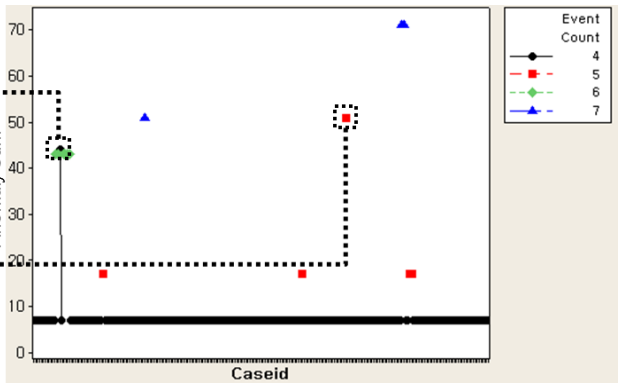

Figure 7. Residual of API

받으면서 복잡한 계산과정을 수행한다. 따라서 DAPID는 제안한 AM를 사용하여 API를 도출함으로써 전통적인 통계적 방법론에서 분석을 위해 수행하는 단계보다 간단하며, 직관적이다. 특히, 기존의 통계적 분석은 특정되지 않은 속성을 고려하여, 다양한 분석법을 적용하고, 분석자 직관력과 부합하는 결과에 따라 반복적인 분석을 수행한 후에야 이상치로서 개별

이벤트를 탐지하였다. 이러한 접근은 프로세스 마이닝을 위해 사용되는 인스턴스 단위의 이상치를 찾기 어렵다. 따라서 선후 이벤트의 순서를 가지고 있는 프로세스 인스턴스의 집합에서 API를 찾는 방법으로 DAPID의 의미가 있다.

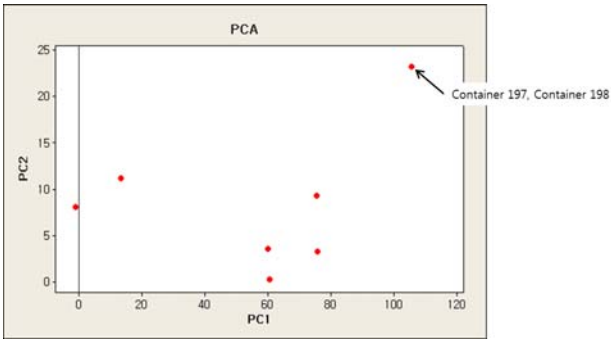

Figure 8. PCA result for representing process instances of containers

<Figure 8>에서 거리 매트릭스의 주성분 분석 결과를 통해 거리 기반의 API를 확인할 수 있다. 'Container 197'과 'Container 198'의 경우 같은 점으로 표시 되었다. 해당 컨테이너는 다른 컨테이너 즉, 다른 인스턴스와 확연히 떨어져 있음을 알 수 있었다.

DAPID를 통해 판별된 API는 항만 운영 시스템에서 제공하는 이벤트 로그에서 특이한 인스턴스이었다. 그 특징으로는 쌍으로 운영되어야 하는 이벤트를 쌍으로 수행하지 않는 경우, 환적물을 이송하는 할 때에 필수적으로 발생하는 이벤트를 누락하는 경우, 같은 작업을 반복적으로 수행하는 경우 등이었다.

사례 연구는 대표적인 항만 운영 프로세스만 고려하여 Activity 수가 많지 않았다. 그러나 항만에서 발생하는 이벤트는 운송물의 특징, 컨테이너의 특징, 운송 경로의 특징 등의 속성으로 인해 더욱 다양한 이벤트를 발생시키고 있다. 향후에 실제 작업 시간, Resource 등의 속성을 활용한다면 더욱 다양한 Activity 관점에서의 이상치 판별이 가능할 것으로 보이며, 이러한 다양한 관점이 결국에는 전문가의 판단에 가까워지는 방법일 것이다.

7. Conclusion

기존의 프로세스 마이닝 연구분야에서는 API를 프로세스 모델 도출, 모니터링, 개선 등의 과정에서 누락하거나 혹은 전처리 과정에서 제외된 프로세스 인스턴스로서 분류하였다. 그러나, 본 논문에서는 API 탐색을 이벤트 로그 분석의 시작점으로 보았다. API를 찾아내기 위한 방법으로서 Case ID, Activity, Timestamp 등의 속성을 활용하여 이벤트의 선후관계와 빈도를 고려하였다. 특히, 해당 선후관계와 빈도를 포함한 AM을 거리 척도(Distance measure)로 제안하였으며, AM을 활용하여 API를 판별하기 위해 DAPID를 제안하였다. 마지막으로 DAPID의 결과를 검증하기 위해 사례 연구를 통해 실제 터미널 데이터로부터 컨테이너의 이상 물류를 확인하였다.

이벤트 로그의 크기가 클수록 특정 목적에 부합하는 인스턴스를 찾아서 분류하고 분석해야 하는 필요성이 증가하고 있다. 이러한 필요성과 비례하여 이벤트 로그의 완전성을 저해하거나 불필요한 오류로 여겨지던 소수의 특이한 인스턴스를 탐지하는 방법으로서 DAPID의 가치가 있다 하겠다. 또한, 거리 척도인 AM을 활용하기 때문에 시간순차적인 데이터를 발생하는 시스템으로부터 특이하고, 일반적이지 않은 패턴을 찾아내고자 하는 경우에 유용할 것이다.

참고문헌

Agarwal, B. and Mittal, N. (2012), Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques, *Procedia Technology*, **6**, 996-1003.

Aggarwal, C. C. (2013), *Outlier analysis* : Springer Science and Business Media.

Agrawal, R. and Srikant, R. (1995), *Mining sequential patterns*, Paper presented at the Data Engineering, Proceedings of the Eleventh International Conference on.

Bae, J., Liu, L., Caverlee, J., and Rouse, W. B. (2006), *Process Mining, Discovery, and Integration using Distance Measures*, Paper presented at the Web Services, ICWS 2006. International Conference on.

Bezerra, F. and Wainer, J. (2013), Algorithms for anomaly detection of traces in logs of process aware information systems, *Information Systems*, **38**(1), 33-44.

Çiflikli, C. and Kahya-Özyirmidokuz, E. (2010), Implementing a data mining solution for enhancing carpet manufacturing productivity, *Knowledge-Based Systems*, **23**(8), 783-788.

Du, W., Fang, L. and Peng, N. (2006), LAD : Localization anomaly detection for wireless sensor networks, *Journal of Parallel and Distributed Computing*, **66**(7), 874-886.

Han, J., Kamber, M., and Pei, J. (2012), *Data mining : concepts and techniques* : Morgan Kaufmann.

Hawkins, D. M. (1980), *Identification of outliers* : Springer, **11**.

Laxhammar, R. (2014), Anomaly Detection, *Conformal Prediction for Reliable Machine Learning : Theory, Adaptations, and Applications, Elsevier Insight Series, Morgan-Kaufmann Publishers*.

Lin, S. and Brown, D. E. (2006), An outlier-based data association method for linking criminal incidents. *Decision Support Systems*, **41**(3), 604-615.

Ngai, E. W. T. *et al*. (2011), The application of data mining techniques in financial fraud detection : A classification framework and an academic review of literature, *Decision Support Systems*, **50**(3), 559-569.

Pei, J. *et al*. (2001), *Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth*, Paper presented at the International Conference on Knowledge Discovery in Databases and Data Mining.

Potter, C. *et al*. (2003), Major disturbance events in terrestrial ecosystems detected using global satellite data sets, *Global Change Biology*, **9**(7), 1005-1021.

Purarjomandlangrudi, A., Ghapanchi, A. H., and Esmalifalak, M. (2014),

A data mining approach for fault diagnosis : An application of anomaly detection algorithm, *Measurement*, **55**, 343-352.

Rebuge, Á. and Ferreira, D. R. (2012), Business process analysis in healthcare environments : A methodology based on process mining. *Information Systems*, **37**(2), 99-116.

Shyur, H.-J., Jou, C., and Chang, K. (2013), A data mining approach to discovering reliable sequential patterns, *Journal of Systems and Software*, **86**(8), 2196-2203.

Sim, S. *et al*. (2012), Healthcare process pattern analysis with triage in the emergency department, *Journal of the Korean Operations Research and Management Science Society*, **37**(4), 111-124.

van der Aalst, W. M. P. (2011), *Discovery, Conformance and Enhancement of Business Processes* : Springer.

van der Aalst, W. M. P. *et al*. (2012), *Process mining manifesto*, Paper presented at the Business Process Management Workshops.

van der Aalst, W. M. P. *et al*. (2007). Business process mining : An industrial application, *Information Systems*, **32**(5), 713-732.

Witten, I. H., Frank, E., and Hall, M. A. (2011), *Data Mining : Practical machine learning tools and techniques* (3rd ed.) : Morgan Kaufmann.

Yang, H. and Song, M. (2015), Analyzing Repair Processes Using Process Mining : A Case Study, *Journal of the Korean Institute of Industrial Engineers*, **41**(1), 86-96.