J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr
Copyright ⓒ 2021 KSII

베이지안 확률 및 폐쇄 순차패턴 마이닝 방식을 이용한 설명가능한 로그 이상탐지 시스템☆

An Interpretable Log Anomaly System Using Bayesian Probability and Closed Sequence Pattern Mining

윤 지 영[1] 신 건 윤[2] 김 동 욱[2] 김 상 수[3] 한 명 묵[1*]
Jiyoung Yun Gun-Yoon Shin Dong-Wook Kim Sang-Soo Kim Myung-Mook Han

요 약

인터넷과 개인용 컴퓨터가 발달하면서 다양하고 복잡한 공격들이 등장하기 시작했다. 공격들이 복잡해짐에 따라 기존에 사용하던 시그니처 기반의 탐지 방식으로 탐지가 어려워졌으며 이를 해결하기 위해 행위기반의 탐지를 위한 로그 이상탐지에 대한 연구가 주목 받기 시작했다. 최근 로그 이상탐지에 대한 연구는 딥러닝을 활용해 순서를 학습하는 방식으로 이루어지고 있으며 좋은 성능을 보여준다. 하지만 좋은 성능에도 불구하고 판단에 대한 근거를 제공하지 못한다는 한계점을 지닌다. 판단에 대한 근거 및 설명을 제공하지 못할 경우, 데이터가 오염되거나 모델 자체에 결함이 발생해도 이를 발견하기 어렵다는 문제점을 지닌다. 결론적으로 사용자의 신뢰성을 잃게 된다. 이를 해결하기 위해 본 연구에서는 설명가능한 로그 이상탐지 시스템을 제안한다. 본 연구는 가장 먼저 로그 파싱을 진행해 로그 전처리를 수행한다. 이후 전처리된 로그들을 이용해 베이지안 확률 기반 순차 규칙추출을 진행한다. 결과적으로 "***If** 조건 **then** 결과, 사후확률(θ)*" 형식의 규칙집합을 추출하며 이와 매칭될 경우 정상, 매칭되지 않을 경우, 이상행위로 판단하게 된다. 실험으로는 HDFS 로그 데이터셋을 활용했으며, 그 결과 F1score 92.7%의 성능을 나타내었다.

☞ 주제어 : 설명가능한 인공지능, 로그 이상탐지 시스템, 베이지안 확률, 규칙 추출

ABSTRACT

With the development of the Internet and personal computers, various and complex attacks begin to emerge. As the attacks become more complex, signature-based detection become difficult. It leads to the research on behavior-based log anomaly detection. Recent work utilizes deep learning to learn the order and it shows good performance. Despite its good performance, it does not provide any explanation for prediction. The lack of explanation can occur difficulty of finding contamination of data or the vulnerability of the model itself. As a result, the users lose their reliability of the model. To address this problem, this work proposes an explainable log anomaly detection system. In this study, log parsing is the first to proceed. Afterward, sequential rules are extracted by Bayesian posterior probability. As a result, the "If condition then results, post-probability" type rule set is extracted. If the sample is matched to the ruleset, it is normal, otherwise, it is an anomaly. We utilize HDFS datasets for the experiment, resulting in F1score 92.7% in test dataset.

☞ keyword : Explainable AI, Log anomaly detection, Bayesian probability, Rule extraction

1. 서 론

개인용 컴퓨터와 인터넷 보급이 활발해지면서 다양하고 복잡한 공격들이 등장하기 시작했다. 공격들이 발전됨에 따라 기존의 시그니처 기반의 이상탐지 시스템으로는 공격들을 탐지하기 어려워졌다. 전통적인 이상탐지 시스템의 경우 학습을 진행하기 전에 미리 다양한 분석들을 수행해야 하며, 시그니처 요소들(예: IP주소, 도메인명 등)이 쉽게 변할 수 있기 때문이다[1]. 이런 문제들을 해결하기 위해 로그를 이용한 이상탐지 시스템이 등장하기 시

1 Department of Software, Gachon University, Sungnam-si, 13120, Korea
2 Department of Computer Engineering, Gachon University, Sungnam-si, 13120, Korea
3 Agency for Defense Development Songpa P.O Box 132, Seoul, 05661 Korea
* Corresponding author (mmhan@gachon.ac.kr)
[Received 26 February 2021, Reviewed 17 March 2021, Accepted 26 March 2021]
☆ 본 연구는 국방과학연구소 연구용역 지원사업의 연구결과로 수행되었음 (UD200020ED)
☆ 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2018R1D1A1B07050864).

http://dx.doi.org/10.7472/jksii.2021.22.2.77

작했다. 로그란 컴퓨터의 중요한 행동, 이벤트 및 상태를 기록하는 도구로서, 이를 학습할 경우, 다양한 상황에서 변화되는 행동에 대해 탐지가 가능하다.

최근에는 딥러닝을 이용해 로그의 순서를 학습하는 탐지 시스템들이 등장하기 시작했다. 특히 로그의 순서를 학습하기 때문에 자연어처리에서 자주 사용되는 딥러닝 방식인 순환신경망이 활발히 이용된다. 해당 로그 이상탐지 시스템들은 약 97% 이상의 정확도를 웃돌며 좋은 성능을 보여준다[1-4]. 하지만 앞선 모델들은 모두 딥러닝 기반으로 판단에 대한 근거 및 설명을 제공하지 못한다는 문제점을 지닌다. 판단에 대한 근거를 제공하지 못할 경우, 데이터 오염 및 모델 자체의 문제를 파악할 수 없으며, 모델 자체에서 새로운 인사이트를 학습해도 이를 사용자에게 전달할 수 없어 결과적으로 모델에 대한 신뢰성이 떨어진다는 문제를 일으키게 된다[5]. 이는 아무리 훌륭한 성능을 가지고 있다 하더라도 실제 필드에서 사용할 수 없음을 시사한다[6].

본 연구에서는 해당 신뢰성 문제를 해결하기 위해 베이지안 확률 기반의 폐쇄 순차패턴 마이닝을 이용한 정확하면서도 설명가능한 로그 이상탐지 시스템을 제안한다. 제안 모델은 결과적으로 "***If*** *조건* ***then*** *결과, 사후확률(θ)*" 형태의 규칙들을 추출한다. 이때 *조건부*에는 로그 템플릿의 순차패턴들이 들어가게 되고, *결과*로서는 해당 순차패턴들이 등장할 때 정상일 경우 등장하는 다음 로그 템플릿이 등장한다. *사후확률부*의 경우, 해당 조건이 만족할 때 결과가 될 확률을 나타낸다. 해당 패턴과 결과값이 매칭될 경우 정상, 매칭되지 않을 경우 이상행위로 판단해 이상탐지를 수행하게 된다.

제안 모델의 학습 흐름은 Deeplog[1]에서 사용된 방식을 따른다. 각각의 로그 시퀀스들을 특정 단위로 분리하고 분리된 시퀀스의 다음 로그 예측을 수행하여 성공할 경우 정상, 성공하지 못할 경우 이상행위로 분류하는 방법이다. Deeplog는 해당 예측에 딥러닝 모델을 사용했지만 본 연구에서는 규칙추출 모델을 활용한다. 연구는 크게 로그 전처리 단계, 베이지안 확률 기반 폐쇄 순차규칙 추출 단계 총 두 단계로 이루어진다. 로그 전처리 단계에서는 로그 전처리 라이브러리인 Drain[7]을 이용해 전처리를 수행한 후 식별자에 따라 전처리된 로그들을 Transaction 형태로 변형한다. 이후 각각의 Transaction을 특정 개수의 로그 시퀀스로 분리한다. 순차규칙 단계에서는 이전 단계에서 얻어진 특정 개수의 로그 시퀀스 학습을 진행한다. 지지도 및 신뢰도를 통해 먼저 필터링을 수행한 뒤 베이지안 사후확률을 이용해서 의미있는 순차규칙들을 추출하게 된다. 제안 연구는 일반적인 머신러닝 기법의 연구들에 비해 좋은 성능을 보이며 딥러닝을 이용한 연구보다 성능이 낮으나 매우 높은 설명성을 제공한다.

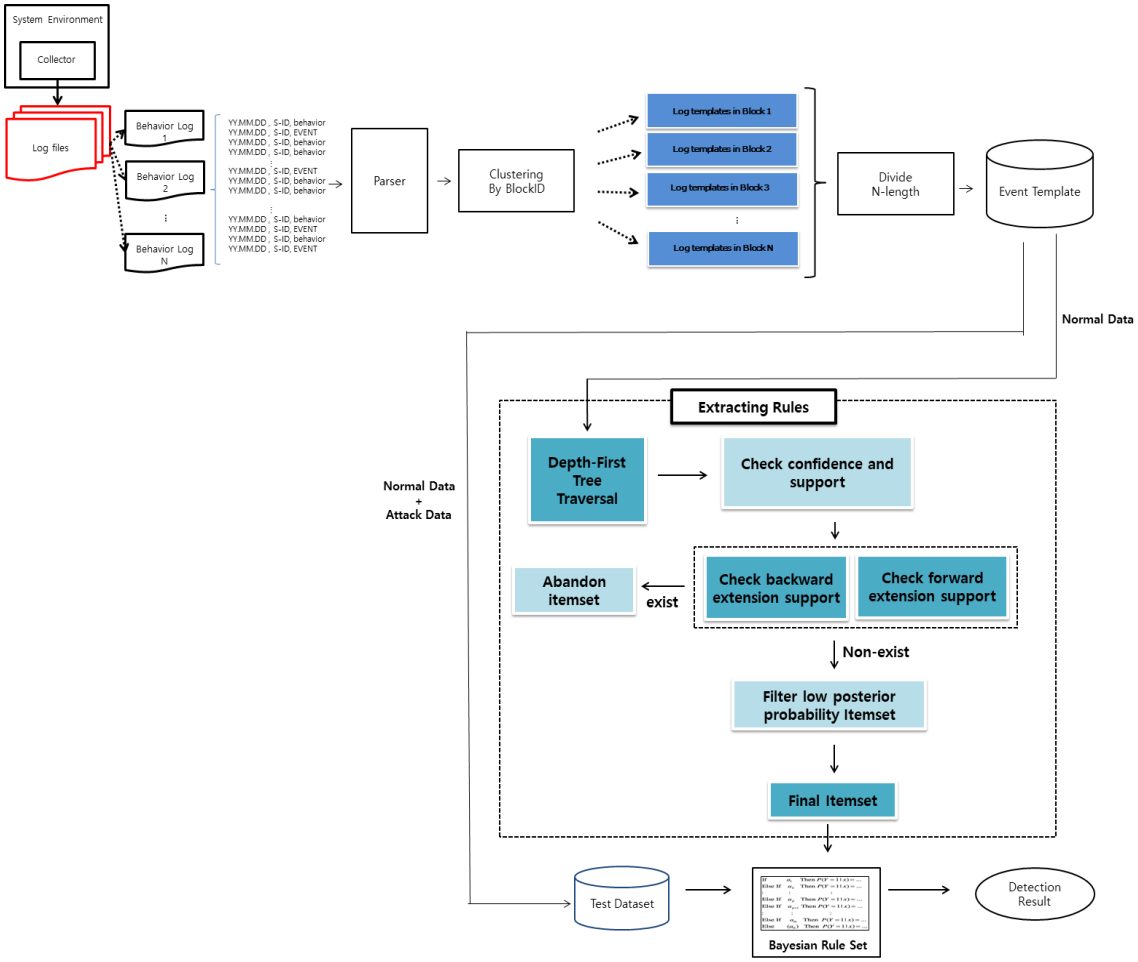
2절에서는 관련 연구에 대해 소개하며, 3절에서는 제안 모델, 그 중에서도 특히 제안한 규칙 추출 알고리즘에 대해 설명을 진행한다. 4절에서는 실험에 대해 언급하며 5절에서는 결론을 내며 마무리한다.

2. 관련 연구

2.1 로그 이상탐지 시스템

로그 이상탐지 시스템이란 정상로그들의 순서 및 패턴을 학습해 여러가지 위협을 탐지하는 이상탐지 시스템을 의미한다. 일반적으로 로그 전처리를 수행해 이벤트 템플릿을 추출하는 단계, 이벤트 템플릿을 이용해 학습을 진행하는 단계 두 단계로 이루어진다. 이벤트 템플릿이란 로그에서 파라미터를 제외한 로그 구조를 제공하는 부분으로 "PacketResponder <*> for block <*> terminating"과 같은 형태를 지닌다. 이벤트 템플릿을 추출하는 알고리즘으로는 대표적으로 SPELL[8], Drain[7] 등이 존재한다. 이벤트 템플릿을 이용한 학습 단계에서는 PCA (Principle Component Analysis), 군집화 및 딥러닝 등을 이용한 연구들이 존재하며 특히 최근에는 높은 정확도로 인해 딥러닝을 이용한 연구들이 큰 각광을 받고 있다.

PCA를 이용한 연구에서는 로그들을 파싱한 후 특정 로그템플릿들의 빈도수를 기준으로 탐지를 진행한다. 가장 먼저 빈도 패턴이 기존의 정상 패턴과 동일한지를 기준으로 필터링을 진행한 후 의심후보 패턴들에 대해 PCA 탐지를 진행하며 이를 통해 정상 케이스와 유사하지 않을 경우, 이상행위로 판단한다[9]. 군집화를 이용한 연구에서는 동적 군집화를 이용한 방식과 시퀀스 자체의 클러스터링 방식이 존재한다. 동적 군집화의 경우, 로그들을 전처리한 후, 윈도우에 맞게 분할한다. 이후 각각의 로그들이 다음 윈도우로 넘어가면서 어떻게 변화하는지 확인하고 변화성이 달라질 경우 이상행위로 판단한다[10]. 시퀀스 군집화 연구의 경우, 로그를 벡터화 한 후 시퀀스를 생성하고 이를 클러스터링해 군집에 따라 로그의 이상을 탐지한다[11]. 딥러닝을 이용한 로그 이상탐지는 자연어학습에서 많이 사용되는 순환신경망을 이용해 진행된다. 가장 먼저 로그들을 특정한 길이로 쪼개거나 패딩한 후 임베딩을 거쳐 학습을 진행한다. 딥러닝 학습



(그림 1) 제안모델 프레임워크
(Figure 1) Framework of the proposed model

의 경우, 상대적으로 간단한 전처리에도 불구하고 매우 높은 성능을 보여준다[1-4].

2.2 폐쇄 순차패턴 마이닝

폐쇄 순차패턴 마이닝이란 순차패턴 중 폐쇄된 패턴을 찾는 알고리즘을 의미한다. 특정 패턴 A에 대해 상위 집합 패턴인 B가 존재하지 않거나, 존재하더라도 지지도 값이 A에 비해 작을 경우, A를 폐쇄 패턴이라고 부른다[12]. 수식 1은 폐쇄 패턴의 조건을 보여준다.

$$\begin{aligned} &1.\ \exists A \subset B,\ \neg(\exists B) \\ &2.\ \exists A \subset B,\ \exists B\ AND(supp(B) < supp(A)) \end{aligned} \quad (1)$$

폐쇄된 순차패턴을 통해 전체적인 패턴 정보는 유지하면서도 일반적인 패턴 마이닝보다 빠르게 중요한 패턴들을 추출할 수 있다. 즉, 정보의 손실없이 빠르고 중복이 적은 패턴들을 추출할 수 있게 된다. 대표적인 알고리즘으로는 CloSpan, BIDE가 있으며 이외에도 다양한 폐쇄 순차패턴 마이닝 방법들이 존재한다.

CloSpan은 널리 활용된 폐쇄 순차패턴 마이닝 방식으로 크게 아이템셋 생성 단계, 폐쇄 패턴 제거 단계로 이루어진다. 아이템셋 생성 단계에서는 지지도가 낮은 아이템셋들을 제거한 후 지지도 내림차순으로 이들을 정렬한다. 이후 하나씩 아이템들을 늘리면서 조건을 확인한다. 폐쇄 패턴 제거 단계에서는 폐쇄 패턴 조건을 확인

하고 해당 조건을 만족하지 않을 경우 제거한다[13]. BIDE 알고리즘은 CloSpan을 발전시킨 알고리즘으로 CloSpan의 비효율성을 개선한다. CloSpan의 경우, 모든 후보들을 유지한 채 마지막 단계에서 조건을 확인하고 조건에 맞지 않을 경우 제거한다. 하지만 BIDE 알고리즘은 메모리 사용 및 학습시간의 문제를 해결하기 위해 후보들을 미리 제거하는 방식으로 제안되었다. 예를 들어, 아이템셋들을 마이닝하면서 기존에 존재했던 아이템셋들의 상위집합이면서 지지도값이 동일한 경우가 있다면 기존에 존재했던 해당 아이템셋들을 바로 제거하는 식으로 진행된다. 이를 통해 전체적으로 메모리 크기를 줄이면서도 학습시간을 단축시킨다[14].

2.3 베이지안 규칙 리스트

베이지안 규칙 리스트란 베이지안 이론을 기반으로 규칙집합을 생성하는 알고리즘을 나타낸다. 베이지안 규칙 리스트는 가장 먼저 사용자가 지정한 지지도 값을 넘는 모든 아이템셋들을 추출하고 이들 중 높은 베이지안 사후확률을 가지는 아이템셋들을 선택해 규칙을 생성한다. 해당 연구의 결과물은 규칙 리스트로 순서가 존재하며 앞 순서에서 매칭될 경우, 뒷 순서에서는 매칭을 진행하지 않는다. 실험으로는 뇌졸중 데이터셋을 활용했으며 대표적인 설명모델인 CART5에 비해 좋은 결과를 얻었다[15].

3. 제안모델

제안 모델은 확률이 포함된 규칙집합을 생성해 설명성을 제공하면서도, 정확한 규칙을 추출해 좋은 성능을 유지하는 것을 목표로 한다. 규칙과 매칭될 경우 정상으로 매칭되지 않을 경우 이상행위로 판단하게 된다. 모델의 전체적인 프레임워크는 그림 1과 같이 총 두 단계로 이루어진다. 대부분의 로그이상탐지 연구들이 크게 두 단계로 나눠지는데, 이유는 로그를 바로 사용할 수 없기 때문에 일정한 형태를 이루는 템플릿으로 파싱 후 각 파싱된 로그 템플릿들을 이용해 학습을 진행하기 때문이다.

3.1 로그 이벤트 템플릿 추출

로그 이벤트 템플릿 추출 단계에서는 로그의 전처리를 진행한다. 전처리 단계는 크게 두 단계로 나눠진다. 첫 번째 단계에서는 로그를 각각의 로그템플릿에 맞게 분리하는 파싱을 진행한다. 해당 단계에서는 비지도 학습 방식의 로그 파싱 라이브러리 Drain을 사용한다. Drain을 이용해 파싱한 결과는 그림 2와 같다.

```
Received block blk_3587508140051953248 of
size 67108864 from /10.251.42.84
Verification succeeded for blk_-2827716238972737794
PacketResponder 1 for block blk_38865049064139660
terminating
```

↓

```
Received block <*> of size <*> from <*>
Verification succeeded for <*>
PacketResponder <*> for block <*> terminating
```

(그림 2) Drain 파싱 결과
(Figure 2) Result of parsing using Drain

파싱을 진행한 후 이들을 식별자 혹은 세션 단위로 묶어 로그 시퀀스를 생성한다. 해당 시퀀스는 사용자가 지정한 특정 길이에 따라 분리된다. 이후 다음 로그 템플릿을 저장해 규칙추출에서 사용할 결과 값으로 활용한다. 예를 들어 1,3,5,5,2,4,7,8,19,10 이라고 하는 로그 시퀀스가 존재하고 분류 길이가 7이라고 한다면 분리된 시퀀스로 (1,3,5,5,2,4,7)를, 예측 클래스로 8을 선택하게 된다. 즉, 7개의 시퀀스를 가지고 학습을 진행하고, 정상데이터에서는 다음 로그가 8이기 때문에 예측에 8이 있다면 정상, 없다면 이상행위로 판단하게 된다.

3.2 순차 규칙 추출 단계

순차규칙 추출 단계에서는 앞선 단계에서 전처리한 특정 길이 시퀀스와 예측 클래스를 이용해 의미있는 규칙들을 추출한다. 해당 단계는 크게 폐쇄 순차패턴 추출 단계, 베이지안 확률을 이용한 필터링 단계 두 단계로 나눠진다. 그림 3은 해당 단계를 통해 추출된 규칙의 예시를 나타낸다.

```
If ('<*>Got exception while serving <*> to <*>' and 'PacketResponder <*>
    <*> Exception <*>' and 'BLOCK* ask <*> to replicate <*> to <*>')
Then [[(' BLOCK* ask <*> to replicate <*> to <*>', 0.07229172446008761)]]
```

(그림 3) 추출된 규칙 예시
(Figure 3) Example of extracted rule

3.2.1 순차패턴 추출

가장 먼저 폐쇄 순차패턴 마이닝 알고리즘인 BIDE를

이용해 의미있는 패턴들을 추출한다. 기존 알고리즘은 선험적 특성에 의해서 지지도 값만을 기준으로 패턴을 추출한다. 하지만 이 경우, 단지 빈번하게 발생하는 패턴을 찾기 때문에 예측 클래스에 맞게 등장하는 로그 패턴을 찾기 어렵다. 이를 위해 본 연구에선 지지도보다 신뢰도에 초점을 맞춰 마이닝을 진행한다. 가장 먼저 트리를 생성하고 각각의 아이템셋(패턴)의 지지도와 신뢰도값 필터링을 진행한다. 이때 지지도 값은 낮추고 신뢰도 값은 상대적으로 높여 신뢰도에 초점을 맞춘다. 이후 폐쇄 패턴의 조건을 확인한다. 폐쇄 패턴의 조건은 동일하게 지지도값을 기준으로 진행된다. 이때 지지도 값만을 폐쇄 패턴의 조건에 추가한 이유는 다음과 같다. 신뢰도 값은 수식 2와 같은 형태를 지닌다.

$$conf = \frac{n(X \cap A)}{n(X)} \tag{2}$$

분모로 아이템셋(패턴) 자신의 빈도수를, 분자로 아이템셋(패턴)과 타겟 클래스의 동시 발생 빈도수를 갖는다. 이때 지지도 기준의 폐쇄 패턴의 조건을 만족한다는 의미는 두 개의 아이템셋(패턴)이 모두 동시에 발생한다는 의미로, 빈도수인 분모와 타겟 클래스가 포함된 빈도수인 분자가 동일할 수 밖에 없기 때문이다. 즉, 지지도값이 동일할 경우, 신뢰도가 동일하기 때문에 기본적으로 계산되는 지지도만을 활용한다.

3.2.2 베이지안 확률을 이용한 필터링 단계

해당 단계에선 앞서 얻어진 패턴들에서 베이지안 확률 필터링을 적용해 의미있는 규칙집합을 생성한다. 즉, 앞서 구해진 패턴들에게서 실제로 예측 로그가 발생할 확률을 계산하고 해당 값을 임계값에 비교해 낮을 경우 제거하고 높을 경우 유지하는 식으로 규칙을 생성하는 방식이다. 이때 해당 확률은 단편적으로 개수에 의해 계산되지 않고 샘플링 방식을 진행해 전역적인 사후확률 분포를 생성한 후 그 내부에서 기대 평균을 구하는 방식으로 진행된다.

베이지안 확률이란 사후확률을 의미하며 사전확률에 가능도를 곱한 값으로 이는 수식 3과 같다.

$$p(A|X) \propto p(X|A)p(A) \tag{3}$$

$p(X|A)$는 가능도로 어떤 사건(A)이 발생했을 때 X가 원인일 확률을 의미한다. 일반적으로 동전던지기와 같은 이진 문제에서는 이항분포를 사용하며 주사위 던지기와 같은 문제에서는 다항분포를 사용한다. $p(A)$는 사전확률로 어떤 사건(A)가 발생할 확률을 나타내는데 이때 확률분포 값은 무엇이든지 사용할 수 있으나 일반적으로 계산하기 쉽도록 가능도와 켤레사전분포를 이루는 분포를 선정한다. 그렇기 때문에 이항분포에서는 Beta 분포를 다항분포에서는 Dirichlet분포를 사전확률 분포로 이용한다. 마지막으로 $p(A|X)$는 사후확률로 X가 발생했을 때 사건(A)가 발생할 확률을 나타낸다. 이때는 사전확률 분포와 동일한 확률 분포를 사용한다. 사후확률은 가능도와 사전확률의 지속적인 곱으로 업데이트되며, 즉, 이는 기존에 갖고 있던 지식을 지속적인 관찰을 통해 더욱 정확한 확률로 업데이트함을 의미한다[16]. 본 연구에서는 해당 패턴이 다양한 예측 로그를 가질 수 있기 때문에 가능도로 다항분포, 사전확률 및 사후확률로 Dirichlet 분포를 이용한다. 디리클레 분포에서 구해진 확률값을 다항분포의 모수 값으로 활용하며 수식 4는 다항분포를 나타내며, 수식 5는 Dirichlet 분포를 나타낸다. 해당 수식에서 *n*은 총 횟수를 나타내며, *m*은 각각 상황 즉, 클래스 당 횟수를 나타내며, *p*는 각 상황 즉, 클래스의 확률을 나타낸다. α는 기존 각 상황(클래스)의 횟수를 나타낸다[17].

$$p(x) = \frac{\Gamma(n)}{\prod_{m=1}^{M} \Gamma(x_m)} \prod_{m=1}^{M} p_m^{x_m} \tag{4}$$

$$p(p|a) = \frac{\Gamma(\sum_{m=1}^{M} \alpha_m)}{\prod_{m=1}^{M} \Gamma(\alpha_m)} \prod_{m=1}^{M} p_m^{\alpha_m} \tag{5}$$

하지만 이런 확률은 우리가 갖고 있는 데이터의 단편적인 확률분포 밖에 알 수 없다. 데이터셋만으로 구해진 확률은 결국 큰 틀에서 발생하는 확률을 대표할 수 없고 이를 해결하기 위해 본 연구에서는 MCMC(Markov Chain Monte Carlo) 샘플링 방식을 이용해 앞서 구해진 단편적인 확률을 기반으로 하는 샘플을 생성하고 이들을 이용해 불확실한 사후확률분포를 근사한다[18]. 마지막으로 이렇게 근사된 사후확률 분포에서 점 추정을 진행한다. 점 추정을 통해 가장 높은 확률의 확률값을 구하고 이들을 특정 패턴이 발생했을 때 예측 클래스가 나올 확률로 사용하게 된다.

```
If <*> Got exception while serving <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*>
to <*> and BLOCK <*> ask <*> to replicate <*> to <*>
Then [[('Verification succeeded for <*>', 0.06458803116052418)], [('<*> Served block <*> to <*>', 0.0713800325741894)],
[('PacketResponder <*> for block <*> Interrupted.', 0.45527447718386804)], [('PacketResponder <*> <*> Exception <*>',
0.26663711554953545)]]

If <*> Got exception while serving <*> to <*> and PacketResponder <*> <*> Exception <*> and PacketResponder <*> for block <*>
Interrupted. and PacketResponder <*> <*> Exception <*> and PacketResponder <*> for block <*> Interrupted.
Then [[('BLOCK <*> ask <*> to replicate <*> to <*>', 0.0755771030086260 6)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*>
Then [[('Verification succeeded for <*>', 0.09119853834794736)], [('<*> Served block <*> to <*>', 0.081526185712421 3)],
[('PacketResponder <*> for block <*> Interrupted.', 0.07637933578708445)], [('Unexpected error trying to delete block <*>. BlockInfo not
found in volumeMap.', 0.21213090790766256)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.32677633571501774)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception
<*>
Then [[('Verification succeeded for <*>', 0.1309927504745695)], [('<*> Served block <*> to <*>', 0.11767112310629602)],
[('PacketResponder <*> for block <*> Interrupted.', 0.19920467742005393)], [('PacketResponder <*> <*> Exception <*>',
0.1143529802485043)], [('Unexpected error trying to delete block <*>. BlockInfo not found in volumeMap.', 0.10757065298274208)],
[('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.21632301711843963)], [('BLOCK <*> ask <*> to replicate <*> to <*>',
0.0641974961015551 7)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception
<*> and PacketResponder <*> <*> Exception <*>
Then [[('Verification succeeded for <*>', 0.21025956036279533)], [('<*> Served block <*> to <*>', 0.1653107551811 9248)],
[('PacketResponder <*> for block <*> Interrupted.', 0.14339375911843544)], [('PacketResponder <*> <*> Exception <*>',
0.11458014507134467)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.0836559517989892)]]
If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception
<*> and Verification succeeded for <*> Then [[('Verification succeeded for <*>', 0.4337891092480201)], [('<*> Served block <*> to <*>',
0.5084937324456472)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and Verification succeeded for <*>
Then [[('Verification succeeded for <*>', 0.2435696691356903)], [('<*> Served block <*> to <*>', 0.3535351930196276 6)], [('BLOCK <*>
NameSystem.allocateBlock: <*>. <*>', 0.1567462277967801)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and BLOCK <*> ask <*> to replicate <*> to <*> and <*> Served block <*> to <*>
Then [[('Adding an already existing block <*>', 0.0891352167060303 2)], [('Verification succeeded for <*>', 0.12278509798042844)], [('<*>
Served block <*> to <*>', 0.1289702072802902)], [('Unexpected error trying to delete block <*>. BlockInfo not found in volumeMap.',
0.13310409546571506)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.3479027724257388)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception <*>
Then [[('PacketResponder <*> for block <*> Interrupted.', 0.05177944902121236)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>',
0.06000720220407383)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception <*> and PacketResponder <*> for block <*>
Interrupted. and Verification succeeded for <*> and Verification succeeded for <*> and <*> Served block <*> to <*> and Verification
succeeded for <*>
Then [[('Adding an already existing block <*>', 0.10909507726655802)], [('Verification succeeded for <*>', 0.45791259404835166)], [('<*>
Served block <*> to <*>', 0.1548225881286940 4)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.14070365285743344)]]

If BLOCK <*> ask <*> to replicate <*> to <*> and PacketResponder <*> <*> Exception <*> and PacketResponder <*> for block <*>
Interrupted. and <*> Served block <*> to <*>
Then [[('Adding an already existing block <*>', 0.1623753253985055)], [('Verification succeeded for <*>', 0.19759886443584837)], [('<*>
Served block <*> to <*>', 0.2700058987084337)], [('BLOCK <*> NameSystem.allocateBlock: <*>. <*>', 0.2050091607580126)]]
```

(그림 4) 추출된 규칙 일부
(Figure 4) Part of extracted rules

4. 실험 및 성능

4.1 데이터셋

본 연구에서는 HDFS(Hadoop File System) 데이터셋을 활용한다. HDFS 데이터셋은 200개 이상의 아마존 EC2 호스트에서 Hadoop 기반의 mapreduce 작업을 수행해 나온 로그를 기록한 로그 데이터셋이다. 해당 로그 데이터셋은 11,197,954 개의 로그가 기록되어 있으며 이 중 2.9%가 이상행위로 라벨링 되었으며 라벨링은 Hadoop 시스템 전문가에 의해 지정되었다. 해당 데이터셋은 *block_id*라는 식별자가 존재하며 이를 기준으로 사용자당 로그 시퀀스를 생성한다. 이렇게 생성할 경우 총 575,079 개의 로그 시퀀스를 생성할 수 있다[19]. 학습 데이터셋으로는 총 4,855 개의 로그 시퀀스를 활용하였으며, 실험에서는 정상 553,366개, 이상행위 16,838개의 로그 시퀀스를 활용해 실험을 진행한다.

4.2 실험 및 평가

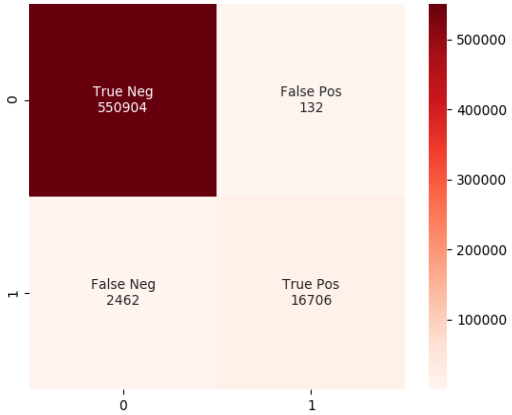
해당 연구에선 총 다섯 가지의 중요 파라미터를 사용했으며 이는 n-length sequence의 n 값, 지지도 임계값, 신뢰도 임계값, 추출 패턴 길이, 베이지안 확률 임계값이다. 결과적으로 다섯 가지의 적절한 파라미터를 선택함으로써, 판별력있고 중복되지 않은 규칙들을 생성할 수 있다. n-length sequence의 n은 학습되는 시퀀스의 길이를 나타내며, 지지도와 신뢰도 임계값은 지지도와 신뢰도 자체에서 활용되는 임계값을 나타낸다. 해당 임계값을 너무 높게 잡을 경우, 오탐율이 높아지며, 너무 낮게 잡을 경우, 미탐율이 높아진다. 추출 패턴 길이는 추출되는 규칙들 자체의 길이로 해당 규칙의 길이가 길 경우, 일반화된 규칙을 찾기가 어려우며, 짧을 경우 이상행위 탐지가 어렵다. 마지막으로 베이지안 확률 임계값은 규칙 자체의 임계값으로 해당 값이 너무 높을 경우, 오탐율이 높아지며, 낮을 경우, 미탐율이 높아진다. 이에 대한 정리는 표 1과 같다.

(표 1) 사용된 파라미터
(Table 1) Used parameter

파라미터	값
n-length sequence	10
지지도 임계값	0.001
신뢰도 임계값	0.01
추출 패턴 길이	2-7
베이지안 확률 임계값	0.05

해당 파라미터들을 이용해 총 2,701 개의 규칙을 추출했으며 그림 4는 추출된 규칙의 일부를 나타낸다. If와 then 사이는 조건을 나타내며, 해당 조건을 만족할 경우 then 뒤의 결과를 만족한다는 의미를 나타낸다. 즉, If 와 then 사이의 조건이 만족되고 then 뒤의 예측 로그 후보들에 실제 다음 로그가 존재한다면 정상, 존재하지 않거나 혹은 아예 매칭되는 규칙이 없을 경우 이상행위로 판단하게 된다.

그림 5는 제안 연구의 confusion matrix를 나타내며, 표 2는 해당 모델의 성능을 나타낸다. 성능은 정확도, F1score, TPR(True Postitive Rate), FPR(False Positive Rate) 을 기준으로 평가된다. 그림 5에서 0은 정상, 1은 이상행위를 의미하며, 숫자는 각 타입에 들어가는 테스트 샘플수를 의미한다.



(그림 5) 제안모델 confusion matrix
(Figure 5) Proposed model's confusion matrix

(표 2) 제안모델 성능평가
(Table 2) Performance in proposed model

모델	정확도	F1score	TPR	FPR
Proposed Model	99.54%	92.79%	99.22%	0.0044

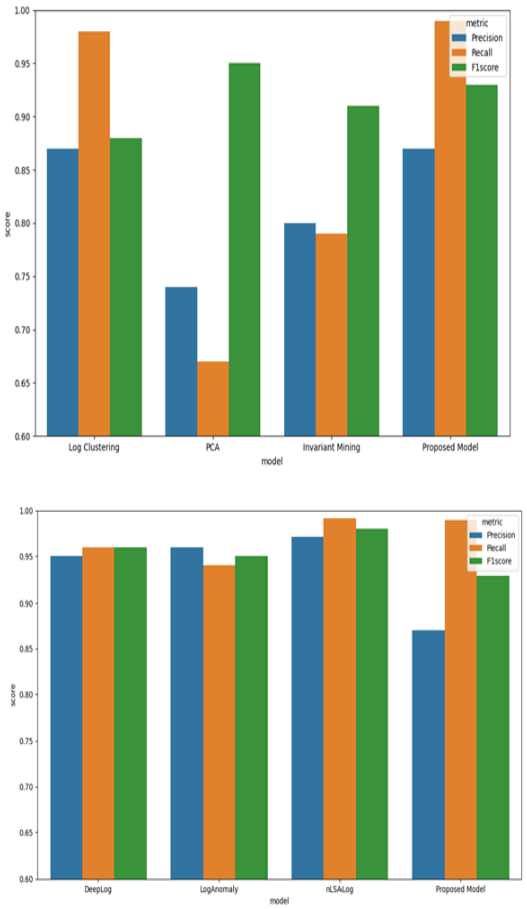
그림 6은 이전 연구들과 성능을 비교한 부분이다. 첫 번째 성능 비교는 머신러닝 알고리즘인 Log Clustering [11], PCA[9], Invariant Mining[20]과 비교했으며 F1socre 92.79%로 전체적으로 제안연구가 가장 좋은 성능을 보인다. 두 번째 성능비교는 딥러닝 방식인 DeepLog, LogAnomaly, nlsalog와 진행된다. nlsalog가 F1score 약 99%로 가장 좋은 성능을 보이며 제안 모델의 경우 전체적으로 99.22%의 가장 좋은 재현율(recall)과 87.15%의 가장 낮은 정밀도(precision)를 보인다.

제안 연구는 딥러닝과 비교에서 성능이 낮았지만 가장 높은 재현율을 기록했다. 해당 연구는 많은 리소스가 필요 없다는 점, 설명가능하다는 점에서 큰 장점을 지닌다. 이를 위해 4.3에서 각각의 테스트 샘플이 들어올 경우, 어떤 식의 설명을 제공하는지 설명한다.

4.3 설명 제공

본 절에서는 어떻게 설명성을 제공하는지 보여준다.

기본은 로그 템플릿이지만 편한 이해를 위해 템플릿이 아닌 템플릿에 매핑되는 숫자로 시각화를 제공한다. 즉, 3이면 "<*> Served block <*> to <*>"을 의미하게 된다.



(그림 6) 선행 연구 성능 비교
(Figure 6) Performance comparison between the proposed model and other models presented in the previous studies

가장 먼저 테스트 샘플이 정상이면서 모델이 정상이라고 예측했을 경우의 규칙이다. 샘플의 시퀀스는 “21, 4, 10, 8, 10, 8, 10, 8, 25, 25”이며 그림 7은 그에 매칭되는 규칙 2개를 보여준다. 해당 규칙을 보면 이와 매칭되는 규칙들에 의해 2, 22, 25 가 예측 다음 로그로 나타난다. 실제로 해당 시퀀스의 다음 로그는 25로 해당 규칙들에 매칭된다.

이외의 다른 샘플의 테스트 샘플의 경우 “8, 25, 10, 8, 25, 1, 1, 3, 3, 2” 의 로그 시퀀스가 존재할 때 이와 매칭되는 2개 로그는 그림 8과 같다. 해당 로그 시퀀스의 다음 로그로는 1이 있으며, 예측 로그에서는 1, 2, 3, 22가

```
If 10.0 and 8.0 and 10.0 and 10.0 and 25.0
Then [(22, 0.3631423063399608), (25, 0.5361762908334533)]
If 10.0 and 8.0 and 10.0 and 10.0 and 25.0 and 25.0
Then [(2, 0.11079915657539804), (22, 0.5980758952769912), (25, 0.22676930350382785)]
```

(그림 7) 샘플 “21,4,10,8,10, 8,10,8,25,25” 설명
(Figure 7) Explanation of sample “21,4,10,8,10, 8,10,8,25,25”

있다. 로그 1은 예측 로그 후보군에 존재하므로 정상을 정상으로 탐지한다.

```
If 8.0 and 25.0 and 8.0 and 3.0 and 2.0
Then [(1, 0.1537629991265742 6), (2, 0.2032385020048923 5),
(3, 0.2349966012215320 7), (22, 0.1210542624077329 8)]
If 8.0 and 25.0 and 1.0 and 3.0
Then [(1, 0.0764971124485224), (2, 0.3775890495389634),
(3, 0.1696924606486415 2), (22, 0.1064759651167051 7)]
```

(그림 8) 샘플 “8,25,10,8,25,1,1,3,3,2”” 설명
(Figure 8) Explanation of sample “8,25,10,8,25,1,1,3,3,2”

다음으로는 실제 로그가 이상행위이며 이를 이상행위로 판단한 경우에 대한 설명을 제공한다. 실제 로그 시퀀스는 “25, 25, 10, 8, 2, 3, 2, 17, 24, 4” 이며 이와 매칭되는 규칙은 그림 9와 같다. 해당 시퀀스의 다음 로그는 5이지만 이와 매칭되는 규칙에는 다음 로그로 5를 예측한 경우가 없다. 즉, 테스트 로그 시퀀스와 매칭되어도 예측 로그가 다르기 때문에 이상행위로 판단하게 된다.

```
If 8.0 and 2.0 and 2.0
Then [(2, 0.4999578397388805), (3, 0.3224962316923 2716), (22, 0.0988517133319 2846)]
If 2.0 and 3.0 and 2.0
Then [(2, 0.4041987147756664 3), (3, 0.1382193852135453 8),
(20,0.0921461767566771 5), (22, 0.1048928424507256 5)]
```

(그림 9) 샘플 “25,25,10,8,2,3,2,17,24,4” 설명
(Figure 9) Explanation of sample “25,25,10,8,2,3,2,17,24,4”

이번에는 잘못 판단한 경우에 대해 설명을 제공한다. 이상행위 시퀀스 “10, 8, 10, 8, 10, 8, 25, 25, 25, 22”가 존재하며 이에 대한 다음 로그 값은 22이다. 그림 10에서 보이는 규칙에 의해서 실제 다음 로그 값은 예측 후보군에 들어가게 되고 이로 인해 이상행위를 정상으로 판단하게 된다.

마지막으로 정상 로그 시퀀스지만 이를 이상행위로 판단한 경우에 대해 설명을 제공한다. 정상 테스트 로그 시퀀스인 “4, 4, 8, 10, 8, 10, 8, 25, 25, 25” 은 다음 로그로 10을 갖는다. 하지만 그림 11 규칙에 따라 10은 존재하지 않아 정상을 이상행위로 분류하게 된다.

```
If 10.0 and 8.0 and 8.0 and 10.0 and 25.0
Then [(2, 0.05422050376750179), (22, 0.20663140165550437),
(25,0.3890196536563644)]
```

(그림 10) 샘플 “10,8,10,8,10,8,25,25,25,22” 설명
(Figure 10) Explanation of sample “10,8,10,8,10,8,25,25,25,22”

```
If 8.0 and 8.0 and 10.0
Then [(2, 0.09748814687333407), (3, 0.08770295127701648),
(8, 0.07124695965760064), (22, 0.2296557417687906),
(25, 0.32493064903768354)]
```

(그림 11) 샘플 “4,4,8,10,8,10,8,25,25,25” 설명
(Figure 11) Explanation of sample “4,4,8,10,8,10,8,25,25,25”

앞선 설명처럼 본 연구는 결과뿐만 아니라 그렇게 판단한 규칙과 그 규칙 내 확률까지 제공함으로써, 더욱 정확하고 이해할 수 있는 설명을 제공하게 된다. 이는 크게 두 가지 이점을 확인할 수 있는데 가장 먼저 일반적인 정상행위와 악성행위의 패턴차를 규칙을 통해 알 수 있다. 즉, 시퀀스와 다음에 나올 로그 템플릿을 통해 정상일 경우, 이상행위일 경우의 패턴을 대략적으로 파악할 수 있다. 두 번째로 모델 개선이 가능하다. 정상을 이상행위로 혹은 이상행위를 정상으로 판단하는 경우처럼, 잘못 예측할 경우 그에 대한 근거가 확인가능하기 때문에 이를 개선 및 수정해 모델을 업데이트 할 수 있다.

5. 결 론

행위기반의 탐지가 가능하고 어디서든 구하기 쉽기 때문에 로그 이상탐지는 활발히 연구되고 있는 추세이다. 특히 딥러닝을 사용하면서 높은 성능과 탐지율을 보여준다. 하지만 아무리 성능이 뛰어나더라도 정상으로 혹은 이상행위로 판단한 근거를 제공하지 못할 경우, 잘못된 이유로 우연히 뛰어난 성능을 보여줘도 이를 판단할 수 없다. 실제로 이를 반영하듯이 유럽 연합에서는 인공지능을 사용할 때, 그 근거를 제공하지 못할 경우, 필드에서 사용할 수 없도록 법을 제정하기도 했다[6]. 본 연구에서 제안하는 방식은 다른 딥러닝 기반 연구들에 비해 성능이 떨어진다. 하지만 설명을 제공받음으로써 무엇이 잘못 판단되었는지 알 수 있으며 이는 새로운 피드백으로 모델의 개선점을 보여준다. 앞으로 본 연구는 크게 두 가지 방향으로 연구를 진행시켜 나갈 예정이다. 첫 번째로는 최적화를 진행할 예정이다. 여기서 선정된 모든 파라미터는 모두 여러 번의 실험으로 선정되었다. 그래서 본 연구에선 이 점을 개선하기 위해 베이즈 최적화[21], 유전알고리즘[22] 등의 최적화 방법을 적용해 알맞은 파라미터를 선정함으로써 모델을 발전시켜 나갈 계획이다. 두 번째로는 성능을 개선할 계획이다. 규칙은 어떤 인공지능 모델보다 설명성이 높고 투명하다. 하지만 그림 6에서 볼 수 있듯이, 딥러닝 기술에 비해서는 확장성과 성능 차원에서 제한점이 분명히 존재한다. 즉, 딥러닝의 경우, 인간의 파라미터 개입없이 스스로 학습을 진행하고 또한 어느정도 학습이 진행되면 규칙에 비해 유연하게 판단하는게 가능하다. 하지만 규칙은 해당 규칙에 매칭되지 않을 경우 즉, 새로운 패턴이 들어올 경우 정확한 판단이 어렵다. 그렇기 때문에 다음 연구에서는 딥러닝 및 최신기술을 활용하되 설명성을 추가한 연구를 진행해 더 높은 성능과 설명성을 동시에 제공할 모델을 생성할 계획이다.

참고문헌(Reference)

[1] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298, 2017. https://doi.org/10.1145/3133956.3134015

[2] W. Meng, Y. Liu, Y. Zhu, S. Zhang, and D. Pei et al. "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs." IJCAI. Vol. 7. pp. 4739-4745, 2019. http://doi.org/10.24963/ijcai.2019/658

[3] R. Yang, D. Qu, Y. Gao, Y. Qian, and Y. Tang, "NLSALog: An anomaly detection framework for log sequence in security management." IEEE Access, Vol. 7., pp. 181152-181164, 2019. http://doi.org/10.1109/ACCESS.2019.2953981

[4] X. Zhang, Y. Xu, Q. Lin, B. Qiao, and H. Zhang et al., "Robust log-based anomaly detection on unstable log data." Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 807–817, 2019. https://doi.org/10.1145/3338906.3338931

[5] D. Gunning, and D. Aha, "DARPA's explainable artificial intelligence (XAI) program." AI Magazine Vol.40, No.2, pp. 44-58, 2019.
https://doi.org/10.1609/aimag.v40i2.2850

[6] M. Hind, "Explaining explainable AI." XRDS: Crossroads, The ACM Magazine for Students, Vol.25 No.3 pp. 16-19, 2019.
https://doi.org/10.1145/3313096

[7] P. He, J. Zhu, Z. Zheng, and M.R. Lyu, "Drain: An online log parsing approach with fixed depth tree." 2017 IEEE International Conference on Web Services (ICWS). IEEE, pp. 33-40, 2017.
https://doi.org/10.1109/ICWS.2017.13

[8] M. Du and F. Li, "Spell: Streaming parsing of system event logs." 2016 IEEE 16th International Conference on Data Mining (ICDM) IEEE, pp. 859-864, 2016.
https://doi.org/10.1109/ICDM.2016.0103

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M.I. Jordan, "Detecting large-scale system problems by mining console logs." Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp.117–132, 2009.
https://doi.org/10.1145/1629575.1629587

[10] M. Landauer, M. Wurzenberger, F. Skopik, G. Settanni, and P. Filzmoser, "Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection." computers & security Vol. 79, pp. 94-116, 2018.
https://doi.org/10.1016/j.cose.2018.08.009

[11] Q. Lin, H. Zhang, J.G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems." 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). IEEE, pp. 102-111, 2016.
http://dx.doi.org/10.1145/2889160.2889232

[12] C. C. Aggarwal, M. A. Bhuiyan, and M. A. l. Hasan "Frequent pattern mining algorithms: A survey." Frequent pattern mining. Springer, Cham, pp.19-64, 2014.
https://doi.org/10.1007/978-3-319-07821-2_2

[13] X. Yan, J. Han, and R. Afshar, "Clospan: Mining: Closed sequential patterns in large datasets." Proceedings of the 2003 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, pp. 166-177, 2003.
https://doi.org/10.1137/1.9781611972733.15

[14] J. Wang, and J. Han, "BIDE: Efficient mining of frequent closed sequences." Proceedings. 20th international conference on data engineering. IEEE, pp. 79-90, 2004.
https://doi.org/10.1109/ICDE.2004.1319986

[15] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model." Annals of Applied Statistics 9.3 Vol. 9, No. 3, pp. 1350-1371, 2015.
https://doi.org/10.1214/15-AOAS848

[16] W. M. Bolstad, and J. M. Curran Curran. Introduction to Bayesian statistics. John Wiley & Sons, 2016.
https://doi.org/10.1002/9781118593165

[17] C. Forbes, M. Evans, N. Hastings, and B. Peacock, Statistical distributions. John Wiley & Sons, 2011.
https://doi.org/10.1002/9780470627242

[18] S. Brooks, A. Gelman, G. Jones, and X. L. Meng, eds. Handbook of markov chain monte carlo. CRC press, 2011.
https://doi.org/10.1201/b10905

[19] W. Xu, L. Huang, A. Fo, D. Patterson and M. I. Jordan, "Detecting large-scale system problems by mining console logs." Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. pp. 117-132, 2009.
https://doi.org/10.1145/1629575.1629587

[20] J. G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining Invariants from Console Logs for System Problem Detection." USENIX Annual Technical Conference, pp. 1-14, 2010.
https://doi.org/10.5555/1855840.1855864

[21] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm." Proceedings of the genetic and evolutionary computation conference GECCO-99. Vol. 1, pp. 525-532, 1999.
https://dl.acm.org/doi/10.5555/2933923.2933973

[22] K. F. Man, K. S. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]." IEEE transactions on Industrial Electronics Vol. 43, No. 5 pp. 519-534, 1996.
https://doi.org/10.1109/41.538609

◐ 저 자 소 개 ◑

윤 지 영(Jiyoung Yun)
2020년 가천대학교 컴퓨터공학과(공학사)
2020년~현재 가천대학교 일반대학원 소프트웨어학과 석사과정
관심분야 : XAI, Machine Learning, Statistics, Intrusion Detection
E-mail : apfhd9043@naver.com

신 건 윤(Gun-Yoon Shin)
2017년 가천대학교 인터랙티브 미디어 융합학과 학사
2018년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2018년~현재 가천대학교 컴퓨터공학과 박사과정
관심분야 : 기계 학습, 악성코드 분석, 공격자 식별, 저자 분석, 인공지능
E-mail : tlsrjsdbs@gmail.com

김 동 욱(Dong-Wook Kim)
2015년 가천대학교 컴퓨터공학과(공학사)
2017년 가천대학교 일반대학원 컴퓨터공학과(공학석사)
2017년~현재 가천대학교 컴퓨터공학과 박사과정
관심분야 : Cyber Security, Data Mining, Artificial intelligence
E-mail : kog7306@naver.com

김 상 수(Sang-Soo Kim)
1997년 경북대학교 전자공학과(공학사)
2003년 경북대학교 컴퓨터공학과(공학석사)
2003년~현재 국방과학연구소 연구원
관심분야 : Cyber Security, Cyber Situation Awareness, Artificial Intelligence
E-mail : wisdory@naver.com

한 명 묵(Myung-Mook Han)
1980년 연세대학교 공과대학(공학사)
1987년 뉴욕공과대학교 대학원 컴퓨터공학과(공학석사)
1997년 오사카시립대학교 대학원 정보공학부(이학박사)
1998년~현재 가천대학교 소프트웨어학과 교수
관심분야 : 정보보호, 알고리즘, 데이터 마이닝, 기계 학습
E-mail : mmhan@gachon.ac.kr