JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 2, pp. 89-97, February 2023
https://doi.org/10.9708/jksci.2023.28.02.089

An Accurate Log Object Recognition Technique

Jiho Ju*, Byungchul Tak**

*Student, School of Computer Science and Engineering, Kyungpook National University, Daegu, Korea
**Professor, Dept. of Computer Science and Engineering, Kyungpook National University, Daegu, Korea

[Abstract]

In this paper, we propose factors that make log analysis difficult and design technique for detecting various objects embedded in the logs which helps in the subsequent analysis. In today's IT systems, logs have become a critical source data for many advanced AI analysis techniques. Although logs contain wealth of useful information, it is difficult to directly apply techniques since logs are semi-structured by nature. The factors that interfere with log analysis are various objects such as file path, identifiers, JSON documents, etc. We have designed a BERT-based object pattern recognition algorithm for these objects and performed object identification. Object pattern recognition algorithms are based on object definition, GROK pattern, and regular expression. We find that simple pattern matchings based on known patterns and regular expressions are ineffective. The results show significantly better accuracy than using only the patterns and regular expressions. In addition, in the case of the BERT model, the accuracy of classifying objects reached as high as 99%.

▸ **Key words**: Log Analysis, Pattern Recognition, GROK, Text classification, BERT

[요 약]

본 논문에서는 로그 분석을 어렵게 하는 요인을 제안하고 이후 분석에 도움을 주는 로그 내 다양한 객체 인식 기법을 설계한다. 오늘날의 IT 시스템에서 로그는 다수의 고급 AI 분석 기술의 핵심적인 원천 데이터이다. 로그에는 유용한 정보가 많이 포함되어 있지만 로그는 본질적으로 반구조화되어 있기 때문에 로그 내 유용 정보에 기술을 직접적으로 적용시키기 어렵다. 로그 분석을 방해하는 요소는 file path, identifier, json 등 다양한 객체이다. 이러한 객체에 대한 BERT기반의 패턴 인식 알고리즘을 설계하고 객체 인식을 수행한다. 본 실험에서 정의한 패턴 인식 알고리즘은 객체의 정의, GROK 패턴, 그리고 정규 표현식에 기반한다. 기존에 알려진 패턴과 정규 표현식을 기반으로 한 간단한 패턴 매칭이 효과적이지 않다는 것을 확인할 수 있었다. 그 결과 기존 패턴과 정규 표현식만을 사용하는 것보다 훨씬 나은 정확도를 보여준다. 또한, BERT 모델의 경우 인식 객체 이외의 객체를 분류하는 정확도가 99%에 달하는 것을 확인할 수 있다.

▸ **주제어**: 로그 분석, 패턴 인식, GROK, 텍스트 분류, BERT

• First Author: Jiho Ju, Corresponding Author: Byungchul Tak
*Jiho Ju (jihoju@knu.ac.kr), School of Computer Science and Engineering, Kyungpook National University
**Byungchul Tak (bctak@knu.ac.kr), Dept. of Computer Science and Engineering, Kyungpook National University
• Received: 2023. 01. 16, Revised: 2023. 02. 02, Accepted: 2023. 02. 06.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

로그 분석은 소프트웨어 시스템의 기록인 로그를 어떠한 문제를 해결하기 위한 정보로 변환하는 과정이다[1-2]. 로그 데이터는 시스템 상태 및 성능 문제를 이해하는 데 중요한 자원이다[3]. (1) 디버깅 문제: 응용 프로그램 및 시스템 내 특정 오류 메시지 또는 이벤트를 확인[4], (2) 성능 분석: 리소스 활용률을 최적화하고 성능 문제 및 시스템을 파악[4], (3) 보안 분석: 모든 조직의 애플리케이션 보안을 관리[4], (4) 예측 분석: 로그 및 이벤트에 의해 생성된 정보를 활용한 위협 및 이상 징후 미래 트래픽 패턴 예측[4], (5) 사물인터넷(IoT) 및 로깅: 스마트 장치의 현재 상태와 성능을 파악[4]. 이처럼 로그 분석은 다양한 분야에서 요구되고 있다.

현대 소프트웨어 시스템의 규모와 복잡성이 증가함에 따라 로그의 양이 폭발적으로 축적됨에 따라 빅데이터 로그 분석이 요구되고 있다[5]. 또한, 클라우드 및 데이터 인프라 관련 애플리케이션의 크기와 복잡도가 증가함에 따라 로그의 구조와 로그 내 객체가 더욱 다양해짐에 따라 유연한 로그 분석 또한 요구되고 있다[5-6].

빅데이터로 존재하는 로그를 효과적으로 분석하는 방법은 여전히 큰 과제이다[5]. 다양한 애플리케이션 및 기기마다 로그 포맷이 다르다. 일관되지 않은 로그 포맷은 검색, 분류, 그리고 분석을 어렵게 만든다[4]. 또한, 로그 내 일관되지 않은 시간 포맷 역시 똑같은 분석에 영향을 미친다. 일관되지 않은 구조는 로그를 분류하고 중요한 정보 추출에 있어 치명적인 문제를 야기할 수 있다. 그렇기에 정확하고 신속한 로그 분석 기술이 요구되고 있다.

현재 모든 로그 분석은 프로그래머가 정의한 로그 문자열 패턴 집합을 마이닝 하는 Log Template Discovery (LTD) 기술에 기반한다[6]. 기존에 존재하는 로그 분석은 주요 분석 알고리즘을 개발하기 위해 LTD 기술을 사용한다[6]. 로그 템플릿은 정적인 부분과 가변적 부분으로 이루어져 있다. 하나의 애플리케이션에서의 전체 로그 집합은 오직 특정 개수의 로그 템플릿 내 가변적 부분만이 바뀌며 반복되는 형태를 갖는다[6]. 로그 템플릿 내 가변적 부분으로 Date, Time, URI, IP Address, Filepath, JSON, XML 등의 객체가 있다. 로그 분석은 수많은 로그를 각각의 로그 템플릿으로 분류하는 것에서부터 시작된다. 하지만, 로그 템플릿 내 가변적 부분은 이러한 로그 분류를 어렵게 만든다. 이러한 가변적 부분을 각 객체 타입으로 정확하게 필터링 혹은 파싱 하는 기술이 필요하다.

로그 파싱 시, 다양한 객체 인식 시 가장 많이 사용하는 기술로 GROK이 존재한다. GROK은 ELK Stack 중 Logstash에 적용되어 로그 데이터 전처리 기술에 활용된다[7-8]. GROK은 패턴을 기반으로 비교적 높은 정확도로 로그 내 특정 객체 인식이 가능하다는 장점이 있다. GROK 패턴 내 정의되지 않은 객체 또는 인식하고자 하는 패턴과 비슷한 구조의 객체를 인식하지 못하거나 잘못된 객체를 인식할 수 있다는 한계점이 존재한다. 현재 여러 기업에서 로그를 수집하고 시각화하기 위해 ELK Stack을 도입하여 사용하고 있다[9]. 또한, 데이터의 유형 탐지[10], 데이터 및 로그 파싱 기술[11-12]을 연구하는데 GROK 기술이 사용되고 있다. 이처럼 GROK을 적용한 기술이 기업 및 연구에 활발히 사용되기 때문에 GROK이 갖는 한계점을 개선할 필요가 있다.

GROK과 같은 패턴 인식만으로 정확하게 인식하지 못하는 가장 큰 문제는 명확한 객체 정의가 없다는 것에서 비롯된다. GROK 패턴으로 Filepath 객체 인식 결과 다음 문제를 발견할 수 있었다. 첫째, Filepath 객체와 유사한 객체(API 주소, RSA Key, URI, IP, Route Path, Date, Hadoop Topology 등)를 정확하게 구별하지 못하고 예외적인 인식 결과가 나타난 것을 확인하였다. 둘째, 변수값이 포함된 Filepath 객체의 경우, 변수 포맷에 따라 인식률이 달라졌다. 이러한 문제점이 발생한 이유는 Filepath 객체의 명확한 정의가 존재하지 않기 때문이다.

명확한 정의가 없는 객체로 인해 패턴 인식 기술은 한계를 갖는다. Table 1은 OS 별 Filepath 객체 구조가 상이한 예시를 보여준다. Table 2는 Filepath 객체와 유사한 객체 예시를 보여주며 Table 3은 Filepath 객체 인식 예외를 발생시키는 예시를 보여준다. 이러한 문제를 해결하기 위해 객체 패턴 인식뿐만 아니라 로그 내 객체 주변의 단어를 고려한 객체 인식 기술을 설계하였다.

패턴 인식만으로 한계점이 존재하는 객체의 경우 객체 주변 단어를 고려하기 위해 BERT[13] 모델을 하여 객체 인식 기술을 개발하였다. BERT는 텍스트 문장이 주어졌을 때 해당 문장을 분류하는데 높은 성능을 보인다[13]. BERT를 활용한 로그 이상 탐지 기술에 대한 연구[14-16]가 활발히 진행되었으며 해당 기술은 뛰어난 성능을 보인다. 이러한 성능과 활용 가능성에 착안하여 본 실험에 적용하였다. 로그 내 전처리된 단어로 이루어진 텍스트 문장 내 객체가 실제로 Filepath 객체인지를 확인하는 분류 단계에 BERT 모델을 활용하였다.

Table 1. Different Path Structure for each OS

OS	Shell	Directory separator	Examples
Unix(incl. macOS)	Unix shell	/	"/home/user/logs/hadoop.log"
Microsoft Windows	cmd.exe	/ or \	"C:\user\logs\hadoop.log" or "/user/logs/hadoop.log"
	Powershell	/ or \	"C:\user\logs\hadoop.log" or "\\home\user\logs\hadoop.log"

Table 2. Object Similar to the Filepath Object

Type	Object
API	"/v2/metadefs/namespaces/OS::Compute::Watchdog"
API	"/s3tokens"
API	"/ec2tokens"
API	"PATH_INFO: /v3/auth/tokens"
API	"SCRIPT_NAME: /identity"
API	"GET /v2/images"
RSA Key	"ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDFFhOE6Qp1OPVgucRavRsx1riSiVIottcZSMAqjER2XD2/pLs043WwCeSoVHuv6GvOkztrOce0C98ItMrDbhI8Vn2gGQKvTL8+DiaoGXIxrN61xYSqFT748RhCX00mtdkBU8MSKGD4+/fIWYCyX02GHMpE3zatYeOU8K6RVVBmfKHvs+nI/xSw+Pc91rCKw+2/vNv2j56rWbE6Zr0Ig1YhEms7ci4o+SvaAFWUEjWIW82Yg0YFIDBp1yI0IDcrkXAt9MY6O035TpBrIzb//oUwNa2YIIhxv5HvrvlJQG+XZY/w7GZYgZ2z1oafIBd8FwPm+ZDA4ZK5m8rCAy477jB/"
Route path	"/networks{.format}"
Route path	"/{project_id}/limits"
URL	"service:jmx:rmi://127.0.0.1/jndi/rmi://127.0.0.1:7199/jmxrmi"
IP	"localhost/127.0.0.1"
IP	"master/172.17.0.2:9000"
IP	"fde3:ce86:6ac::/64"
IP	"/172.17.0.2:8032"
Date	"19/02/17"
Hadoop Topology	"/default-rack/155.230.91.227:50010"
Throughput	"0.093KiB/s"
Heap Size	"1.421GiB/1.421GiB"
Versions	"3/v3", "4/v4", "5/v5-beta"
Word	"N/A"
Word	"15/s
MIME TYPE	"Accept: */*"
MIME TYPE	"application/javascript"
MIME TYPE	"text/jscript"
MIME TYPE	"audio/wave"

Table 3. Filepath Object with Variable

Filepath Object
"{{JAVA_HOME}}/bin/java"
"{{PWD}}/tmp"
"$JAVA_HOME/bin/java"
"$HADOOP_HDFS_HOME/share/hadoop/hdfs/*"
"{{PWD}}/__spark_conf__/__spark_conf__.properties"

본 논문은 2장에서 객체 인식에 필요한 기술을 분석하고 소개한다. 3장에서는 본 연구에서 소개하는 Date, Time, URI, IP, Filepath, JSON 객체 인식 기술을 소개하고 기존 패턴과 비교한다. 그리고 본 연구에서 중점적으로 다루는 Filepath 객체 인식 정확도 향상을 위해 적용한 객체 분류 실험 결과에 대한 분석을 진행한다. 마지막으로 5장에서는 본 논문의 결론을 정리한다.

II. Preliminaries

1. Background

1.1 Path

Path는 디렉터리 구조에서 위치를 고유하게 식별하는데 사용되는 문자열이다. 구분 문자로 구분된 구성 요소가 각 디렉터리를 나타내는 디렉터리 트리 계층 구조를 따라 구성된다. 구분 문자는 가장 일반적으로 구분 문자로 슬래시(/), 백슬래시(\), 그리고 콜론(:)이 있다. 일부 운영체제에서 다른 구분 기호를 사용할 수 있기 때문에 운영체제마다 상이할 수 있다. 이로 인해 Path의 명확한 정의가 존재하지 않고 운영체제별 상이한 포맷 구조 때문에 패턴 인식의 한계가 존재한다.

1.2 GROK

GROK은 구조화되지 않은 로그 데이터를 구조화된 데이터로 파싱 하는 기법이다[17]. 120개 이상의 객체를 정규표현식으로 생성한 패턴을 GROK 패턴이라 한다[17]. GROK 패턴의 구문은 %{SYNTAX:SEMANTIC}으로 표현된다. SYNTAX는 텍스트와 매칭되는 패턴 이름이며, SEMANTIC은 SYNTAX로 매칭되는 객체에 부여하는 이름이다. 로그 내 Filepath 객체 인식을 위한 GROK 패턴은 %{PATH}로 정의되어 있다. %{PATH:filepath} 패턴에서 SYNTAX는 PATH이며 SEMANTIC은 filepath가 된다. 현재 GROK 패턴에 정의되지 않은 객체는 기존 패턴에 추가 정규 표현식을 더해 커스터마이징이 가능하다. 본 실험에서 기존 패턴으로 인식에 한계가 있는 객체에 대해 기존 GROK 패턴을 커스터마이징하여 사용하였다.

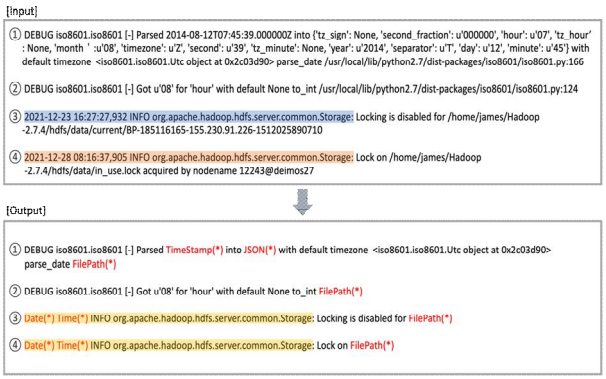

Fig. 1. Object Recognition Result

Table 4. Time GROK Pattern Exception

GROK Pattern	Not Time Object	Recognition Result
%{TIME}	"000:00:00:00.000"	"00:00:00:00"
%{TIME}	"0000:00:06.2"	"00:00:06.2"
%{TIME}	"02:42:ac:ff:fe:11:00:02"	"11:00:02"

1.3 Log Parsing Tools

로그 파싱은 구조화되지 않은 로그 데이터에서 속성(키:값 쌍)을 추출하는 프로세스이다. 즉, 원시 로그 메시지를 자동으로 구문 분석의 과정이다[18]. 로그 파싱 기술은 대표적으로 Lognroll[6], Drain[18], SLCT[19], IPLoM[20], LogCluster[21], LKE[22], MoLFI[23], Spell[24], LenMa[25], LogMine[26], LogSig[27], SHISO[28], AEL[29] 등이 있다. Drain은 고정 깊이 구문 분석 트리를 구축하는 것을 기반으로 하는 최근 기술 중 하나이다[6]. Drain은 자동 로그 파싱 기술로 Online에 특화되어 있다[18]. 전통적인 로그 파싱 기술은 정규식에 크게 의존하는 반면에, Drain은 고정 깊이 트리를 사용한다는 점에서 로그의 양이 방대한 환경에서 성능이 매우 뛰어나다는 장점을 가진다[18]. 하지만, Drain[18]은 로그 메시지 내 다양한 객체 인식 정확도가 낮다는 한계를 갖는다.

2. Software Specification

본 실험은 macOS Ventura 13.0.1 운영체제를 사용하는 PC에서 진행되었다. 실험을 위해 Python 3.9.13을 사용하였으며, 가상환경으로 Anaconda 4.13.0을 사용하였다. Logstash 1.4.1 내 정의된 GROK 패턴을 참고하였으며, Python 라이브러리 pygrok 1.0.0 버전을 사용하였다. Tensorflow 2.9.2를 사용하였으며 TensorFlow Hub에서 사전 트레이닝된 Uncased BERT 모델인 bert_en_uncasedL-12_H-768_A-12/2를 사용하였다.

Table 5. Date GROK Pattern Exception

GROK Pattern	Date Object	Recognition Result
%{DATE}	"11/Aug/2014"	Not Recognized
%{DATE}	"Mon, 11 Aug 2014"	"11 Aug 2014"
%{DATE}	"1 Mar 2016"	"Mar 2016"
%{DATE}	"Wed Jun 14"	"Jun 14"
%{DATE}	"March 2016"	Not Recognized
%{DATE}	"July 20"	Not Recognized
%{DATE}	"15/October/2019"	Not Recognized
%{DATE}	"2020-11-11"	"20-11-11"

III. The Proposed Scheme

Fig. 1은 본 논문에서 다루는 로그 메시지 내 특정 객체를 인식 결과이다. 한 로그 문자열 객체가 입력으로 주어지면 Date, Time, URI, IP, Filepath, JSON 객체를 인식한다. 기본적으로 정규 표현식과 GROK 패턴을 활용해 직접 구현한 패턴 인식 알고리즘으로 객체 인식을 수행하였다. 본 실험에서 Cassandra (51,318개), MongoDB (144,269개), Hadoop(222,955), OpenStack(420,554), Spark(150,530개) 애플리케이션에서 발생한 총 989,626개의 로그 데이터를 사용하였다.

Date, Time, URI, IP 객체 인식을 위해 GROK 패턴을 활용한 패턴 인식 알고리즘을 구현하였다. Date, Time 객체의 경우, 본 실험에서 사용한 로그 데이터 기준으로 기존 GROK 패턴으로 인식 결과 예외 객체들이 존재하였다.

Table. 4는 %{TIME} GROK 패턴이 Time 객체가 아닌 객체를 Time 객체로 잘못 인식하는 예시이다. 이러한 문제를 해결하기 위해 Table. 4에 해당하는 객체를 Time 객체 인식 전 제거하는 방법을 활용했다. 커스텀 GROK 패턴 "(?<sub_time>0{3,}:0{2,}:|0{3,}:|%{MAC}([:]\d*)*)"으로 로그 내 인식되는 객체를 제거 후 새로 정의한 GROK 패턴으로 Time 객체를 인식하여 위의 예외를 처리하였다.

Table. 5는 %{DATE} GROK 패턴이 Date 객체임에도 인식하지 못하거나 Date 객체를 정확하게 인식하지 못하는 예시이다. 이러한 문제를 해결하기 위해 커스텀 GROK 패턴 “(?<date>%{MONTHDAY}/%{MONTH}/%{YEAR}|%{YEAR}[/-]%{MONTHNUM}[/-]%{MONTHDAY}|%{DAY}([\S])? %{MONTHDAY} %{MONTH} %{YEAR}|%{MONTHDAY} %{MONTH} %{YEAR}|%{MONTH} %{YEAR}|%{YEAR} %{MONTH} %{MONTHDAY}|%{DAY} %{MONTH} %{MONTHDAY}|%{MONTH} %{MONTHDAY})“을 생성 후 Date 객체를 인식하여 위의 예외를 해결하였다.

Table 6. Similarity between URI and Filepath

Type	Object
URI	"GET /placement/resource_providers/9bbad80a-94c9-4d34-ada3-4cae3af1baf6/allocations"
Filepath	"/etc/cinder/rootwrap.conf"

Table 7. URIPATH GROK Pattern Exception

GROK Pattern	Not URI Object	Recognition Result
%{URIPATH}	"/etc/cinder/rootwrap.conf"	"/etc/cinder/rootwrap.conf"
%{URIPATH}	"/opt/stack/placement/placement/requestlog.py:38}}"	"/opt/stack/placement/placement/requestlog.py:38}}"

Table 8. IP GROK Pattern Result

GROK Pattern	IP Object	Recognition Result
%{IP}	"10.0.0.0/24"	"10.0.0.0"
%{IP}	"/0.0.0.0"	"0.0.0.0"
%{IP}	"0.0.0.0:8042"	"0.0.0.0"
%{IP}	"0.0.0.0/0.0.0.0:0"	"0.0.0.0", "0.0.0.0"
%{IP}	"NORMAL:155.230.91.227:50010"	"155.230.91.227"
%{IP}	"ACCEPT-localhost/127.0.0.1"	"127.0.0.1"

Table 9. IP Pattern Recognition Algorithm Result

IP Object	Recognition Result
"10.0.0.0/24"	"10.0.0.0/24"
"/0.0.0.0"	"/0.0.0.0"
"0.0.0.0:8042"	"0.0.0.0:8042"
"0.0.0.0/0.0.0.0:0"	"0.0.0.0/0.0.0.0:0"
"NORMAL:155.230.91.227:50010"	"NORMAL:155.230.91.227:50010"
"ACCEPT-localhost/127.0.0.1"	"ACCEPT-localhost/127.0.0.1"

Table. 7은 %{URIPATH} GROK 패턴이 URI 객체가 아닌 객체를 URI 객체로 잘못 인식하는 예시이다. 이러한 문제가 발생하는 이유는 Table. 6처럼 URI 객체와 Filepath 객체는 매우 유사한 형식을 가지고 있다. 이러한 이유로 로그 메시지 내 Filepath 객체가 있다면 URI 객체로 잘못 인식되는 문제가 발생한다. 이러한 문제를 해결하기 위해 본 실험에서는 HTTP 메서드의 정의 존재 여부로 URI 객체와 Filepath 객체를 구분하였다. 이를 적용한 커스텀 GROK 패턴 "(?<url>%{URI}|GET %{URIPATH} [\S]*|POST %{URIPATH}[\S]*|PUT %{URIPATH} [\S]*|DELETE %{URIPATH}[\S]*)"으로 URI 객체를 인식하여 위의 문제를 해결하였다.

Table. 8은 %{IP} GROK 패턴의 인식 결과이다. 본 실험은 로그 분석 및 분류를 어렵게 하는 로그 내 객체를 인식하기 위함에 중점을 둔다. 기존 Grok 패턴이 인식하는 IP 객체보다 더 넓은 범위로 IP 객체를 인식하여 원활한 로그 분석 환경을 제공한다.

Table. 9는 본 실험에서 제시하는 IP 패턴 인식 알고리즘을 기반으로 IP 객체를 인식한 결과이다. IP 패턴 인식 알고리즘은 다음 과정을 수행하며 로그 내 IP 객체를 인식한다. (1) 로그 메시지 내 '\n' 문자열과 본 연구에서 정의한 커스텀 URI GROK 패턴 "(?<url>%{URI}|GET %{URIPATH} [\S]*|POST %{URIPATH}[\S]*|PUT %{URIPATH} [\S]*|DELETE %{URIPATH}[\S]*)"으로 인식된 URI 객체를 제거한다. (2) 커스텀 GROK 패턴 "(?<ip>%{HOSTNAME}[/:]%{IPV4}([:](?:[0-9]*))?|[/]?%{IPV4}([:](?:[0-9]*))?(/[0-9]{2})?|[^-]%{IPV4}([/]%{IPV4})?([:](?:[0-9]*))?(/[0-9]+)?|%{IPV6}(/[0-9]+)?|%{IP})"으로 로그 메시지 내 IP 객체를 인식한다. (3) 인식된 IP 객체 중 양옆의 문자열 '-', ':', '&', '[', ']', '(', ')', '=', '@', ',', '\s(공백)'이 있다면 제거한다.

JSON 객체 인식의 경우 JSON 공식 문서에 정의된 문법을 기반으로 인식 알고리즘을 구현하였다. Python 내장 모듈 중 json 내 loads() 함수를 통해 인식된 JSON 객체 검증을 수행하였다.

본 연구에서 중점적으로 다루는 Filepath 객체 인식은 다음 과정을 수행하다. (1) 한 개의 문자열 로그가 input으로 들어온다. (2) Filepath 객체에 포함되지 않는 구두점('?', '%', '*', '|', '&, '<', '>', ",", ';', '=', 공백, '{', '}', '[', ']')을 기준으로 파이썬 split() 함수로 단어 단위로 분리한다. (3) 분리된 단어들에 대해 파이썬 내 strip() 함수로 단어 앞, 뒤에 모든 구두점을 제거하며 전처리 과정을 수행한다. (4) 분리된 각 단어들에 대해 앞, 뒤 단어 2개씩 총 5개의 단어로 문장을 구성한다. (5) 생성된 문장들에 대해 학습 모델이 Filepath 객체 여부를 분류한다. Filepath 객체 분류의 대상은 5개의 단어로 구성된 문장의 가운데 단어이다. 5개의 단어로 이루어진 문장에서 가운데 단어가 Filepath 객체이면 가운데 단어가 Filepath 객체라 분류한다. (6) 인식 정확도 향상을 위해 Filepath 객체로 분류된 문장의 가운데 단어를 Filepath 커스텀 GROK 패턴을 활용한 패턴 인식 알고리즘을 통해 검증한다. (7) 검증이 완료된 단어를 Filepath 객체로 인식한다.

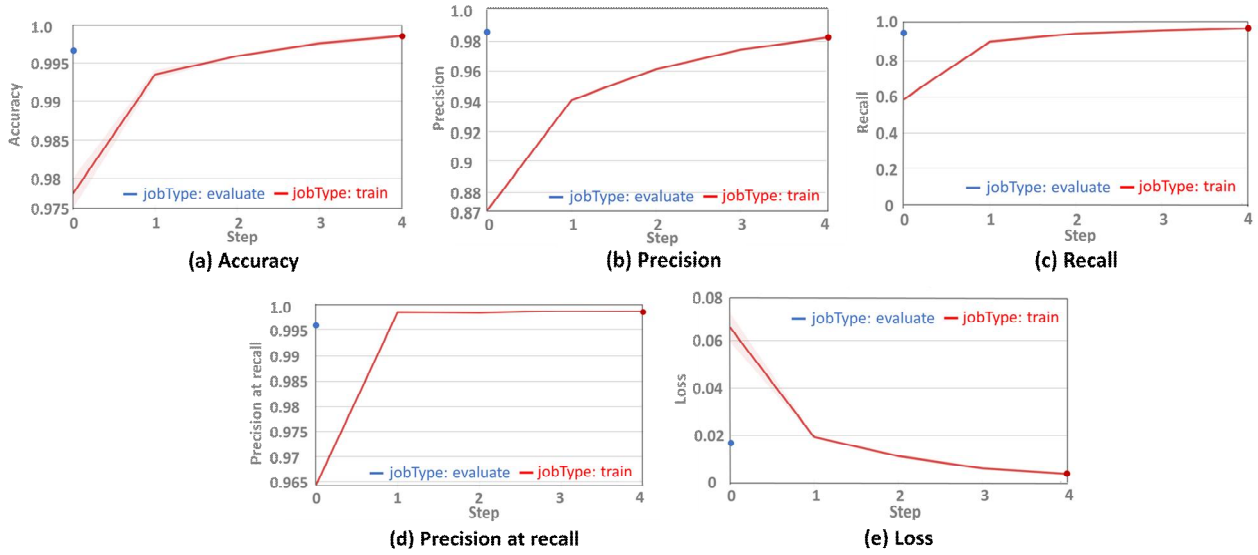

Fig. 2. BERT Learning Result

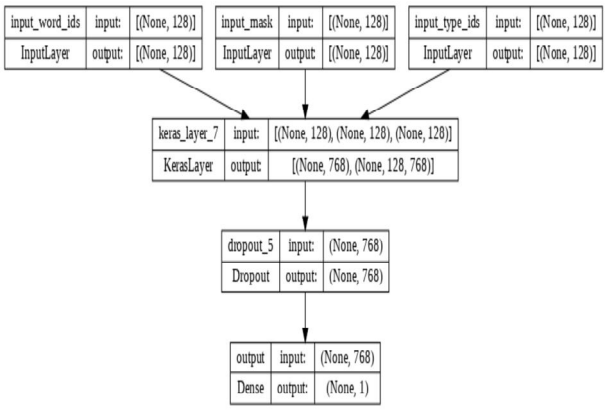

Fig. 3. BERT Model Architecture Summary

Table 10. Number of Learning Data per Dataset

Dataset	Train	Evaluate	Total
Total	22625	7542	30167
Cassandra	4019	1413	5432
MongoDB	3609	1204	4813
Hadoop	4441	1481	5922
OpenStack	6092	2031	8123
Spark	4407	1470	5877

Fig. 2는 Filepath 객체 판별을 위한 BERT 모델 학습 결과이다. 각 그래프의 x축은 step(epochs), y축은 (a)부터 (e)까지 각 Accuracy 값, Precision 값, Recall 값, Precision at recall 값, Loss 값을 나타내며, 모두 0~1 값을 갖는다. Table. 10은 데이터 셋의 개수를 보여준다. 학습 데이터의 경우, 5개의 단어로 이루어진 텍스트 문장 데이터로 총 30167개의 데이터를 사용하였으며, 이 중 22625개의 학습 데이터를 모델 학습에 사용하였으며 7542개의 테스트 데이터를 평가에 사용하였다.

Table 11. Model Learning and Evaluation Result

Dateset	Accuracy	Precision	Recall	Precision at recall	Loss
Total Train	0.9982	0.9822	0.9805	0.9989	0.0046
Total Evaluate	0.9964	0.9843	0.9401	0.9959	0.0167
Cassandra Train	0.9988	0.992	0.9893	1.0	0.004
Cassandra Evaluate	0.9974	1.0	0.9578	1.0	0.0257
MongoDB Train	0.9964	0.9333	0.84	0.9615	0.0108
MongoDB Evaluate	0.9934	0.7692	0.8333	1.0	0.0175
Hadoop Train	0.9973	0.9464	0.9217	1.0	0.0073
Hadoop Evaluate	0.9898	0.7419	0.7931	1.0	0.0256
Openstack Train	0.9996	0.9979	0.9979	1.0	0.0015
Openstack Evaluate	0.9979	0.9758	1.0	1.0	0.0025
Spark Train	0.9911	0.8529	0.7073	1.0	0.0207
Spark Evaluate	0.9916	1.0	0.6	1.0	0.0391

Table. 11은 Filepath 객체 분류에 대한 BERT 모델 학습 및 평가 결과이다. 전체 데이터 셋을 기준으로 accuracy 값, precision 값, recall 값이 모두 0.98로 정확도가 높은 것을 확인할 수 있다. 하지만, MongoDB의 경우 train 결과의 recall 값이 0.84로 비교적 작게 나온 것을 확인할 수 있으며 evaluate 결과의 precision 값, recall 값 모두 0.7692, 0.8333으로 작게 나온 것을 확인

할 수 있다. 또한 Hadoop Evaluate, Spark Train, Spark Evaluate의 결과 역시 비교적 precision 값 혹은 recall 값이 작게 나온 것을 확인할 수 있다.

IV. Conclusions

본 연구에서는 로그 내 여러 객체(Date, Time, IP, URI, Filepath, JSON)를 인식하기 위해 GROK 패턴을 사용하였지만 인식 정확도가 매우 낮은 것을 확인할 수 있었다. 객체 인식 정확도를 높이기 위해 커스텀 GROK 패턴을 활용한 패턴 인식 알고리즘을 적용하였다. 명확한 정의가 없고 OS마다 문법이 다른 Filepath 객체의 경우, 패턴 인식의 한계가 존재하여 주변 단어들을 고려하고자 BERT 기반 객체 인식 기술을 개발하였다. 본 실험에서 사용한 로그 데이터 총 989,626개를 기준으로 Date, Time, IP, URI 객체 관련 기존 GROK 패턴이 가진 예외를 모두 보완하였다. Filepath 객체의 경우, BERT 모델을 활용해 객체 분류를 수행하고 분류된 객체에 대하여 커스텀 GROK 패턴을 활용한 Filepath 패턴 인식 알고리즘을 활용하여 검증 단계를 수행하였다. BERT 모델을 적용한 결과 기존 Grok 패턴의 가장 큰 문제점인 Filepath 객체가 아닌 객체를 Filepath로 인식하는 문제점을 해결하였다.

본 연구는 다양한 대용량 로그 데이터 분석 전 이를 방해하는 객체를 정확히 인식 후 특정 문자열로 변경함으로써 로그 분석을 원활하게 해주는 것에 의의가 있다. 기존 로그 파싱 기법의 연구에 따르면, 전처리는 로그 파싱의 정확도를 향상시킬 수 있다[18, 30]. 최근 로그 파싱 기술 중 하나인 Drain은 로그 분석 전 로그 파싱의 정확성 향상을 위해 IP, block ID 등의 객체를 정규식으로 전처리 과정을 수행한다[18]. 패턴 인식만으로 로그 내 객체 인식에 한계가 있음을 본 실험을 통해 알 수 있었다. 본 연구에서 보여준 로그 내 객체 인식 기술 성능의 개선으로 기존 로그 파싱 기술 내 전처리 단계에 활용되어 로그 파싱 기술 성능 향상에 도움을 줄 것이다. 또한, 객체 인식 기술이 필요한 로그 이상 탐지 기술[14-16, 31-32] 성능 향상에 도움을 줄 것이다.

본 실험에서 MongoDB, Hadoop, Spark 데이터 셋의 경우 Filepath 객체 판별 AI 모델의 recall 값이 accuracy, precision 값보다 작게 나왔다. Filepath가 아닌 객체는 Filepath 객체가 아니라고 정확히 판별하지만 Filepath 객체를 Filepath 객체라고 판별하는 정확도가 비교적 작은 것을 확인할 수 있었다. 이는 모델 학습 시 사용한 데이터 셋이 Filepath 객체로 분류되는 데이터와 Filepath 객체가 아니다로 분류되는 데이터의 비율이 약 20:1이라 recall 값이 비교적 작은 결과가 도출되었다. 본 연구에서는 이를 해결하기 위해 커스텀 GROK 패턴을 사용한 검증 단계를 수행하였다. 하지만, 현재 MongoDB, Hadoop, Spark 데이터 셋 내 Filepath 객체로 분류되는 데이터 개수가 Filepath 객체로 분류되지 않는 데이터 개수보다 약 20배 부족하여 실제 Filepath 객체를 Filepath 객체로 분류하는 정확도가 낮다는 한계가 존재한다. 또한, Date, Time, IP, URI 객체 인식을 위한 패턴 인식 알고리즘의 경우 현재 정의된 커스텀 GROK 패턴 이외의 새로운 패턴의 객체를 갖는 로그 데이터에 대한 예외가 발생할 수 있어 로깅 코드가 수시로 업데이트되고 새로운 오픈소스 플랫폼을 도입하는 환경에서 한계점이 존재한다.

향후 연구 방향으로는 다양한 포맷의 Filepath 객체로 분류되는 데이터를 수집하여 AI 모델만으로 Filepath 객체 인식이 가능하도록 AI 모델의 recall 값과 정확도를 더욱 향상시킬 예정이다. 또한, 본 논문에서 다룬 객체 이외 명확한 정의가 존재하는 XML, Python Dictionary 객체, Python List 객체 등 역시 분석 후 빈틈없는 패턴 인식 알고리즘을 구현할 예정이다. 이를 기반으로 시중에 나와있는 Lognroll[6], Drain[18], IPLoM[20], LKE[22], Spell[24], SHISO[28] 등 로그 파싱 기술에 해당 기술을 적용하여 성능을 직접 비교 분석하는 작업을 할 예정이다.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A5A1021944).

REFERENCES

[1] Debnath, Biplob, et al, "LogLens: A real-time log analysis system," 2018 IEEE 38th international conference on distributed computing systems (ICDCS), pp. 1052-1062, July 2018. DOI: 10.1109/ICDCS.2018.00105

[2] Alspaugh, Sara, et al, "Analyzing log analysis: An empirical study of user log mining," 28th Large Installation System Administration Conference (LISA14), pp. 62-77, Nov 2014.

[3] Kulkarni, Juily, et al, "Analysis of system logs for pattern detection

and anomaly prediction," Proceeding of International Conference on Computational Science and Applications. Springer, Singapore, pp. 427-436, January 2020. DOI: 10.1007/978-981-15-0790-8_42

[4] Chaudhari, Swati, et al, "Real time logs and traffic monitoring, analysis and visualization setup for IT security enhancement," 5th International Conference on Next Generation Computing Technologies (NGCT-2019) 2020, Nov 2019. DOI: 10.2139/ssrn.3527383

[5] J. Zhu et al, "Tools and Benchmarks for Automated Log Parsing," 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 121-130, May 2019. DOI: 10.1109/ICSE-SEIP.2019.00021

[6] Tak, Byungchul, and Wook-Shin Han, "Lognroll: discovering accurate log templates by iterative filtering," Proceedings of the 22nd International Middleware Conference, pp. 273-285, December 2021. DOI: 10.1145/3464298.3493400

[7] S. J. Son and Y. Kwon, "Performance of ELK stack and commercial system in security log analysis," 2017 IEEE 13th Malaysia International Conference on Communications (MICC), Johor Bahru, Malaysia, pp. 187-190, March 2018. DOI: 10.1109/MICC.2017.8311756

[8] M. Bajer, "Building an IoT Data Hub with Elasticsearch, Logstash and Kibana," 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Prague, Czech Republic, pp. 63-68, November 2017. DOI: 10.1109/FiCloudW.2017.101

[9] A. F. Rochim, M. A. Aziz and A. Fauzi, "Design Log Management System of Computer Network Devices Infrastructures Based on ELK Stack," 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), Batam, Indonesia, pp. 338-342, February 2020. DOI: 10.1109/ICECOS47637.2019.8984494

[10] Jin, Zhongjun, Yeye He, and Surajit Chauduri, "Auto-transform: learning-to-transform by patterns," Proceedings of the VLDB Endowment, 13, 12, pp. 2368-2381, September 2020. DOI: 10.14778/3407790.3407831

[11] A. Leadbetter, D. Smyth, R. Fuller, E. O'Grady and A. Shepherd, "Where big data meets linked data: Applying standard data models to environmental data streams," 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, pp. 2929-2937, February 2017. DOI: 10.1109/BigData.2016.7840943

[12] Diotalevi, Tommaso, et al, "Collection and harmonization of system logs and prototypal Analytics services with the Elastic (ELK) suite at the INFN-CNAF computing centre," arXiv preprint arXiv:2106.02612, May 2021. DOI: 10.48550/arXiv.2106.02612

[13] Devlin, Jacob, et al, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, October 2018. DOI: 10.48550/arXiv.1810.04805

[14] Guo, Haixuan, Shuhan Yuan, and Xintao Wu, "Logbert: Log anomaly detection via bert," 2021 international joint conference on neural networks (IJCNN). IEEE, March 2021. DOI: 10.48550/arXiv.2103.04475

[15] Lee, Yukyung, Jina Kim, and Pilsung Kang, "LAnoBERT: System log anomaly detection based on BERT masked language model," arXiv preprint arXiv:2111.09564, November 2021. DOI: 10.48550/arXiv.2111.09564

[16] Chen, Song, and Hai Liao, "BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model," Applied Artificial Intelligence 36.1 (2022): 2145642, Nov 2022. DOI: 10.1080/08839514.2022.2145642

[17] T. Prakash, M. Kakkar and K. Patel, "Geo-identification of web users through logs using ELK stack," 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), pp. 606-610, July 2016. DOI: 10.1109/CONFLUENCE.2016.7508191

[18] 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), pp. 33-40, September 2017. DOI: 10.1109/ICWS.2017.13

[19] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764), pp. 119-126, December 2003. DOI: 10.1109/IPOM.2003.1251233

[20] Makanju, Adetokunbo AO, A. Nur Zincir-Heywood, and Evangelos E. Milios, "Clustering event logs using iterative partitioning," Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1255-1264, June 2009. DOI: 10.1145/1557019.1557154

[21] R. Vaarandi and M. Pihelgas, "LogCluster - A data clustering and pattern mining algorithm for event logs," 2015 11th International Conference on Network and Service Management (CNSM), pp. 1-7, November 2015. DOI: 10.1109/CNSM.2015.7367331

[22] Q. Fu, J. -G. Lou, Y. Wang and J. Li, "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis," 2009 Ninth IEEE International Conference on Data Mining, pp. 149-158, December 2009. DOI: 10.1109/ICDM.2009.60

[23] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand and R. Sasnauskas, "A Search-Based Approach for Accurate Identification of Log Message Formats," 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC), pp. 167-16710, January 2020

[24] M. Du and F. Li, "Spell: Streaming Parsing of System Event

Logs," 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 859-864, February 2017. DOI: 10.1109/ICDM.2016.0103

[25] Shima, Keiichi. "Length matters: Clustering system log messages using length of words," arXiv preprint arXiv:1611.03213, November 2016. DOI: 10.48550/arXiv.1611.03213

[26] H. Hamooni, B. Debnath, J. Xu, H. Zhang, G. Jiang, and A. Mueen, "LogMine: fast pattern recognition for log analytics," in CIKM, pp. 1573-1582, October 2016. DOI: 10.1145/2983323.2983358

[27] Tang, Liang, Tao Li, and Chang-Shing Perng, "LogSig: Generating system events from raw textual logs," Proceedings of the 20th ACM international conference on Information and knowledge management, pp. 785-794, October 2011. DOI: 10.1145/2063576.2063690

[28] Mizutani, Masayoshi, "Incremental mining of system log format," 2013 IEEE International Conference on Services Computing, October 2013. DOI: 10.1109/SCC.2013.73

[29] Jiang, Zhen Ming, et al, "An automated approach for abstracting execution logs to execution events," Journal of Software Maintenance and Evolution: Research and Practice 20.4 (2008): 249-267, July 2008. DOI: 10.1002/smr.374

[30] He, Pinjia, et al, "An evaluation study on log parsing and its use in log mining," 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN), October 2016. DOI: 10.1109/DSN.2016.66

[31] Du, Min, et al, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, October 2017. DOI: 10.1145/3133956.3134015

[32] Xu, Wei, et al, "Detecting large-scale system problems by mining console logs," Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pp. 117-132, October 2009. DOI: 10.1145/1629575.1629587

Authors

Jiho Ju received the B.S. degree in Computer Science and Engineering from Kyungpook National University, Daegu, Korea in 2020. He is interested in cloud computing, and distributed systems.

Byungchul Tak received his B.S. degree from Yonsei University in 2000, M.S. from KAIST in 2003 and Ph.D. degrees in Computer Science and Engineering from the Pennsylvania State University at University Park in 2012. Dr. Tak joined the faculty of the Department of Computer Science at Kyungpook National University, Dague, Korea, in 2017. He is currently an Associate Professor in the Department of Computer Science. His research interests are in cloud computing, distributed systems, operating system and big data analytics.