North American Academic Research

TWASP

Research

Benchmarking Log Parsing Methods: Empirical Study from Real-World Datasets

Tasnaim Ahmad Hosen*

College of Software, Nankai University, Tianjin, China

Abstract: In today's era dominated by software applications and smart systems, our daily activities increasingly rely on various smart applications, including shopping apps, e-commerce, web, and social media applications. As these applications and systems grow, so does the challenge of monitoring and maintaining their 24/7 operation, posing a significant task for online service providers. Logs, key tools for recording system runtime information, are crucial in managing web services. However, as systems and applications become more complex, manual review of log records becomes time-consuming and impractical. The development of automated log analysis tools has recently garnered significant attention from researchers in the academic and industrial sectors. These tools are pivotal in several downstream tasks, such as anomaly detection, failure prediction, and system diagnosis. The primary step in log analysis is log parsing, which involves transforming unstructured log messages into structured data for subsequent mining tasks. To date, over 30 log parsing tools have been developed. This paper focuses on an empirical study of fifteen log parsing tools, chosen for their public availability of source code and proven high accuracy and efficiency in prior research. The study was conducted using seven real datasets collected from servers at the National Agency for Network Services (NANS) at the Ministry of Communications and Technology in Syria, including an Apache web server, Linux Mail server, WHMCS (Web Hosting Billing & Automation Platform), Microsoft web server IIS (Microsoft Internet Information Services 10.0), Plesk (Web Hosting Control Panel), and a Cisco ASA 5512 Firewall device. A comparative analysis of these tools in terms of accuracy, efficiency, and scalability was performed to assist system administrators in selecting the most suitable log-parsing tool for their analysis tasks. The study finds that Drain demonstrates the best performance in these aspects. Our contributions provide a strong link between research and industry log parsing, consolidating past research efforts and facilitating future advancements.

Keywords: Log parsing, system logs, log template extraction, log analysis.

*Corresponding Author

Accepted: 27 February, 2025; **Published**: 28 February, 2025

How to cite this article*: Tasnaim Ahmad Hosen (2025). Benchmarking Log Parsing Methods: Empirical Insights from Diverse Real-World Datasets. North American Academic Research, 8(2), 263-283. doi: https://doi.org/10.5281/zenodo.14977237*

Conflicts of Interest: There are no conflicts to declare.

Publisher's Note: NAAR stays neutral about jurisdictional claims in published maps/image and institutional affiliations.

Introduction

Over the past decade, data centres and computer networks have seen dramatic growth in processing power and size, handling vast amounts of log data daily. In cases of service interruption or failure, system administrators are required

to manually review logs to identify faults, a process that is time-consuming and dependent on their expertise. Given the critical importance of maintaining uninterrupted national data centre services, the implementation of tools for monitoring and analyzing log records is imperative. Any server malfunction can lead to significant damage. A study by the Ponemon Institute, sponsored by Emerson Network Power, highlights the escalating costs of data centre downtime. The study reports that the average cost of an unplanned data centre outage in the US has risen to over US$7,900 per minute, a 41% increase from US$5,600 in 2010, underscoring the economic impact of such outages.

In recent decades, modern software such as search engines, instant messaging apps, and cloud systems, has become increasingly integrated into our daily lives and indispensable. Most of these software systems are expected to be available 24/7, as any significant downtime can lead to substantial revenue loss, especially in large-scale distributed systems (1).

For distributed systems like electronic payment applications, commercial applications, and cloud systems, malfunctions, ranging from server outages to slow responses or incorrect results, can result in user dissatisfaction, loss of confidence, and significant revenue losses. This is particularly critical given the presence of competitors offering similar services with varying quality and availability.

Identifying the source of system malfunctions is challenging due to a range of potential causes, including network errors, physical server malfunctions, software system errors, or, in the worst case, hacking and malicious activities. Log files, which record system run-time information, are the primary data source for mitigating the negative effects of system failures or predicting anomalies.

Logs are semi-structured texts generated by logging statements in software source code, added by developers during system or application development. Systems record log messages for monitoring malfunctions, identifying errors, and system maintenance. However, manually reviewing logs is often a futile and time-consuming task due to several reasons: (1) the size and complexity of modern systems result in an ever-increasing volume of log records, demanding extensive review time; (2) systems developed collaboratively or using third-party tools may generate logs that are inaccessible or unfamiliar to developers and system engineers; and (3) logs, which reflect developer-specific logging instructions, often require domain-specific knowledge, making them difficult for system engineers to interpret. These challenges highlight the need for automated and efficient log parsing solutions to streamline log analysis and reduce manual effort.

Therefore, the development of automated tools for analyzing and extracting useful information from log records has become crucial to ensure the continued availability and quality of services.

Automated log mining tools employ statistical, data analysis, machine learning, and deep learning techniques to automate log analysis tasks. These tools typically require structured data, whereas log records are semi-structured due to their free-text style. Addressing the semi-structured nature of log messages has attracted significant attention from researchers in academic and industrial fields, leading to the development of up to 30 log parsing tools (1) (2). The objective of log parsing is to transform semi-structured log records into structured data, resulting in a set of record templates and parameters.

A log message generally comprises a header and message content. The header usually contains basic information such as timestamps, the generating component, IP addresses, and the message's severity level (info, warning, error, etc.). Thanks to the commonality of header contents, it can be easily distinguished and extracted using regular expressions. The content of the log message is divided into two main parts:

- The fixed part, which is the constant text written by developers within the source code's print statement

remains unchanged in each log repetition.

- The variable part that reflects the system's state during runtime and varies from one message to another.

```
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-305011: Built dynamic UDP
translation from PKI_Public:*.*.*.*/46905 to Outside:*.*.*.*/17481
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-302015: Built outbound UDP
connection 160979231 for Outside:*.*.*.*/53 (*.*.*.*/53) to PKI_Public:*.*.*.*/46905
(*.*.*.*/17481)
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-302020: Built inbound ICMP
connection for faddr *.*.*.*/1482 gaddr *.*.*.*/0 laddr *.*.*.*/0
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-302021: Teardown ICMP
connection for faddr *.*.*.*/1482 gaddr *.*.*.*/0 laddr *.*.*.*/0
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-302020: Built inbound ICMP
connection for faddr *.*.*.*/1482 gaddr *.*.*.*/0 laddr *.*.*.*/0
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-6-302021: Teardown ICMP
connection for faddr *.*.*.*/1482 gaddr *.*.*.*/0 laddr *.*.*.*/0
Oct 05 10:13:14 *.*.*.* :Oct 04 10:25:53 EEST: %ASA-session-4-106023: Deny tcp src
Outside:*.*.*.*/40981 dst Outside:*.*.*.*/8888 by access-group "Outside_access" [0x0,
0x0]
```

Fig. 1: Log messages

Month	Day	Time	TimeZone	Componer	Content	EventTemplate	ParameterList
Oct	4	10:25:53	EEST	%ASA-sess	Built dynamic UDP trans	Built dynamic <*> translation from <	['UDP', 'PKI_Public:*/46905', 'Outside:*/17481']
Oct	4	10:25:53	EEST	%ASA-sess	Built outbound UDP con	Built outbound UDP connection <*>	['160979231', 'Outside:8.8.8.8/53 (8.8.8.8/53)', 'PKI_Public:*/469
Oct	4	10:25:53	EEST	%ASA-sess	Built inbound ICMP conr	Built <*> ICMP connection for faddr	['inbound', '*/1482', '*/0', '*/0']
Oct	4	10:25:53	EEST	%ASA-sess	Teardown ICMP connect	Teardown ICMP connection for fadd	['*/1482', '*/0', '*/0']
Oct	4	10:25:53	EEST	%ASA-sess	Built inbound ICMP conr	Built <*> ICMP connection for faddr	['inbound', '*/1482', '*/0', '*/0']

Fig. 2: Structured log

For instance, in the first log message in Figure 1, the header (e.g., 'Oct 04 10:25:53 EEST %ASA-session-6-305011') can be readily identified through regular expressions. The log message consists of a template 'Built dynamic <*> translation from <*> to <*>' and the parameter list is '['UDP', 'PKI_Public:ip_add/46905', 'Outside:ip_add/17481']' as shown in Figure 2.

Automated Log Analysis

For better comprehension of the log parsing step, it is worth noting that the entire automated log analysis process, as shown in Figure 3, primarily involves the following steps (1):

- **Logging:** It is the first step, where the developers add the print statements to the system or application source code which will generate the log messages later during the system runtime. In the logging step, developers must consider three main questions: where to log, what to log, and how to log.
- **Log Compression:** Log compression is a technique used to reduce the size of log files generated by distributed systems. These log files can be very large, often reaching several gigabytes per day, which poses a challenge for service providers to provide sufficient storage space. There are several approaches to log compression, including bucket-based compression, dictionary-based compression, and statistics-based compressors.
- **Log Parsing:** Converts semi-structured log data to structured data, making it compatible with log analysis tools. Structured data allows for efficient search, filtering, grouping, counting, and mining of logs.
- **Log Mining:** Refers to the process of analyzing large volumes of log data to extract meaningful patterns and trends. This is done through the use of statistical methods, data mining techniques, and machine learning algorithms. By analyzing log data, it is possible to gain valuable insights that can be used to guide and improve the monitoring, administration, and troubleshooting of software systems.

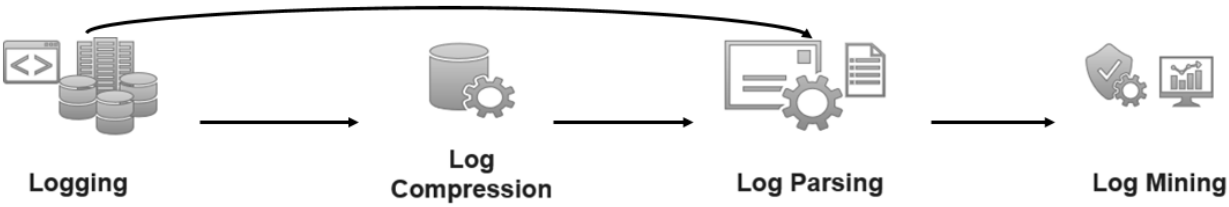

Fig. 3: An overall framework for automated log analysis.

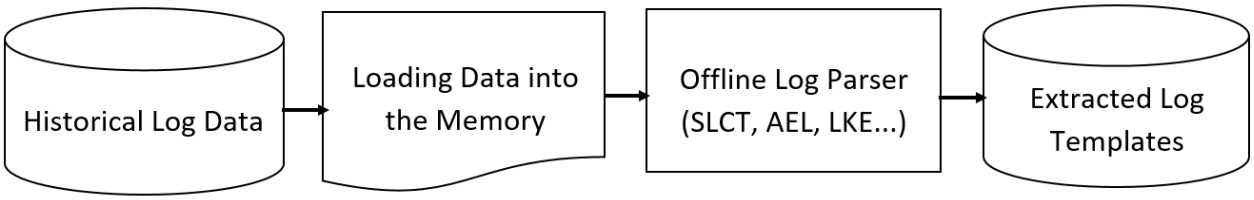

Fig. 4: Offline log parser working flow.

Log Parsing Applications

Usage analysis is crucial during software development and maintenance to understand how users interact with a system, and log parsing plays a key role in enabling this. Anomaly detection has also become an essential part of system monitoring and maintenance tasks, with various techniques being employed. For instance, SwissLog (6) presents a robust and unified anomaly detection model for diverse faults, including sequential log anomalies and performance issues, using deep learning. Similarly, in (7), Tan et al. developed a deep learning (DL)-based log anomaly detection framework for 5G CN, which encompasses log parsing, log grouping, feature extraction, and model training, with each module designed for distinct functionalities to enable combinational usage in various situations. On the other hand, transfer learning has been used for anomaly detection, as seen in (8), where log parsing serves as a necessary data preprocessing step.

Frequently, system issues such as disk errors or network disconnections can occur repeatedly or be reported multiple times by different users, leading to many duplicate issues. Automatically identifying these duplicates is essential to reduce the workload of developers and support engineers (2). Additionally, extracting all possible event templates from logs is a critical step before constructing a performance model, which takes event sequences as inputs (2). Diagnosing failures manually is challenging and time-consuming due to the large, verbose, and unstructured nature of logs, making log parsing a necessary step (2).

Log Parsers Working Mode

Log parsers can be categorized into three modes of operation: offline, online, and hybrid.

- **Offline Mode:** Offline log parsers process the log messages in batches where the historical log data should be loaded into the memory, refer to Figure 4. Early log parsers such as SLCT (16), LKE (17), LogMine (18), and IPLoM (19) operate in this mode. Offline parsing allows log parsers to scan all the messages and parse the logs with a global view. However, offline log parsers are not suitable for real-time log analytics, making them less effective for hyper-scale distributed systems.
- **Online Mode:** Online log parsers operate in real-time on log streams, as shown in Figure 5. Examples of such parsers include Spell (20), Drain (21), and SHISO (22). These parsers are advantageous over offline solutions in two ways.

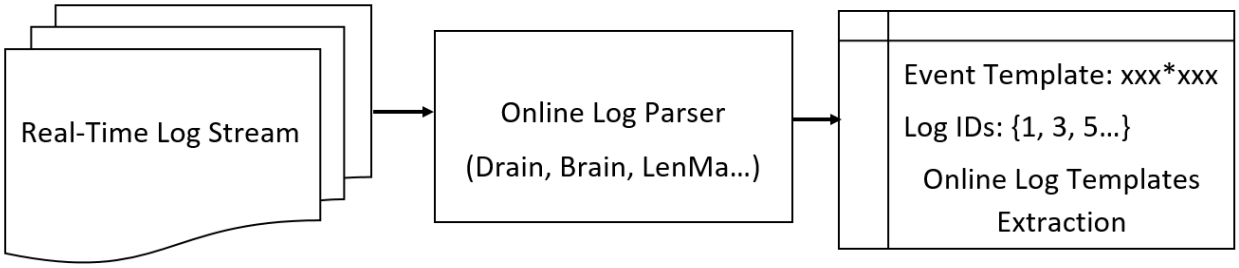

Fig. 5: Online log parser working flow.

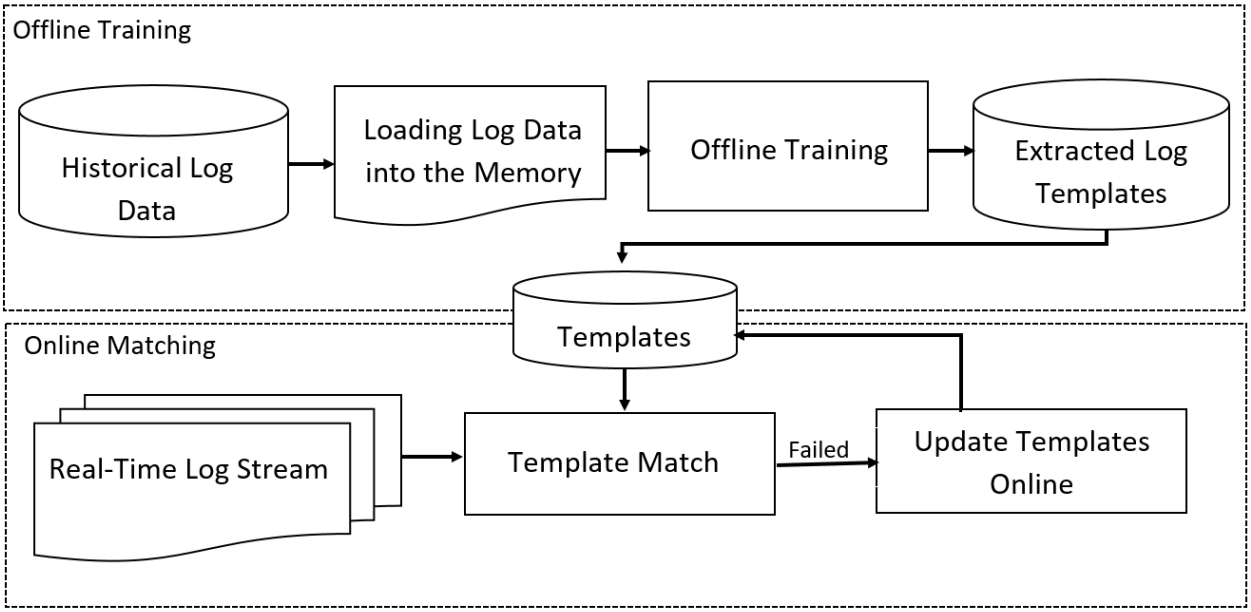

Fig. 6: Hybrid log parser working flow.

Firstly, they can analyze newly collected logs on the fly and incrementally refine their parsing results without requiring offline training. This makes them ideal for real-time tasks like system monitoring and fault diagnosis. Secondly, online parsers do not require loading the entire input data into memory, making them more accessible to users with limited resources.

- **Hybrid Mode:** Hybrid log parsers, such as NuLog (23) and LogParse (24), work in both an offline training phase and an online parsing phase. These parsers are trained offline to populate the model parameters and then parse input logs in an online mode. By combining the advantages of offline and online approaches, hybrid log parsers can better learn log characteristics from different sources. Refer to Figure 6 for a visual representation.

Scope

In this study, we have applied fifteen different log parsers to our dataset for several reasons. Firstly, log formats can vary significantly, and a single parser might not be effective in handling all the variations present in the NANS Agency's log data. Secondly, different parsers might have strengths and weaknesses in terms of accuracy, efficiency, and handling specific log structures. By applying multiple parsers, we aimed to obtain a comprehensive evaluation of parsing effectiveness for the NANS Agency's log data.

We applied these fifteen parsers to seven real datasets sourced from the NANS servers. This work underscores the significance of the log parsing process and how the accuracy of its results impacts the subsequent stages of log analysis (10) (11). The chosen tools are open-source with publicly available codes and were selected for their high accuracy based

on prior evaluations (2) (34). These tools are AEL (29), Brain (35), Drain (21), IPLoM (31), LenMa (28), LFA (25), LKE (17), LogSig (27), LogMine (18), LogCluster (26), LogPPT(33), MoLFI (36), SHISO (22), Spell (20) and USTEP (37).

Motivation

Despite the extensive study of log parsing in recent years, there remains a lack of awareness among users about the distinct advantages and impacts of various log parsers on subsequent log mining tasks. This gap often leads to unnecessary reimplementation or redesign of log parsers, a process that is not only time-consuming but also redundant. To bridge this gap, this paper examines fifteen log parsers and packages them into a toolkit to facilitate their reuse.

Paper Structure

The following is the organization of the paper. In Section 2, we will review the current state-of-the-art log parsing tools. We will highlight the limitations of the existing tools, as well as the characteristics of a good parser. Section 3 will describe the study methodology, the design, and the log parsers that we have selected to apply. Section 4 will report on the experimental setup and evaluation results. Finally, in Section 5, we will conclude the paper.

Literature Review

Log data is essential for system monitoring and management. Log parsing transforms raw log data into structured information, enabling businesses to communicate, govern, and make decisions based on data [38]. Effective log parsing reduces complexity, improves understanding, and provides insights into system behaviour, facilitating better resource allocation and operational management.

Existing Log Parsing Tools

The field of log parsing has seen the development of various tools, each with unique approaches to improving accuracy and efficiency. **LLMParser** (39) is a framework that leverages large language models (LLMs) for log parsing. It employs in-context learning and an adaptive parsing cache to enhance accuracy and efficiency, with three LLMs (ChatGPT, Davinci, and Curie) tested for performance. Another study (40) explores the potential of **ChatGPT for log parsing**, highlighting its promise with appropriate prompts but also noting challenges such as handling log-specific data and designing effective prompts. **LogDiv** (41) utilizes GPT-3 for in-context learning, selecting diverse log samples to generate precise log templates without the need for model tuning. **AdaptParse** (42) approaches log parsing by transforming the template generation problem into a word classification task. It uses semantic patterns and an attention network to distinguish between template words and variable words, improving parsing accuracy. **DA-Parser** (43) introduces a domain-aware head to identify the source domains of logs, effectively transforming multi-domain parsing into a series of single-domain parsing problems. This allows for more effective log parsing across diverse domains. **LogSlaw** (38) is an online log parsing algorithm that clusters logs based on improved Jaccard Similarity, making it suitable for heterogeneous logs with large datasets. **PatCluster** (44) is an offline parsing method that uses frequent words to refine log templates, improving parsing accuracy for complex log structures. **Documentation-based Semantic-Aware Log Parsing** (45) is an innovative approach that leverages software documentation to enhance parsing accuracy, even when source code access is unavailable. **Logram** (46) uses n-gram dictionaries for efficient log parsing, making it suitable for streaming scenarios where logs are continuously generated. **UniParser** (32) transforms log parsing into a token classification problem, using token and context encoders to discern patterns in log messages. **Prefix-Graph** (49) is an online parsing technique that merges prefix trees with probabilistic graphs, offering versatility in handling diverse log formats. **LogParse** (24) supports incremental template learning by classifying words into template or variable categories, enabling real-time log matching and new template learning. Finally, **NuLog** (23) operates in both offline and online modes, using self-supervised learning to extract log templates and embedding vectors for efficient log parsing.

Limitations of Current Log Parsing Tools

Limitations of current log parsing tools have been extensively documented in the literature [41, 33, 50, 35, 32, 38, 9]. These limitations include: (1) reliance on manual features, where parsers depend on human-crafted rules, limiting their

adaptability; (2) suboptimal deep learning approaches, as deep learning models require extensive training and often struggle with unseen log formats; (3) limited robustness, with parsers frequently failing to handle unexpected log structures and lacking semantic understanding; (4) difficulties with variable-length logs, as parsers struggle to process messages of varying lengths, often requiring multiple templates; (5) limited adaptability, where parsers need retraining for new software versions and face challenges with large datasets or evolving log formats; and (6) incomplete evaluation, as many methods lack thorough and comprehensive evaluation, hindering their real-world effectiveness. These issues collectively highlight the need for more advanced and flexible log parsing solutions.

Key Characteristics for Efficiency and Adaptability

Effective log parsers should possess key characteristics to ensure efficiency and adaptability. These include: **no-supervision**, enabling operation without prior knowledge or human input; **heterogeneity**, allowing the handling of logs from diverse applications and systems; **efficiency**, ensuring the ability to process logs faster than their generation rate; and **scalability**, enabling the management of large-scale log data without performance bottlenecks. These properties are essential for developing robust and versatile log parsing tools capable of meeting the demands of modern systems.

The field of log parsing has seen significant advancements, but challenges remain, including the need for automation, industrial adoption, comprehensive evaluation, and standardized datasets. Addressing these issues is crucial for developing robust and widely applicable log parsing solutions.

Research Methodology

Research Design

The study aims to answer three research questions:

- **RQ1:** What are the most effective log parsing tools for NANS, considering their specific logs and needs?
- **RQ2:** How can existing log parsing tools be adapted or improved to better serve the agency's log mining requirements?
- **RQ3:** What are the strengths, weaknesses, and underlying principles of the existing log parsing tools?

To address these questions, fifteen log parsing tools were selected based on the following criteria:

- Open-source and publicly available: The tools are open-source, allowing for cost-effective implementation and future customization.
- Previous evaluations on public datasets: The tools have been evaluated in prior studies, but their performance on NANS's private dataset needs to be assessed.
- High accuracy in previous studies: Most tools have demonstrated high accuracy in prior evaluations, though this study will use a broader range of metrics (e.g., Group Accuracy, Message Level Accuracy, Template Accuracy, F1 Score, Robustness, and Efficiency).
- Python implementation: The tools are implemented in Python, aligning with NANS's existing infrastructure.

Data Collection

Log data was collected from six servers within the National Data Center at NANS, including a Linux mail server, Apache web server, WHMCS, Microsoft IIS, Plesk, and Cisco ASA 5512 Firewall. A central Syslog server was deployed to aggregate logs from these servers, ensuring comprehensive coverage while adhering to privacy policies. The Syslog protocol was used to collect and store log messages, which include timestamps, severity levels, and event details. Figures 7 and 8 illustrate the Syslog protocol and sample configuration entries, respectively. To summarize the available datasets under the study, refer to Table 1.

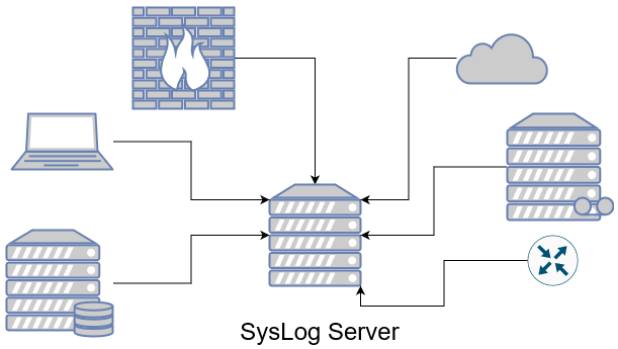

Fig. 7: Syslog Server.

```
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
Oct 05 09:01:34 local Listening for Syslog messages on IP address: *.*.*.*
```

Fig. 8: Sample configuration entries

To evaluate the accuracy of the applied log parsing algorithms, it is required to have the **ground truth dataset**. In the context of log parsing, the ground truth dataset refers to a dataset that is manually annotated or labelled with the correct parsing information, event templates and variables. This dataset serves as a reference or benchmark against which the performance of log parsing algorithms or techniques can be evaluated. It allows us to assess the accuracy, precision, recall, and other performance metrics of the log parsing algorithms. Creating a ground truth dataset is a time-consuming process, as it requires human expertise to annotate the logs accurately. The ground truth datasets for this study have been created manually by the operators in NANS. Each dataset contains 2,000 manually labelled log messages. This is one of the good points of this study, as all of the applied tools were applied and evaluated on publicly available datasets provided by Loghub with their ground truth datasets.

Data Preprocessing

The collected log data required preprocessing to match the input format expected by the log parsing tools. Python was used to clean the datasets, removing unwanted symbols and aligning them with the required log formats. Table 2 summarizes the log formats for each dataset. Preprocessing ensured that the datasets were ready for parsing, despite their initial heterogeneity.

Log Parsers Overview

At a high level, log parsing algorithms can be divided into two main categories **syntax-based** and **semantically-based**. Syntax-based algorithms rely on the structure and format of log messages and involve defining a set of rules or patterns that describe the expected syntax of log entries. Semantically-based log parsing, on the other hand, focuses on understanding the meaning or semantics of log entries. This approach aims to extract higher-level information about the events and activities described in the logs. Fifteen log parsing algorithms were selected for this study, categorized into syntax-based and semantic-based approaches.

Table.1: Log Dataset Information.

Dataset	Description	Data Size	#Messages	2k # Templates
Apache	Apache server log	77 MB	338,166	26
Mail Server	Linux mail server info log	21.5 MB	179,703	236
Mail Server EW	Linux mail server error and warning log	600 KB	4,210	16
Firewall	Cisco ASA 5512 firewall log	31 MB	162,730	15
WHMCS	Web hosting billing and automation platform	11 MB	34,913	67
Plesk	Web hosting control panel	2.488 MB	8,345	71
IIS	Internet Information Services (Microsoft)	905 KB	5,263	21

Table.2: Datasets Log Format.

Dataset	Log Format
Apache	<RemoteHost> <RemoteUser> <AuthUser> <Datetime> <Timezone> <Content>
Mail Server	<Month> <Day> <Time> <Host> <Service> <QueueID> <Content>
Mail Server EW	<Month> <Day> <Time> <Host> <Service> <SeverityLevel> <Content>
Firewall	<Month> <Day> <Time> <TimeZone> <Component> <Content>
IIS	<Date> <Time> <S_IP> <Content>
Plesk	<IP> <User> <Date> <Time> <Content>
WHMCS	<IP> <User> <Date> <Time> <Content>

Syntax-Based:

USTEP (37) uses an evolving tree structure to handle dynamic log messages. It adapts to new log formats by updating its tree structure, making it suitable for environments with evolving log data. **MoLFI** (36) employs a multi-objective approach to identify log message formats, ensuring the high frequency and specificity of the generated templates. **Drain** (21) is an online log parsing method that uses a fixed-depth tree structure. It preprocesses log messages, searches for log groups based on message length and token similarity, and updates the parse tree dynamically. **Spell** (20) is designed for real-time parsing of streaming log data, leveraging the Longest Common Subsequence (LCS) algorithm to identify patterns. **LogMine** (18) handles heterogeneous log messages by generating a hierarchical structure of patterns, minimizing computational overhead. **LenMa** (28) groups log messages based on word length similarity using hierarchical clustering. **LogCluster** (51) clusters logs to identify problems, using a knowledge base to reduce redundant effort. **SHISO** (22) incrementally mines log formats by constructing a hierarchical tree structure, reducing the need for manual analysis. **LogSig** (27) generates system events from textual logs by categorizing messages into event types using message signatures. **LFA** (25) separates static and variable parts of log lines using clustering algorithms, identifying recurring event types. **IPLoM** (31) iteratively partitions log messages to identify line formats, improving clustering accuracy. **LKE** (17) combines clustering algorithms and heuristic rules to generate log templates, refining clusters for

granular insights. **AEL** (29) abstracts log lines into execution events by anonymizing dynamic parts, tokenizing log lines, and categorizing them into groups.

Semantic-Based:

LogPPT (33) transforms log parsing into a parameter recognition problem, using a pre-trained language model to predict parameter positions in log messages. It employs few-shot learning, requiring only a small set of labelled log messages for training, and leverages semantic information for more accurate parsing.

Evaluation

We evaluate fifteen log parsers in six key areas crucial for real-world applications to answer the research questions, the experimental setup and the algorithms' parameters will be covered, and the choice of the accuracy metrics will be explained in detail. The evaluation criteria used are message level accuracy, robustness, efficiency, group accuracy, edit distance, and template accuracy. By analyzing these six key areas, this study provides a comprehensive insight into the strengths and weaknesses of different log parsers, aiding in selecting the most suitable tool that best aligns with NANS-specific logs and needs. Furthermore, this evaluation can reveal potential weaknesses in specific parsers that could be addressed through adaptation or improvement. For example, if a parser exhibits low edit distance but poor group accuracy, it might struggle with nuanced variations within log groups. Identifying such weaknesses can inform targeted improvements, such as incorporating domain-specific knowledge or refining template-matching algorithms.

Experimental Setup

Working Environment

The experiments were conducted on a Windows PC with an 11th Gen Intel Core i5 processor and 16GB of RAM. Python 3.8 and PyCharm IDE were used for scripting. Fifteen log parsers were evaluated, with thirteen sourced from Zhu et al. and two additional open-source parsers, USTEP and LogPPT.

Parameter Tuning

Each algorithm has input parameters, and experiments were run multiple times to determine the best values. Parameters include preprocessing, pattern matching, clustering, and output parameters.

Evaluation Metrics

Several metrics are used to assess log parsing effectiveness:

Group Accuracy (GA): Measures the proportion of correctly grouped log messages. A log message is considered "correctly parsed" only if it is grouped with other log messages in a way that is consistent with the ground truth.

Parsing Accuracy (PA): Measures the percentage of correctly parsed log messages. To be considered "correctly parsed," a log message must have all its static text and dynamic variables (fixed and variable parts) correctly identified. This definition is based on a study by Khan et al. (34). Another study by Liu et al. (32) defines PA as Message Level Accuracy (MLA). For a log message to be "correctly parsed" according to MLA, every token of the log message must be correctly identified as a constant or a variable part.as shown in (equation 1):

$$PA = \text{count(correctly parsed messages)} / \text{count(the total number of log messages)} \quad (1)$$

Edit Distance (ED): Measures the similarity between parsed templates and ground truth templates using the Levenshtein distance algorithm.

Precision, Recall, and F1-score: Precision measures true positives among positive predictions (equation 2), recall measures true positives among actual positives (equation 3), and F1-score balances precision and recall (equation 4):

$$\text{Precision} = TP / TP + FP \quad (2)$$

$$\text{Recall} = TP / TP + FN \quad (3)$$

F1 = 2 * Precision * Recall / Precision + Recall (4)

Time Cost: Measures the time taken to parse log datasets.

Robustness: Evaluates the tool's ability to handle diverse log formats, adapt to changing patterns, and scale efficiently.

The Chosen Evaluation Metrics

Chosen Metrics Breakdown:

- Group Accuracy (GA): Focuses solely on how well messages are grouped based on templates, ignoring variable content. Ideal when variable data is unused (e.g., anomaly detection based on event sequences).
- Message Level Accuracy (MLA): Considers both grouping and individual message parsing accuracy. Suitable when variable data matters but message frequency doesn't significantly affect importance (e.g., parameter value analysis).
- Edit Distance (ED): When analyzing individual messages parsing accuracy and capturing nuances in variable data is crucial, ED provides a fine-grained measure of similarity beyond simply "correct" or "incorrect."
- Template Accuracy (TA): Evaluate how well templates themselves are identified, independent of message frequency. TA is used when variable data matters and message importance isn't frequency-based (e.g., analyzing specific log events regardless of their repetition).

GA, MLA, and TA are linked, with perfect scores in TA or MLA implying a perfect GA score. However, the reverse isn't true, incorrect templates can achieve the same grouping as correct ones.

Log Parsing as a Classification Problem

We can consider the log parsing problem as a multi-classification problem; in this part, we will explain a new approach to deal with the log parsing problem as a multi-classification problem to find the F1 Score of the messages parsing results. As long as the ground truth dataset has a limited number of unique event templates EV = {E1, E2, ...EN}, in this case, these templates will be our classes. For this purpose, we have three important concepts mainly TP, FP, and FN for each class, where:

- TP: the current content belongs to E1 in the ground truth dataset and the parsed result dataset→TPE1
- FP and FN: the current content belongs to E1 in the ground truth dataset but is incorrectly identified to belong to E2 in the parsed result →FNE1 and FPE2
- FN: the current content belongs to E1 in the ground truth dataset but is incorrectly identified to belong to a newly defined template in the parsed result which does not exist in the ground truth→FNE1

Now we can calculate Precision and Recall for each class in EV = {E1, E2, ...EN} using equations 5 and 6 respectively:

Precision for class (Ei): Precision Ei = TPEi / TPEi + FPEi (5)

Recall for class (Ei): R Ei = TPEi / TPEi + FNEi (6)

Overall Precision and Recall

To find the overall Precision and Recall of the classes we will take into account the number of instances of each class in the ground truth, number of occurrences, in other words, we will calculate the weighted average of class precision and recall using (equations 7 and 8) respectively:

$$\text{Weighted Avg.Precision} = \sum (\text{Precision}_i * \text{Instances}_i) / \sum (\text{Instances}_i) \quad \text{for } i = 0\ldots..n \tag{7}$$

$$\text{Weighted Avg.Recall} = \sum (\text{Recall}_i * \text{Instances}_i) / \sum (\text{Instances}_i) \quad \text{for } i = 0\ldots\ldots\ldots n \tag{8}$$

Now we can compute the F1-score of the parsed messages using (equation 9):

F1 = 2 * Weighted Avg.Precision * Weighted Avg.Recall / Weighted Avg.Precision + Weighted Avg.Recall (9)

Experimental Results

Accuracy of Log Parsers

Table 3 shows the message level accuracy (MLA) of each parser across datasets. Drain achieved the highest average MLA (0.78), followed by AEL (0.66) and LogMine (0.46). Conversely, MoLFI, LKE, LogSig, and Spell had the lowest MLA.

Table.3: Message level accuracy on different datasets.

Dataset	AEL	Brain	Drain	LenMa	LFA	LKE	LogPPT	LogCluster	LogMine	LogSig	MoLFI	SHISO	Spell	USTEP
Apache	0.55	0.01	**0.9**	0.12	0.02	0.01	0	0.01	0.49	0.01	0	0.53	0.01	0.57
ECCMail	0.49	0.5	**0.8**	0.49	0.45	0.01	0.17	0.25	0.36	0.05	0.13	0.24	0.15	0.56
ECCMailEW	0.88	0.82	**1**	0.4	0.37	0.25	0.39	0.29	0.77	0.29	0.02	0.7	0.27	0.88
Firewall	0.85	0.66	**1**	0.49	0.15	0	0	0	0.78	0	0	0.42	0	0.49
IIS	0.01	0	**0.01**	0.002	0.002	0.001	0.001	0.001	0	0	0	0.001	0	0.09
Plesk	**0.96**	0.57	**0.96**	0.9	0.12	0.02	-	0.12	0.29	0	0	0.74	0	0.43
Whmcs	0.86	0.35	**0.9**	0.89	0	0	0.09	0.31	0.5	0	0	0.55	0	0.86
Average	0.66	0.42	**0.78**	0.47	0.16	0.04	0.09	0.14	0.46	0.05	0.02	0.45	0.06	0.55

Table 4 shows the template accuracy (TA). Drain again performed best, with an average TA of 0.82. LenMa achieved the highest TA in the two datasets, but Drain outperformed the overall.

Table.4: Template accuracy on different datasets.

Dataset	AEL	Brain	Drain	LenMa	LFA	LKE	LogPPT	LogCluster	LogMine	LogSig	MoLFI	SHISO	Spell	USTEP
Apache	0.68	0.02	**0.85**	0.41	0.05	0.09	0	0.003	0.52	0.07	0	0.53	0.06	0.76
ECCMail	0.92	0.37	**0.94**	0.54	0.89	0.04	0.02	0.29	0.68	0.09	0.03	0.36	0.01	0.91
ECCMailEW	0.94	0.81	**1**	0.03	0.41	0.67	0.01	0.06	0.8	0.64	0.09	0.87	0.04	0.94
Firewall	0.67	0.23	**1**	0.5	0.14	0	0	0	0.5	0	0	0.36	0	0.01
IIS	0.16	0	0.24	**0.27**	0.03	0.01	0.001	0.003	0.17	0	0	0.25	0	0.06
Plesk	0.89	0.05	0.9	**0.93**	0.09	0.13	-	0	0.7	0	0	0.75	0	0.92
Whmcs	0.52	0.03	0.84	0.56	0	0	0.02	0.01	0.41	0	0	0.4	0	0.56
Average	0.68	0.22	**0.82**	0.46	0.23	0.13	0.01	0.07	0.54	0.8	0.02	0.5	0.02	0.59

Table 5 shows the group accuracy (GA). Drain had the highest average GA (0.79), with AEL and LenMa also performing well

Table.5: Group accuracy on different datasets.

Dataset	AEL	Brain	Drain	LenMa	LFA	LKE	LogPPT	LogCluster	LogMine	LogSig	MoLFI	SHISO	Spell	USTEP
Apache	0.61	0.03	**0.75**	0.27	0.05	0.001	0	0.002	0.5	0	0	0.55	0.001	0.73
ECCMail	0.57	0.55	**0.95**	0.48	0.46	0.001	0.02	0.23	0.47	0	0.08	0.25	0.09	0.57
ECCMailEW	**1**	0.99	**1**	0.28	0.27	0.01	0.001	0.02	0.28	0.01	0.003	0.99	0.19	0.99
Firewall	0.86	0.67	**1**	0.48	0	0	0	0	0.8	0	0	0.43	0	0.48
IIS	0.005	0	0.005	**0.01**	0	0.001	0.002	0.002	0.001	0	0	0.001	0	0.02
Plesk	**0.97**	0.7	**0.97**	**0.96**	0	0.01	-	0.14	0.23	0	0	0.79	0	0.29
Whmcs	0.83	0.22	**0.88**	0.83	0	0	0.04	0.22	0.45	0	0	0.45	0	0.22
Average	0.69	0.45	**0.79**	0.47	0.11	0.003	0.01	0.09	0.39	0.001	0.01	0.49	0.04	0.47

Table 6 shows the edit distance (ED). Drain had the lowest average ED (7), indicating better alignment with ground truth messages.

Table.6: Edit distance on different datasets..

Dataset	AEL	Brain	Drain	LenMa	LFA	LKE	LogPPT	LogCluster	LogMine	LogSig	MoLFI	SHISO	Spell	USTEP
Apache	20	18	**2**	37	66	78	48	76	26	58	66	20	37	20
ECCMail	11	15	**4**	12	18	24	17	29	14	26	24	25	18	9
ECCMailEW	3	3	**0**	31	26	25	8	12	4	12	12	22	10	3
Firewall	1	2	**0**	5	14	21	16	66	3	18	43	7	20	35
IIS	39	43	39	37	48	44	14	71	28	54	42	**26**	32	45
Plesk	**1**	13	**1**	2	110	132	-	118	22	131	156	7	36	18
Whmcs	5	20	**3**	5	74	97	12	111	17	51	176	15	83	5
Average	11	16	**7**	18	51	60	16	69	16	50	75	17	34	19

Robustness of Log Parsers

Figures 9, 10, and 11 show the distribution of MLA, TA, and GA across datasets. Drain consistently achieved the highest median accuracy. Figure 12 shows the F1-score distribution, with Drain achieving the highest F1-score (around 1.0). Figures 13 to 16 show MLA across different log volumes. Drain maintained stable accuracy even as log volume increased. In the same way, we assess the performance of log parsers across different log volumes in terms of Template Accuracy, in summary, Drain outperforms the other log parsers in 4 out of 4 datasets included in these figures.

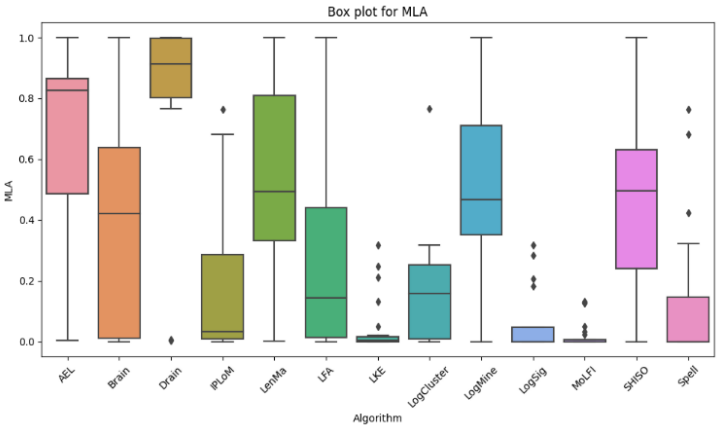

Fig.9: Message level accuracy distribution of log parsers across different datasets

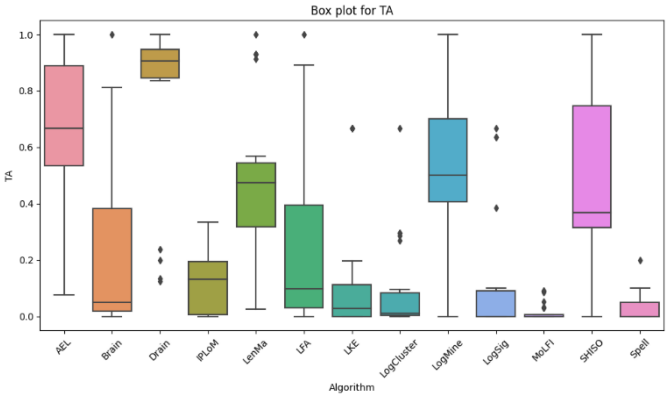

Fig.10: Template accuracy distribution of log parsers across different datasets

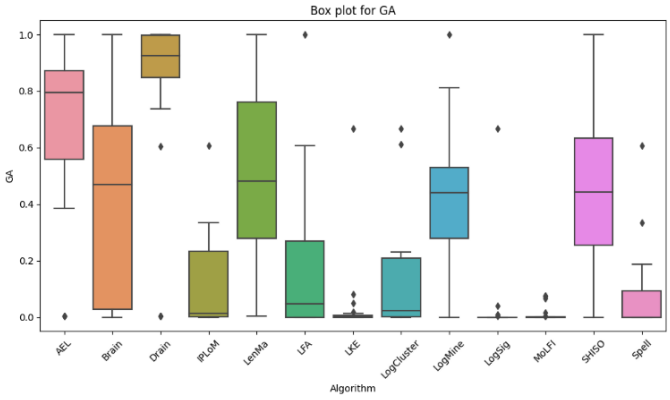

Fig.11: Group accuracy distribution of log parsers across different datasets

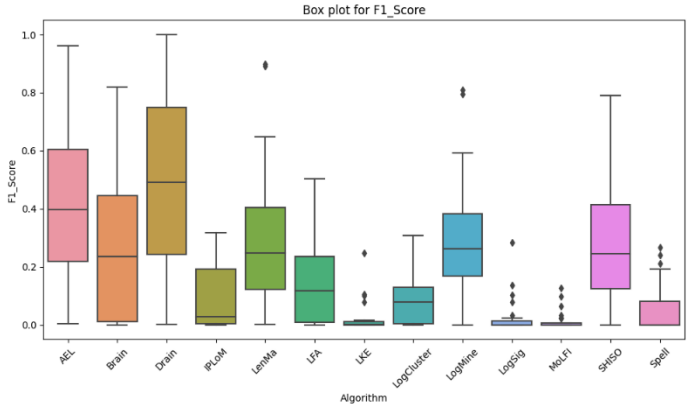

Fig.12: F1 score distribution of log parsers across different datasets

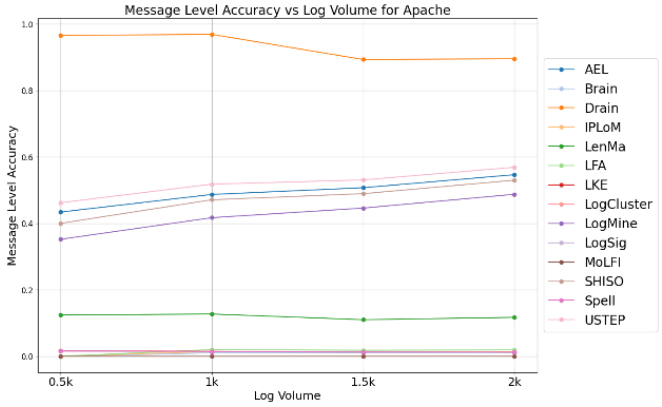

Fig.13: Apache message level accuracy with different log volumes

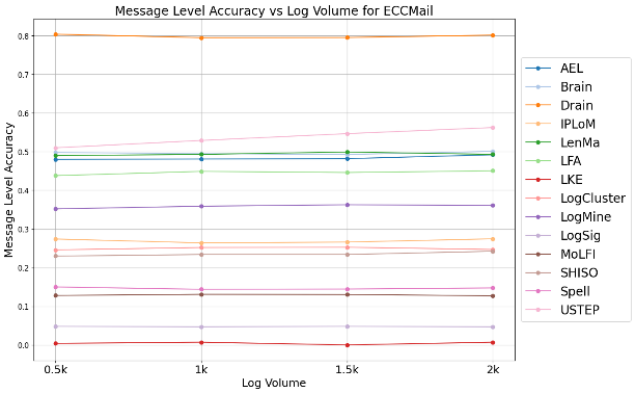

Fig.14: ECCMail message level accuracy with different log volumes

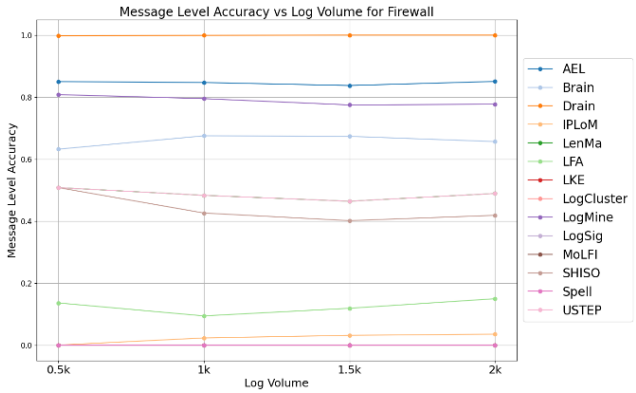

Fig.15: Firewall message level accuracy with different log volumes

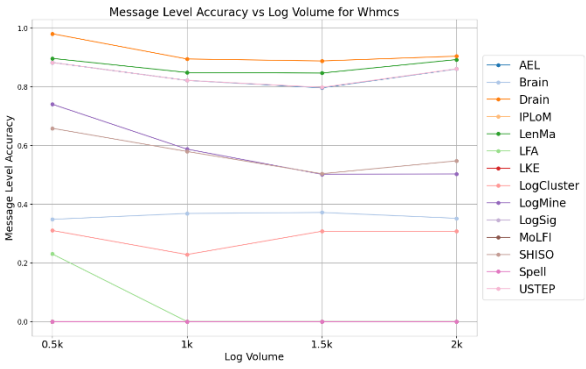

Fig.16: Whmcs message level accuracy with different log volumes

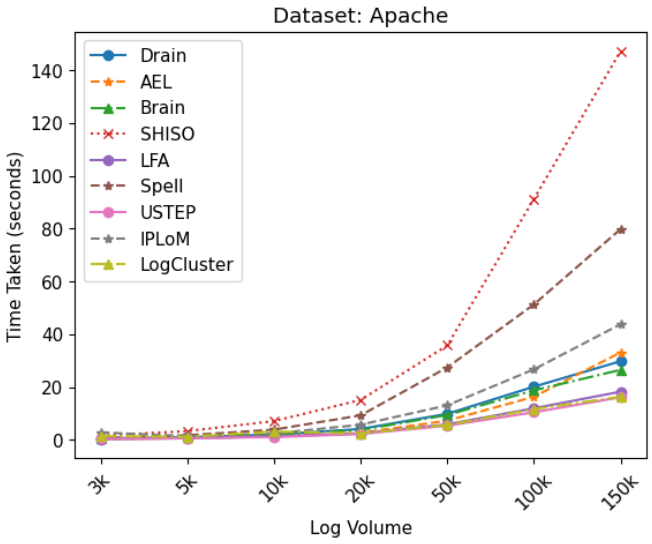

Fig.17: Execution time of log parsers on different Apache log volumes.

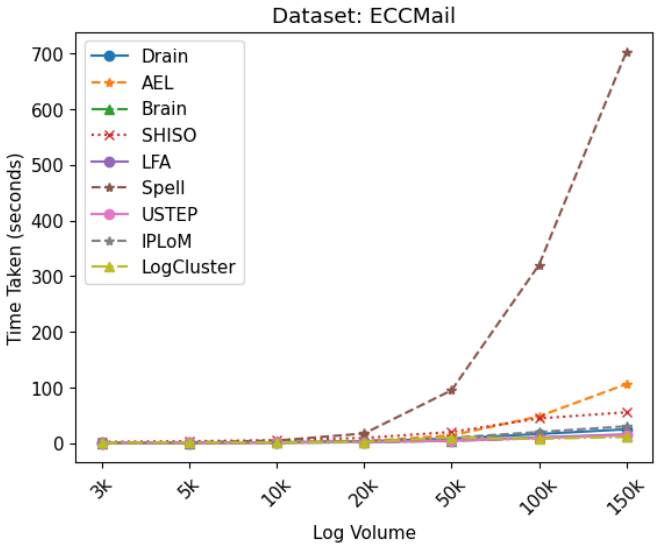

Fig.18: Execution time of log parsers on different ECCMail log volumes.

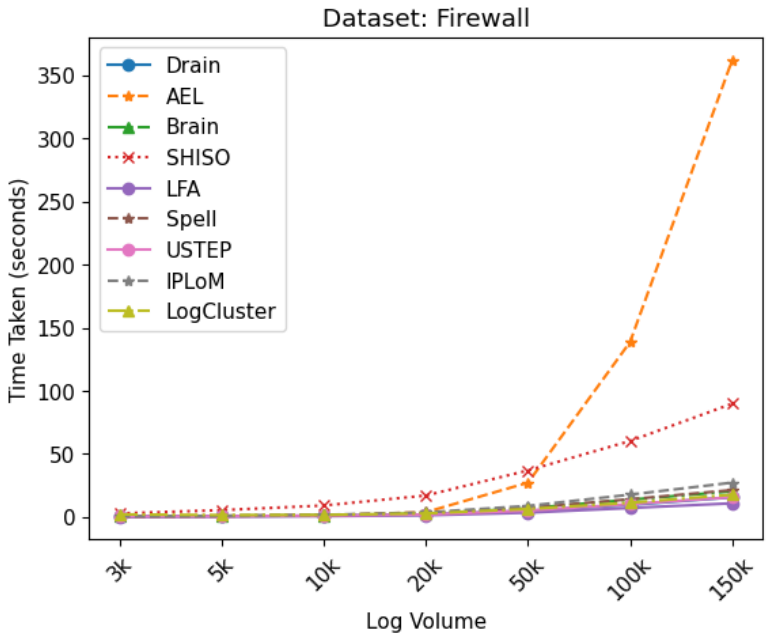

Fig.19: Execution time of log parsers on different Firewall log volumes.

Efficiency of Log Parsers

Figures 17, 18, and 19 show the execution time of parsers across different log volumes. Drain recorded the lowest execution time and minimal changes in processing time, even with large log sizes.

Findings and Results

RQ1: Selecting the Most Effective Log Parsing Tool for NANS

Drain is the most effective log parser for NANS, with high accuracy, robustness, and efficiency. However, it performed poorly on the IIS dataset, indicating room for improvement.

RQ2: Adapting Existing Log Parsing Tools

To better serve NANS, log parsers need to support **Arabic** language parsing and improve the handling of log messages starting with digits. Integrating Natural Language Processing (NLP) techniques could enhance template identification.

RQ3: Strengths, Weaknesses, and Underlying Principles

- Strengths: Automation, efficiency, accuracy, scalability, and flexibility.
- Weaknesses: Domain-specific limitations, false positives/negatives, complexity, and limited user interfaces.
- Underlying Principles: Regular expressions, lexical analysis, and machine learning (ML).

Conclusion

The study distinguishes itself from previous research in several key ways. First, it employs a comprehensive set of six evaluation metrics—message-level accuracy, template accuracy, group accuracy, edit distance, F1 score, and execution time—ensuring a thorough and nuanced assessment of log parsing tools. This provides NANS administrators with a more informed basis for decision-making. Second, the use of real-world datasets sourced directly from NANS operations ensures that the evaluated tools are tested on data reflecting actual challenges faced by administrators, enhancing the relevance and generalizability of the findings. These datasets differ significantly from those used in prior research, contributing new insights to the field. Third, this study represents the first exploration of log parsing within NANS, paving the way for future research and potential improvements in log mining tasks and system efficiency. Finally, by identifying the most accurate tool, the study streamlines tool selection for NANS administrators, reducing their initial scope from fifteen tools to one and allowing them to focus their efforts on learning, utilizing, and improving this tool for broader applicability.

Limitations

This empirical study acknowledges several limitations that must be considered when interpreting and applying the results to real-world scenarios. First, the limited dataset diversity, with only seven datasets used due to access restrictions, may restrict the generalizability of findings across all log types and environments, despite representing various formats and sources. Second, the reliance on ground truth datasets, which were manually parsed and limited to 2,000 records per dataset, poses a constraint, as larger datasets are known to enhance evaluation accuracy, potentially affecting the results' applicability to real-world, larger-scale deployments. Third, the manual generation of ground truth datasets introduces the possibility of unintended bias, which could influence evaluation outcomes despite efforts to mitigate it. Additionally, the study focused on event information extraction without directly addressing the effectiveness of parsed logs for downstream tasks like anomaly detection and failure prediction, which may depend on other factors. Despite these limitations, the study provides valuable insights into the capabilities of the evaluated log-parsing tools. By acknowledging these constraints and considering them in decision-making, administrators can better select tools suited to their specific needs and log types. The study also underscores the importance of addressing data diversity, ground truth generation methods, and downstream task requirements in future log-parsing research.

Future Work

This study establishes a foundation for advancing log parsing capabilities within NANS, identifying several promising directions for future work. First, expanding ground truth datasets by increasing their size and incorporating diverse log formats, sources, and complexities will enhance evaluation accuracy and generalizability, while automating ground truth generation can reduce manual effort. Second, developing a log parsing toolkit that integrates top-performing parsers, offers a user-friendly interface, and supports multiple export formats will provide NANS administrators with customizable, efficient tools. Third, enabling real-time parsing and monitoring through streaming integration, a graphical interface for visualization, and an alert system for anomaly detection will ensure proactive service availability. Additionally, exploring domain-specific parsers, advanced techniques like deep learning and natural language processing, and benchmarking emerging tools will further refine parsing accuracy and flexibility. By pursuing these avenues, NANS can significantly improve log analysis, anomaly detection, and overall service security and efficiency.

Author Contributions: At first page.

Approval: All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable

Acknowledgments: Not Mentioned.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," ACM Computing Survey, 2020. Submitted on 15 Sep 2020 (v1), last revised 1 Jun 2021 (this version, v2).

[2] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSESEIP), (Shenzhen, China; Hong Kong; Guangzhou, China; Zurich, Switzerland; Chengdu, China), IEEE, 2019.

[3] G. Lee, J. J. Lin, C. Liu, A. Lorek, and D. V. Ryaboy, "The unified logging infrastructure for data analytics at twitter," PVLDB, vol. 5, no. 12, pp. 1771–1780, 2012.

[4] "Overview of logs-based metrics." https://cloud.google.com/logging/docs/logs-based-metrics.

[5] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using magpie for request extraction and workload modelling," in OSDI, pp. 259–272, 2004.

[6] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults," in 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), 2020.

[7] Y. Tan, J. Wang, J. Liu, and Y. Li, "Deep learning-based log anomaly detection for 5g core network," 2023 IEEE/CIC International Conference on Communications in China (ICCC), pp. 979–8–3503–4538–4/23, 2023. ISBN: 979-8-3503-4538-4/23.

[8] R. Chen, S. Zhang, D. Li, Y. Zhang, F. Guo,W. Meng, D. Pei, Y. Zhang, X. Chen, and Y. Liu, "Logtransfer: Cross system log anomaly detection for software systems with transfer learning," in 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), (Coimbra, Portugal), IEEE, 2020.

[9] X. Xie, Z.Wang, X. Xiao, et al., "A confidence-guided evaluation for log parsers inner quality," Mobile Networks and Applications, vol. 26, pp. 1638–1649, August 2021.

[10] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), (Hong Kong), IEEE, 2016.

[11] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), (Melbourne, Australia), IEEE, IEEE, 2021. Published in: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE).

[12] D. Shin, Z. A. Khan, D. Bianculli, and L. Briand, "A theoretical framework for understanding the relationship between log parsing and anomaly detection," in RV 2021 (L. Feng and D. Fisman, eds.), vol. 12974 of LNCS, pp. 277–287, Springer Nature Switzerland AG, 2021.

[13] Z. A. Khan, D. Shin, D. Bianculli, and L. Briand, "Impact of log parsing on log-based anomaly detection," arXiv preprint arXiv:2305.15897, 2023.

[14] Y. Fu, M. Yan, Z. Xu, X. Xia, X. Zhang, and D. Yang, "An empirical study of the impact of log parsers on the performance of log-based anomaly detection," Empirical Software Engineering, vol. 28, no. 6, 2023. l' The Author(s), under exclusive license to Springer Science+Business Media, LLC, part of Springer Nature 2022.

[15] X.Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," Empirical Software Engineering, vol. 28, no. 137, 2023. l' The Author(s), under exclusive license to Springer Science+Business Media, LLC, part of Springer Nature 2023.

[16] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764), (Kansas City, MO, USA), IEEE, IEEE, October 2003.

[17] Q. Fu, J.-G. Lou, Y.Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in 2009 Ninth IEEE International Conference on Data Mining, IEEE, IEEE, 2009.

[18] H. Hamooni, B. Debnath, J. Xu, H. Zhang, G. Jiang, and A. Mueen, "Logmine: Fast pattern recognition for log analytics," in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, (Indianapolis, IN, USA), ACM, ACM, October 2016. c 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

[19] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," IEEE Transactions on Knowledge and Data Engineering, vol. 24, pp. 1921–1936, November 2012.

[20] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in 2016 IEEE 16th International Conference on Data Mining (ICDM), (Barcelona, Spain), IEEE, 2016.

[21] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in 2017 IEEE 24th International Conference on Web Services, (Honolulu, HI, USA), IEEE, 2017.

[22] M. Mizutani, "Incremental mining of system log format," in 2013 IEEE International Conference on Services Computing, (Santa Clara, CA, USA), IEEE, IEEE, June 2013. Published in: 2013 IEEE International Conference on Services Computing.

[23] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-supervised log parsing," arXiv preprint arXiv:2003.07905, 2020.

[24] W. Meng, Y. Liu, F. Zaiter, S. Zhang, Y. Chen, Y. Zhang, Y. Zhu, E. Wang, R. Zhang, S. Tao, D. Yang, R. Zhou, and D. Pei, "Logparse: Making log parsing adaptive through word classification," in 2020 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020.

[25] M. Nagappan and M. Vouk, "Abstracting log lines to log event types for mining software system logs," in 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pp. 114–117, IEEE, 2010.

[26] R. Vaarandi and M. Pihelgas, "Logcluster: A data clustering and pattern mining algorithm for event logs," in 2015 11th International Conference on Network and Service Management (CNSM), pp. 1–7, IEEE, 2015.

[27] L. Tang, T. Li, and C.-S. Perng, "Logsig: Generating system events from raw textual logs," in Proceedings of the 20th ACM International Conference on Information and Knowledge Management, p. 785–794, ACM, 2011.

[28] K. Shima, "Length matters: Clustering system log messages using length of words," arXiv preprint arXiv:1611.03213, Nov 2016.

[29] Z. M. Jiang, A. E. Hassan, P. Flora, and G. Hamann, "Abstracting execution logs to execution events for enterprise applications," in The Eighth International Conference on Quality Software (QSIC), (Canada), IEEE, 2008.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[31] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in KDD'09, June 28–July 1, 2009, Paris, France, (New York, NY, USA), ACM, 2009.

[32] Y. Liu, X. Zhang, S. He, H. Zhang, L. Li, Y. Kang, Y. Xu, M. Ma, Q. Lin, Y. Dang, S. Rajmohan, and D. Zhang, "Uniparser: A unified log parser for heterogeneous log data," arXiv preprint arXiv:2202.06569, 2022.

[33] V.-H. Le and H. Zhang, "Log parsing with prompt-based few-shot learning," in Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), IEEE, IEEE, May 2023. Published in the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia. Date Added to IEEE Xplore: 14 July 2023.

[34] Z. A. Khan, D. Shin, D. Bianculli, and L. Briand, "Guidelines for assessing the accuracy of log message template identification techniques," in 44th International Conference on Software Engineering (ICSE '22), (Pittsburgh, PA, USA), p. 12, ACM, May 2022. This work is licensed under a Creative Commons Attribution International 4.0 License.

[35] S. Yu, P. He, N. Chen, and Y. Wu, "Brain: Log parsing with bidirectional parallel tree," IEEE Transactions on Services Computing, vol. 16, pp. 3224–3237, September/October 2023.

[36] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, and R. Sasnauskas, "A search based approach for accurate identification of log message formats," in Proceedings of the 26th Conference on Program Comprehension, ICPC '18, (New York, NY, USA), pp. 167–177, Association for Computing Machinery, 2018.

[37] A. Vervaet, R. Chiky, and M. Callau-Zori, "Ustep: Unfixed search tree for efficient log parsing," in 2021 IEEE International Conference on Data Mining (ICDM), IEEE, 2021.

[38] P. Chen, G. Chao, L. Yang, H. He, L. Hong, M. Li, D. Gao, and S. Guo, "A robust log parsing algorithm—practice of logslaw in heterogeneous logs of pacific credit card center of bank of communications (pccc)," in 2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA), IEEE, August 11-13 2023.

[39] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu, "Llmparser: A llm-based log parsing framework," arXiv preprint arXiv:2310.01796, October 2023.

[40] V.-H. Le and H. Zhang, "Log parsing: How far can chatgpt go?," arXiv preprint arXiv:2306.01590, August 2023.

[41] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Prompting for automatic log template extraction," July 2023. Affiliations: aThe Chinese University of Hong Kong, Shenzhen; the Chinese University of Hong Kong; cETH Zurich.

[42] H. Yang, D. Sun, Y. Wang, N. Zhao, S. Zhang, and W. Huang, "Adaptparse: Adaptive contextual aware attention network for log parsing via word classification," in 2023 International Joint Conference on Neural Networks (IJCNN), (Beijing, China), IEEE, 2023.

[43] S. Tao, Y. Liu, W. Meng, J. Wang, Y. Zhao, C. Su, W. Tian, M. Zhang, H. Yang, and X. Chen, "Da-parser: A pre-trained domain-aware parsing framework for heterogeneous log analysis," in 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), (Seoul, South Korea), IEEE, 2023.

[44] Y. Bai, Y. Chi, and D. Zhao, "Patcluster: A top-down log parsing method based on frequent words," IEEE Access, vol. 11, pp. 16003–16015, January 2023. This work was supported by the National Natural Science Foundation of China under Grant 52275136.

[45] L. Yu, T. Wu, J. Li, P. Chan, H. Min, and F. Meng, "Documentation based semanticaware log parsing," arXiv preprint arXiv:2202.07169, February 2022.

[46] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, "Logram: Efficient log parsing using n-gram dictionaries," IEEE Transactions on Software Engineering, vol. 48, pp. 879–892, March 2022.

[47] M. Siu, "Variable n-grams and extensions for conversational speech language modelling" IEEE Transactions on Speech and Audio Processing, vol. 8, pp. 63–75, January 2000.

[48] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," Environmental Research Institute of Michigan, 1994.

[49] G. Chu, J. Wang, Q. Qi, H. Sun, S. Tao, and J. Liao, "Prefix-graph: A versatile log parsing approach merging prefix tree with probabilistic graph," in 2021 IEEE 37th International Conference on Data Engineering (ICDE), IEEE, 2021.

[50] T. Marlaithong, V. C. Barroso, and P. Phunchongharn, "A log parsing framework for alice o2 facilities," IEEE Access, vol. 11, pp. 1–1, 2023. Received 11 June 2023, accepted 28 June 2023, date of publication 7 July 2023, date of current version 13 July 2023.

[51] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in 2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion (ICSE Companion), (Austin, TX, USA), IEEE, 2016.

[52] J.-O. Palacio-Niño and F. Berzal, "Evaluation metrics for unsupervised learning algorithms," arXiv preprint arXiv:1905.05667, 2019.

[53] S. Tao, W. Meng, Y. Cheng, Y. Zhu, Y. Liu, C. Du, T. Han, Y. Zhao, X. Wang, and H. Yang, "Logstamp: Automatic online log parsing based on sequence labelling," ACM SIGMETRICS Performance Evaluation Review, vol. 49, no. 4, pp. 93–98, 2022.

[54] S. Petrescu, A. Uta, F. den Hengst, and J. S. Rellermeyer, "Log parsing evaluation in the era of modern software systems," arXiv preprint arXiv:2308.09003, 2023. [cs.SE] 17 Aug 2023.

[55] S. Jain, A. de Buitléir, and E. Fallon, "A review of unstructured data analysis and parsing methods," in 2020 International Conference on Emerging Smart Computing and Informatics (ESCI), (Pune, India), AISSMS Institute of Information Technology, Mar 2020. Emails: sjain@ait.ie, efallon@ait.ie, amy.de.buitleir@ericsson.com.

[56] J. Ma, Y. Liu, H. Wan, and G. Sun, "Automatic parsing and utilization of system log features in log analysis: A survey," Applied Sciences, vol. 13, no. 8, p. 4930, 2023. Academic Editor: Christos Bouras, Received: 18 March 2023, Revised: 11 April 2023, Accepted: 12 April 2023, Published: 14 April 2023.

[57] T. Xiao, Z. Quan, Z.-J.Wang, K. Zhao, X. Liao, H. Huang, Y. Du, and K. Li, "Lpv: A log parsing framework based on vectorization," Journal of LaTeX Class Files, vol. 14, August 2021.

[58] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Paddy: An event log parsing approach using dynamic dictionary," in 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS), (Beijing, China), pp. 1–9, IEEE, 2020.

[59] H. Dai, Y. Tang, H. Li, and W. Shang, "Pilar: Studying and mitigating the influence of configurations on log parsing," in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. xx–xx, IEEE, 2023.

[60] X. Han, S. Yuan, and M. Trabelsi, "Loggpt: Log anomaly detection via gpt," arXiv preprint arXiv:2309.14482, 2023.

Tasnaim Ahmad Hosen
MSc. Software Engineering
College of Software
Nankai University, Tianjin, China
tasnemhosen91@gmail.com