



# An Assessment of ChatGPT on Log Data

Priyanka Mudgal<sup>(✉)</sup> and Rita Wouhaybi

Intel Corporation, Hillsboro, OR 97124, USA  
{priyanka.mudgal,rita.h.wouhaybi}@intel.com

**Abstract.** Recent development of large language models (LLMs), such as ChatGPT has been widely applied to a wide range of software engineering tasks. Many papers have reported their analysis on the potential advantages and limitations of ChatGPT for writing code, summarization, text generation, etc. However, the analysis of the current state of ChatGPT for log processing has received little attention. Logs generated by large-scale software systems are complex and hard to understand. Despite their complexity, they provide crucial information for subject matter experts to understand the system status and diagnose problems of the systems. In this paper, we investigate the current capabilities of ChatGPT to perform several interesting tasks on log data, while also trying to identify its main shortcomings. Our findings show that the performance of the current version of ChatGPT for log processing is limited, with a lack of consistency in responses and scalability issues. We also outline our views on how we perceive the role of LLMs in the log processing discipline and possible next steps to improve the current capabilities of ChatGPT and the future LLMs in this area. We believe our work can contribute to future academic research to address the identified issues.

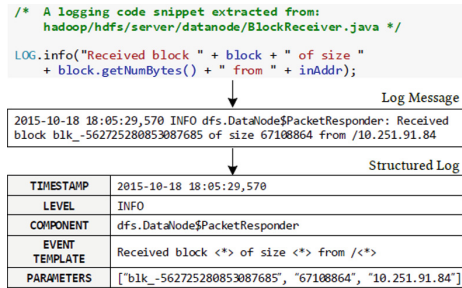
**Keywords:** log data · log analysis · log processing · ChatGPT · log analysis using LLM · large language model · deep learning · machine learning

## 1 Introduction

In recent years, the emergence of generative AI and large language models (LLMs) such as OpenAI's ChatGPT have led to significant advancements in NLP. Many of these models provide the ability to be fine-tuned on custom datasets [1–3] and achieve the state-of-the-art (SOTA) performance across various tasks. A few of the LLMs such as GPT-3 [4] have demonstrated in-context-learning capability without requiring any fine-tuning on task-specific data. The impressive performance of ChatGPT and other LLMs [5–9, 79] in zero-shot and few-shot learning scenarios is a major finding as this helps LLMs to be more efficient [74–78]. With such learning methodologies, the LLMs can be used as a service [10] to empower a set of new real-world applications.

Despite the impressive capability of ChatGPT in performing a wide range of challenging tasks, there remain some major concerns about it in solving real-world problems like log analysis [93]. Log analysis is a vast area, and much

research has been done. It mainly comprises three major categories, namely, log parsing, log analytics, and log summarization. Log parsing is an important initial step of system diagnostic tasks. Through log parsing, the raw log messages are converted into a structured format while extracting the template [11–14]. Log analytics can be used to identify the system events and dynamic runtime information, which can help the subject matter experts to understand system behavior and perform system diagnostic tasks, such as anomaly detection [15–18], log classification [19], error prediction [20, 21], and root cause analysis [22, 23]. Log analytics can further be used to perform advanced operations e.g., identify user activities, and security analysis e.g., detect logged-in users, API/service calls, malicious URLs, etc. As logs are huge in volume, log summarization enables the operators to provide a gist of the overall activities in logs and empowers the subject matter experts to read and/or understand logs faster. Recent studies leverage pre-trained language models [17, 24, 25] for representing log data. However, these methods still require either training the models from scratch [26] or tuning a pre-trained language model with labeled data [17, 24], which could be impractical due to the lack of computing resources and labeled data.

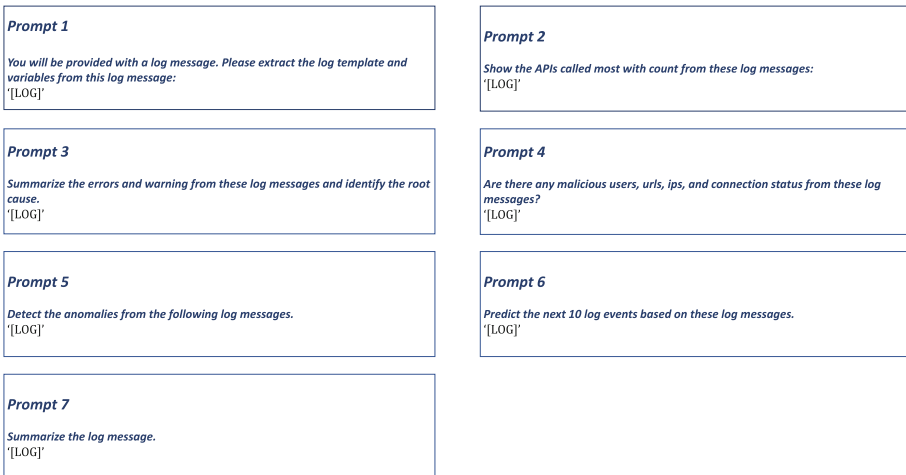


**Fig. 1.** An example of log code, log message, and structured log from [34]

More recently, LLMs such as ChatGPT [93] have been applied to a variety of software engineering tasks and achieved satisfactory performance [27, 28]. With a lack of studies to analyze ChatGPT’s capabilities on log processing, it is unclear whether it can be performed well on the logs. Although many papers have performed the evaluation of ChatGPT on software engineering tasks [29, 30, 33], specific research is required to investigate its capabilities in system log area. We are aware that the LLMs are fast evolving, with new models, versions, and tools being released frequently, and each one is improved over the previous ones. However, our goal is to assess the current situation and to provide a set of experiments that can enable the researchers to identify possible shortcomings of the current version for analyzing logs and provide a variety of specific tasks to measure the improvement of future versions. Hence, in this paper, we conduct an initial level of evaluation of ChatGPT on log data. Specifically, we divide the log processing [32] into three subsections: log parsing, log analytics, and log

summarization. We design appropriate prompts for each of these tasks and analyze ChatGPT’s capabilities in these areas. Our analysis shows that ChatGPT achieves promising results in some areas, but limited outcomes in others and contains several real-world challenges in terms of scalability. In summary, the major contributions of our work are as follows:

- To the best of our knowledge, we are the first to study and analyze ChatGPT’s ability to analyze the log data in multiple detailed aspects.
- We design the prompts for multiple scenarios in log processing and record ChatGPT’s response.
- Based on the findings, we outline several challenges and prospects for ChatGPT-based log processing.



**Fig. 2.** Various prompt designs to address the research questions.

## 2 Related Work

### 2.1 Log Data

With the increasing scale of software systems, it is complex to manage and maintain them. To tackle this challenge, engineers enhance the system observability [31, 99] with logs.

Logs capture multiple system run-time information such as events, transactions, and messages. A typical piece of log message is a time-stamped record that captures the activity that happened over time (e.g., software update events or received messages). Logs are usually generated when a system executes the

corresponding logging code snippets. An example of the code snippet and generated code is shown in Fig. 1. A system with mature logs essentially facilitates the system behavior understanding, health monitoring, failure diagnosis, etc. Generally, there are three standard log formats, i.e., structured, semi-structured, and unstructured logs [72]. These formats share the same components: a timestamp and a payload content.

Structured logs usually keep a consistent format within the log data and are easy to manage. Specifically, the well-structured format allows easy storing, indexing, searching, and aggregation in a relational database. The unstructured log data achieves its high flexibility at the expense of the ease of machine processing. The characteristic of free-form text becomes a major obstacle for efficient query and analysis on unstructured or semi-structured logs. For instance, to count how often an API version appears in unstructured logs, engineers need to design a complex query with ad-hoc regular expressions to extract the desired information. The manual process takes lots of time and effort and is not scalable.

## 2.2 Log Processing

Logs have been widely adopted in software system development and maintenance. In industry, it is a common practice to record detailed software runtime information into logs, allowing developers and support engineers to track system behaviors and perform postmortem analysis. On a high level, log processing can be categorized in three types as discussed below.

**Log Parsing.** Log parsing is generally the first step toward automated log analytics. It aims at parsing each log message into a specific log event/template and extracting the corresponding parameters. Although there are many traditional regular expression-based log parsers, but, they require a predefined knowledge about the log template. To achieve better performance in comparison to traditional log parsers, many data-driven [12,37–42] and deep learning based approaches [24,26] have been proposed to automatically distinguish template and parameter parts.

**Log Analytics.** Modern software development and operations rely on log monitoring to understand how systems behave in production. There is an increasing trend to adopt artificial intelligence to automate operations. Gartner [97] refers to this movement as AIOps. The research community, including practitioners, has been actively working to address the challenges related to extracting insights from log data also being referred to as “Log Analysis” [96]. Various insights that can be gained are in terms of log mining [85], error detection and root cause analysis, security and privacy, anomaly detection, and event prediction.

**Log Mining.** Log mining seeks to support understanding and analysis utilizing abstraction and extracting useful insights. However, building such models is a challenging and expensive task. In our study, we confine ourselves to posing

specific questions in terms of most API/service calls that can be extracted out of raw log messages. This area is well studied from a deep learning aspect and most of those approaches [49–56] require to first parse the logs and then process them to extract the detailed level of knowledge.

**Error Detection and Root Cause Analysis.** Automatic error detection from logs is an important part of monitoring solutions. Maintainers need to investigate what caused that unexpected behavior. Several studies [22, 43, 45–48] attempt to provide their useful contribution to root cause analysis, accurate error identification, and impact analysis.

**Security and Privacy.** Logs can be leveraged for security purposes, such as malicious behaviour and attack detection, URLs, and IP detection, logged-in user detection, etc. Several researchers have worked towards detecting early-stage malware and advanced persistence threat infections to identify malicious activities based on log data [57–61].

**Anomaly Detection.** Anomaly detection techniques addresses to identify the anomalous or undesired patterns in logs. The manual analysis of logs is time-consuming, error-prone, and unfeasible in many cases. Researchers have been trying several different techniques for automated anomaly detection, such as deep learning [62–65] and data mining, statistical learning methods, and machine learning [23, 66–71].

**Event Prediction.** The knowledge about the correlation of multiple events, when combined to predict the critical or interesting event is useful in preventive maintenance or predictive analytics that can reduce the unexpected system downtime and result in cost saving [80–82]. Thus, the event prediction method is highly valuable in real-time applications. In recent years, many rule-based and deep learning based approaches [83, 88–92] have evolved and performing significantly.

**Log Summarization.** Log statements are inserted in the source code to capture normal and abnormal behaviors. However, with the growing volume of logs, it becomes a time-consuming task to summarize the logs. There are multiple deep learning-based approaches [19, 44, 96, 98] that perform the summarization, but they require time and compute resources for training the models.

## 2.3 ChatGPT

ChatGPT is a large language model which is developed by OpenAI [93, 94]. ChatGPT is trained on a huge dataset containing massive amount of internet text. It offers the capability to generate text responses in natural language that are based on a wide range of topics. The fundamental of ChatGPT is generative pre-training transformer (GPT) architecture. GPT architecture is highly effective for natural language processing tasks such as translation in multiple languages, summarization, and question answering (Q & A). It offers the capability

to be fine-tuned on specific tasks with a smaller dataset with specific examples. ChatGPT can be adopted in a variety of use cases including chatbots, language translation, and language understanding. It is a powerful tool and possesses the potential to be used across wide range of industries and applications.

## 2.4 ChatGPT Evaluation

Several recent works on ChatGPT evaluation have been done, but most of the papers target the evaluations on general tasks [33, 73], code generation [27], deep learning-based program repair [28], benchmark datasets from various domains [29], software modeling tasks [30], information extraction [87], sentiment analysis of social media and research papers [84] or even assessment of evaluation methods [86]. The closest to our work is [35], but they focus only on log parsing.

We believe that the log processing area is huge and a large-level evaluation of ChatGPT on log data would be useful for the research community. Hence, in our work, we focus on evaluating ChatGPT by conducting an in-depth and wider analysis of log data in terms of log parsing, log analytics, and log summarization.

## 3 Context

In this paper, our primary focus is to assess the capability of ChatGPT on log data. In line with this, we aim to answer several research questions through experimental evaluation.

### 3.1 Research Questions

**Log Parsing RQ1.** How does ChatGPT perform on log parsing?

**Log Analytics RQ2.** Can ChatGPT extract the errors and identify the root cause from raw log messages?

**RQ3.** How does ChatGPT perform on advanced analytics tasks e.g., most called APIs/services?

**RQ4.** Can ChatGPT be used to extract security information from log messages?

**RQ5.** Is ChatGPT able to detect anomalies from log data?

**RQ6.** Can ChatGPT predict the next events based on previous log messages?

**Log Summarization RQ7.** Can ChatGPT summarize a single raw log messages?

**RQ8.** Can ChatGPT summarize multiple log messages?

**General RQ9.** Can ChatGPT process bulk log messages?

**RQ10.** What length of log messages can ChatGPT process at once?

To examine the effectiveness of ChatGPT in answering the research questions, we design specific prompts as shown in Fig. 2. We append the log messages in each of the prompts (in place of the slot ‘[LOG]’).

### 3.2 Dataset

To perform our experiments, we use the datasets provided from the Loghub benchmark [13, 34]. This benchmark covers log data from various systems, including, windows and linux operating systems, distributed systems, mobile systems, server applications, and standalone software. Each system dataset contains 2,000 manually labeled and raw log messages.

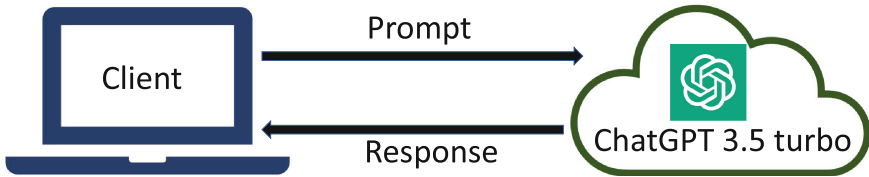


Fig. 3. Flow Diagram.

### 3.3 Experimental Setup

For our experiments, we are using the ChatGPT API based on the gpt-3.5-turbo model to generate the responses for different prompts [93]. As shown in Fig. 3, we send the prompts appended with log messages to ChatGPT from our system with Intel<sup>®</sup> Xeon<sup>®</sup> E3-1200 v5 processor and Intel<sup>®</sup> Xeon<sup>®</sup> E3-1500 v5 processor and receive the response. To avoid bias from model updates, we use a snapshot of gpt3.5-turbo from March 2023 [95].

### 3.4 Evaluation Metrics

As our study demands a detailed evaluation and in some cases, there was no state-of-the-art tool, we evaluated the output by our manual evaluation.

## 4 Experiments and Results

Each of the subsections below describes the individual evaluation of ChatGPT in different areas of log processing.

### 4.1 Log Parsing

In this experiment, we assess the capability of ChatGPT in parsing a raw log message and a preprocessed log message and find the answer to **RQ1**. For the first experiment, we provide a single raw log message from each of the sixteen publicly available datasets [34] and ask ChatGPT to extract the log template. We refer to it as first-level log parsing. ChatGPT performs well in extracting the

**Prompt: You will be provided with a log message. Please extract the log template and variables from this log message:**

```
03-17 16:13:38.819 1702 8671 D PowerManagerService: acquire
lock=233570404, flags=0x1, tag="View Lock", name=com.android.systemui,
ws=null, uid=10037, pid=2227
```

**ChatGPT Response:**

Log Template: {timestamp} {process\_id} {thread\_id} {log\_level} {tag}:  
{message}

Variables:

- timestamp: 03-17 16:13:38.819
- process\_id: 1702
- thread\_id: 8671
- log\_level: D
- tag: PowerManagerService
- message: acquire lock=233570404, flags=0x1, tag="View Lock", name=com.android.systemui, ws=null, uid=10037, pid=2227

**Fig. 4.** Log parsing of raw log message.

specific parts of log messages for all sixteen log messages. One of the examples of ChatGPT’s response for first-level log parsing is shown in Fig. 4. Next, we preprocess the log message, extract the content, and ask chatGPT to further extract the template from the log message. ChatGPT can extract the template and variables from the log message successfully on all sixteen log messages with a simple prompt. One of the examples of ChatGPT’s response is shown in Fig. 5.

## 4.2 Log Analytics

To evaluate ChatGPT’s capability in log analytics, we perform several experiments in each of the categories described in Sect. 2.2.

**Log Mining.** In this experiment, we are seeking the answer of **RQ2** by investigating if ChatGPT can skim out the knowledge from raw logs without building an explicit parsing pipeline. We perform our experiments in several parts. We provide a subset of log messages containing 5, 10, 20, and 50 log messages from Loghub benchmark [34] and ask ChatGPT to identify the APIs. Figure 6 shows an example of ChatGPT response when a smaller set of log messages were passed. We notice that ChatGPT consistently missed identifying some APIs from the log messages irrespective of the count of log messages, but still shows 75% or more accuracy in all cases. Results are reported in Table 1.

**Error Detection and Root Cause Analysis.** In this experiment, we explicitly ask ChatGPT [95] to identify the errors, warnings, and possible root causes of



<p><b>Prompt:</b> You will be provided with a log message. Please extract the log template and variables from this log message:</p> <p>PowerManagerService: acquire lock=233570404, flags=0x1, tag="View Lock",</p> <p><b>ChatGPT Response:</b></p> <p>Log Template: PowerManagerService: acquire lock={lock}, flags={flags}, tag="{tag}", name={name}, ws={ws}, uid={uid}, pid={pid}</p> <p>Variables:</p> <ul style="list-style-type: none"> <li>- lock: 233570404</li> <li>- flags: 0x1</li> <li>- tag: "View Lock"</li> <li>- name: com.android.systemui</li> <li>- ws: null</li> <li>- uid: 10037</li> <li>- pid: 2227</li> </ul>
--

**Fig. 5.** Log parsing of preprocessed log message.

**Table 1.** ChatGPT’s performance to identify the APIs, errors and root cause from Loghub dataset [34].

Log Message Count	API Count	API captured	API Accuracy (%)	API Response Time (s)	Error Count	Error captured	Error Accuracy (%)	Error Response time (s)
5	5	4	80	2.48	2	2	100	18.49
10	10	8	80	3.96	3	3	100	27.61
20	20	15	75	6.44	5	3	60	36.38
50	50	46	92	5.66	13	5	38.46	46.46

those in the provided log messages and address **RQ3**. Aligning towards our study structure, we first provide five log messages from the Loghub dataset [34] and later increase the size of log messages to ten, twenty, and fifty. Figure 7 shows the identified errors from five log messages and a detailed report for all the combinations with their response time is being reported in Table 1. It is evident from Table 1 that ChatGPT successfully identifies the errors and warnings on a smaller set of log messages than a larger set.

**Security and Privacy.** In this experiment, we focus on addressing **RQ4** and investigate if ChatGPT can identify the URLs, IPs, and logged users from the logs and extract knowledge about malicious activities. We use the open source dataset from Loghub [100] and follow the same approach of sending the set of five, ten, twenty, and fifty log messages to chatGPT to detect the URLs, IPs, and users from them. We use the ‘Prompt 4’ from Fig. 2 to ask if there are any malicious activities present in the logs. As shown in Table 2, ChatGPT extracts out the IPs and logged-in users with high accuracy irrespective of the length

**Prompt: Show the APIs called most with count from these log messages:**

```
03-17 16:13:46.671 1702 17633 I NotificationManager:
onNotificationVisibilityChanged called
03-17 16:13:46.672 2227 2318 I PhoneStatusBar:
logNotificationVisibilityChanges runInThread over
03-17 16:13:46.764 2227 2794 E KeyguardUpdateMonitor: isSimPinSecure
mSimDatas is null or empty
03-17 16:13:46.765 2227 2794 W KeyguardUpdateMonitor: registerCallback
```

**ChatGPT Response:**

API Name	Count
NotificationManager	1
PhoneStatusBar	1
KeyguardUpdateMonitor	2

**Fig. 6.** ChatGPT response to extract the APIs from log messages.

of log messages. An example of ChatGPT’s response is shown in Fig. 8. The detailed report is shown in Table 2.

**Table 2.** ChatGPT performance to extract urls, IPs, and users from the log messages from Loghub dataset [34].

Log Message Count	URLs Count	URLs captured	URL Accuracy (%)	User Count	User captured	User Accuracy (%)	Response time (s)
5	4	4	100	2	2	100	13.77
10	9	9	100	7	7	100	46.41
20	13	13	100	14	14	100	112.14
50	24	20	83.33	16	14	87.5	163.76

### Anomaly Detection

To evaluate ChatGPT’s capability to detect anomalies in logs and to address **RQ5**, we use ‘Prompt 5’ from Fig. 2. As detecting anomalies through log messages would require context, we append 200 log message entries and ask ChatGPT to detect anomalies from it. Without showing any examples to ChatGPT of how an anomaly might look like, it still tries to identify the possible anomalies and provide its analysis in the end. One of the examples is shown in Fig. 9.

### Event Prediction

It is interesting to evaluate ChatGPT’s performance in predicting future events

**Prompt: Summarize the errors and warning from these log messages and identify the root cause.**

```
[Sun Dec 04 04:52:49 2005] [notice] workerEnv.init() ok
/etc/httpd/conf/workers2.properties
[Sun Dec 04 04:52:49 2005] [notice] workerEnv.init() ok
/etc/httpd/conf/workers2.properties
[Sun Dec 04 04:52:52 2005] [error] mod_jk child workerEnv in error state 7
```

**ChatGPT Response:**

Date	Log Message	Tag	Root Cause
Sun Dec 04 04:52:49 2005	workerEnv.init() ok /etc/httpd/conf/workers2.properties	Notice	N/A
Sun Dec 04 04:52:52 2005	workerEnv.init() ok /etc/httpd/conf/workers2.properties	Notice	N/A
Sun Dec 04 04:52:52 2005	mod_jk child workerEnv in error state 7	Error	Issue with child workerEnv
Sun Dec 04 04:52:52 2005	mod_jk child workerEnv in error state 6	Error	Issue with child workerEnv
Sun Dec 04 04:53:05 2005	jk2_init() Found child 6750 in scoreboard slot 7	Notice	N/A

Root Cause: There is an issue with the child workerEnv which is causing it to be in an error state (state 7 or state 6).

**Fig. 7.** ChatGPT response to identify the errors and root cause from set of 5 log messages from Loghub dataset [34].

in log messages. Typically, for future event prediction, a context of past event is required, hence, we append 200 log messages to ‘Prompt 6’ from Fig. 2 and ask ChatGPT to predict the next 10 messages for simplicity. This experiment addresses the **RQ6**. While ChatGPT predicts the next 10 events in log format, it fails to predict even a single log message correctly when compared with the ground truth. ChatGPT’s response is shown in Fig. 10.

### 4.3 Log Summarization

This experiment is designed to understand if ChatGPT could succinctly summarize logs. We perform this study in two steps. First, To address the **RQ7**, we provide a single log message from each of the sixteen datasets of opensource benchmark [34] to ChatGPT to understand its mechanics. This is useful to understand the log message in natural language. Figure 11 shows one of the log messages from the Android subset of the Loghub dataset [34] and ChatGPT response. It is evident from the response that ChatGPT provides a detailed explanation of the log message. Next, to address the **RQ8**, we provide a set of ten log messages from each of the sixteen subsets of the Loghub dataset [34]

**Prompt: Are there any malicious users, urls, ips, and connection status from these log messages.**

```
Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
```

**ChatGPT Response:**

Date	Time	User	IP	Connection Status
Jun 14	15:16:01	Unknown	218.188.2.4	Authentication Failure
Jun 14	15:16:02	Unknown	218.188.2.4	Authentication Failure
Jun 15	02:04:59	root	220.135.151.1	Authentication Failure
Jun 15	02:04:59	root	220.135.151.1	Authentication Failure

Based on the log messages provided, there are no indicators of malicious activities. However, the repeated authentication failures from different IP addresses may suggest potential unauthorized access attempts. It is recommended to monitor and investigate any further unusual activities.

**Fig. 8.** ChatGPT response to extract urls, IPs, and users from set of 5 log messages from Loghub dataset [100].

to ChatGPT and ask to summarize the logs. ChatGPT generates a concrete summary collectively from the provided log messages as shown in Fig. 12. In Fig. 12, we only show a few log messages for visual clarity. ChatGPT generates an understandable summary for all the sixteen subsets.

## 5 Discussion

Based on our study, we highlight a few challenges and prospects for ChatGPT on log data analysis.

### 5.1 Handling Unstructured Log Data

For our experiments, we send the unstructured raw log messages to ChatGPT to analyze its capabilities on various log-specific tasks. Our study indicates that ChatGPT shows promising performance in processing the raw log messages. It is excellent in log parsing and identifying security and privacy information, but encounters difficulty in case of API detection, event prediction, and summarizing. It misses out on several APIs and events from raw log messages.

**Prompt: Detect the anomalies from the following log messages.**

```
Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
```

**ChatGPT Response:**

*Anomalies detected in the log messages are as follows:*

```
Jun 15 04:06:20 combo logrotate: ALERT exited abnormally with [1]
This indicates an abnormal exit of logrotate process.
```

```
Jun 15 12:12:34 combo sshd(pam_unix)[23397]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23395]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
```

*These log messages indicate multiple authentication failures from the same IP address (rhost). This could signify a security threat or unauthorized access attempts.*

**Fig. 9.** ChatGPT response for anomaly detection for a sample from Loghub dataset [34].

## 5.2 Performance with Zero-Shot Learning

We perform our experiments with zero-shot learning. Our experimental results show that ChatGPT exhibits good performance in the areas of log parsing, security, and privacy, and average performance in the case of API detection, incident detection, and root cause identification. As ChatGPT supports few-shot learning, it remains an important future work to select important guidelines to set effective examples and evaluate ChatGPT's performance with them.

## 5.3 Scalability - Message Cap for GPT

Most of the intelligent knowledge extraction from logs depends on processing a large amount of the logs in a short period. As ChatGPT 3.5 can only process limited tokens at once, it poses a major limitation in feeding the bigger chunk of log data. For our experiments, we could only send 190 to 200 log messages appended (addressing **RQ9** and **RQ10**) with the appropriate prompt at once. As most of

**Prompt: Predict the next 10 log events based on these log messages.**

```
Jun 14 15:16:01 combo sshd(pam_unix)[19939]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: check pass; user unknown
Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-
ip.hinet.net user=root
```

**ChatGPT Response:**

*Based on the given log messages, the next 10 log events could be:*

1. Jun 17 04:08:47 combo sshd(pam\_unix)[28324]: check pass; user unknown
2. Jun 17 04:08:47 combo sshd(pam\_unix)[28324]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
3. Jun 17 04:08:47 combo sshd(pam\_unix)[28322]: check pass; user unknown
4. Jun 17 04:08:47 combo sshd(pam\_unix)[28322]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
5. Jun 17 04:08:47 combo sshd(pam\_unix)[28329]: check pass; user unknown
6. Jun 17 04:08:47 combo sshd(pam\_unix)[28329]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
7. Jun 17 04:08:47 combo sshd(pam\_unix)[28325]: check pass; user unknown
8. Jun 17 04:08:47 combo sshd(pam\_unix)[28325]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
9. Jun 17 04:08:47 combo sshd(pam\_unix)[28328]: check pass; user unknown
10. Jun 17 04:08:47 combo sshd(pam\_unix)[28328]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4

**Fig. 10.** ChatGPT response for event prediction from Loghub dataset [34].

**Prompt: Summarize the log message.**

```
PowerManagerService: acquire lock=233570404, flags=0x1, tag="View Lock",
name=com.android.systemui, ws=null, uid=10037, pid=2227
```

**ChatGPT Response:**

On March 17th at 16:13:38.819, the PowerManagerService acquired a lock with certain flags and attributes. The tag for this lock was "View Lock" and it belonged to the package "com.android.systemui". The lock was associated with a specific process ID (PID) and user ID (UID).

**Fig. 11.** Summary generated by ChatGPT for single log message from Loghub dataset [34].

**Prompt: Summarize the log messages.**

```
03-17 16:13:38.811 1702 2395 D WindowManager: printFreezingDisplayLogsopening app
wtoken = AppWindowToken{9f4ef63 token=Token{a64f992 ActivityRecord{de9231d u0
com.tencent.qt.qml/.activity.info.NewsDetailXmlActivity t761}}}, allDrawn= false,
startingDisplayed = false, startingMoved = false, isRelaunching = false
03-17 16:13:38.819 1702 8671 D PowerManagerService: acquire lock=233570404,
flags=0x1, tag="View Lock", name=com.android.systemui, ws=null, uid=10037, pid=2227
03-17 16:13:38.820 1702 8671 D PowerManagerService:
ready=true,policy=3,wakefulness=1,wksummary=0x23,uasummary=0x1,bootcompleted=
true,boostinprogress=false,waitmodeenable=false,mode=false>manual=38,auto=-
```

**ChatGPT Response:**

The log message shows various activities and operations happening in the system. These include the opening of an app window token, acquisition of a lock by the PowerManagerService, and manipulation of TextView visibility.

**Fig. 12.** Collective summary generated by ChatGPT for ten log messages from Loghub dataset [34].

the real-time applications would require to continuously send larger chunks of log messages to a system for processing, this limitation of ChatGPT 3.5 may pose a major hindrance in terms of scalability making them less suitable for tasks that require up-to-date knowledge or rapid adaptation to changing contexts. With the newer versions of ChatGPT, the number of tokens may be increased which would make it more suitable for its application in the log processing area.

## 5.4 Latency

The response time of ChatGPT ranges from a few seconds to minutes when the number of log messages is increased in the prompt. The details about response time are shown in Table 1 and 2. Most of the intelligent knowledge extraction from logs depends on the processing time of the large amount of the logs. With the current state of response time, ChatGPT would face a major challenge in real-time applications, where a response is required in a shorter period. As currently, we have to call openAI API to get ChatGPT's response, with the newer versions of ChatGPT, it may be possible to deploy these models close to applications and reduce the latency significantly.

## 5.5 Privacy

Log data often contains sensitive information that requires protection. It is crucial to ensure that log data is stored and processed securely to safeguard sensitive information. It is also important to consider appropriate measures to mitigate any potential risks.

## 6 Conclusion

This paper presents the first evaluation to give a comprehensive overview of ChatGPT’s capability on log data from three major areas: log parsing, log analytics and log summarization. We have designed specific prompts for ChatGPT to reveal its capabilities in the area of log processing. Our evaluations reveal that the current state of ChatGPT exhibits excellent performance in the areas of log parsing, but poses certain limitations in other areas i.e., API detection, anomaly detection, log summarization, etc. We identify several grand challenges and opportunities that future research should address to improve the current capabilities of ChatGPT.

## 7 Disclaimer

The goal of this paper is mainly to summarize and discuss existing evaluation efforts on ChatGPT along with some limitations. The only intention is to foster a better understanding of the existing framework. Additionally, due to the swift evolution of LLMs especially ChatGPT, they would likely become more robust, and some of their limitations described in this paper are remediated. We encourage interested readers to take this survey as a reference for future research and conduct real experiments in current systems when performing evaluations. Finally, with continuous evaluation of LLMs, we may miss some new papers or benchmarks. We welcome all constructive feedback and suggestions to help make this evaluation better.

## References

1. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: GLUE: a multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 353–355 (2018). <https://aclanthology.org/W18-5446>
2. Wang, B.: Mesh-Transformer-JAX: model-parallel implementation of transformer language model with JAX (2021)
3. Wang, B., Komatsuzaki, A.: GPT-J-6B: a 6 billion parameter autoregressive language model (2021)
4. Brown, T., et al.: Language models are few-shot learners (2020)
5. Tay, Y., et al.: UL2: unifying language learning paradigms (2023)
6. Thoppilan, R., et al.: LaMDA: language models for dialog applications (2022)
7. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.* **23**(1), 5232–5270 (2022)
8. Hoffmann, J., et al.: Training compute-optimal large language models (2022)
9. Zeng, A., et al.: GLM-130B: an open bilingual pre-trained model (2022)
10. Sun, T., Shao, Y., Qian, H., Huang, X., Qiu, X.: Black-box tuning for language-model-as-a-service (2022)



11. Du, M., Li, F.: Spell: streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 859–864 (2016). <https://api.semanticscholar.org/CorpusID:206784678>
12. He, P., Zhu, J., Zheng, Z., Lyu, M.: Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 33–40 (2017)
13. Zhu, J., et al.: Tools and benchmarks for automated log parsing (2019)
14. Khan, Z., Shin, D., Bianculli, D., Briand, L.: Guidelines for assessing the accuracy of log message template identification techniques. In: Proceedings of the 44th International Conference on Software Engineering, pp. 1095–1106 (2022). <https://doi.org/10.1145/3510003.3510101>
15. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298 (2017). <https://doi.org/10.1145/3133956.3134015>
16. Zhang, X., et al.: Robust log-based anomaly detection on unstable log data. In: 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>
17. Le, V., Zhang, H.: Log-based anomaly detection without log parsing. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, pp. 492–504 (2022). <https://doi.org/10.1109/ASE51524.2021.9678773>
18. Zhang, B., Zhang, H., Le, V., Moscato, P., Zhang, A.: Semi-supervised and unsupervised anomaly detection by mining numerical workflow relations from system logs. *Autom. Softw. Eng.* **30** (2022). <https://doi.org/10.1007/s10515-022-00370-w>
19. Ramachandran, S., Agrahari, R., Mudgal, P., Bhilwaria, H., Long, G., Kumar, A.: Automated log classification using deep learning. *Procedia Comput. Sci.* **218**, 1722–1732 (2023). International Conference on Machine Learning and Data Engineering. <https://www.sciencedirect.com/science/article/pii/S1877050923001503>
20. Das, A., Mueller, F., Siegel, C., Vishnu, A.: Desh: deep learning for system health prediction of lead times to failure in HPC. In: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, pp. 40–51 (2018). <https://doi.org/10.1145/3208040.3208051>
21. Russo, B., Succi, G., Pedrycz, W.: Mining system logs to learn error predictors: a case study of a telemetry system. *Empirical Softw. Eng.* **20**, 879–927 (2015). <https://api.semanticscholar.org/CorpusID:19032755>
22. Gurumdimma, N., Jhumka, A., Liakata, M., Chuah, E., Browne, J.: CRUDE: combining resource usage data and error logs for accurate error detection in large-scale distributed systems. In: 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS), pp. 51–60 (2016)
23. Lu, S., Rao, B., Wei, X., Tak, B., Wang, L., Wang, L.: Log-based abnormal task detection and root cause analysis for spark. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 389–396 (2017)
24. Le, V., Zhang, H.: Log parsing with prompt-based few-shot learning (2023)
25. Tao, S., et al.: LogStamp: automatic online log parsing based on sequence labelling (2022)
26. Liu, Y., et al.: UniParser: a unified log parser for heterogeneous log data. In: Proceedings of the ACM Web Conference 2022 (2022). <https://doi.org/10.1145/3485447.3511993>

27. Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M., Chen, H.: Investigating code generation performance of ChatGPT with crowdsourcing social data. In: 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 876–885 (2023)
28. Cao, J., Li, M., Wen, M., Cheung, S.: A study on prompt design, advantages and limitations of ChatGPT for deep learning program repair (2023)
29. Laskar, M., Bari, M., Rahman, M., Bhuiyan, M., Joty, S., Huang, J.: A systematic study and comprehensive evaluation of ChatGPT on benchmark datasets (2023)
30. Cámara, J., Troya, J., Burgueño, L., Vallecillo, A.: On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Softw. Syst. Model.* **22**, 781–793 (2023). <https://doi.org/10.1007/s10270-023-01105-5>
31. Sridharan, C.: *Distributed Systems Observability: A Guide to Building Robust Systems*. O'Reilly Media Inc. (2018)
32. He, S., et al.: An empirical study of log analysis at Microsoft. In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1465–1476 (2022)
33. Sridhara, G., Ranjani, H.G., Mazumdar, S.: ChatGPT: a study on its utility for ubiquitous software engineering tasks (2023)
34. He, S., Zhu, J., He, P., Lyu, M.: Loghub: a large collection of system log datasets towards automated log analytics (2020)
35. Le, V., Zhang, H.: An evaluation of log parsing with ChatGPT (2023)
36. Fu, Q., Lou, J., Wang, Y., Li, J.: Execution anomaly detection in distributed systems through unstructured log analysis. In: 2009 Ninth IEEE International Conference on Data Mining, pp. 149–158 (2009)
37. Tang, L., Li, T., Perng, C.: LogSig: generating system events from raw textual logs. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 785–794 (2011). <https://doi.org/10.1145/2063576.2063690>
38. Shima, K.: Length matters: clustering system log messages using length of words. [arXiv:1611.03213](https://arxiv.org/abs/1611.03213) (2016). <https://api.semanticscholar.org/CorpusID:16326353>
39. Dai, H., Li, H., Shang, W., Chen, T., Chen, C.: Logram: efficient log parsing using n-gram dictionaries (2020)
40. Nagappan, M., Vouk, M.: 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 (2010)
41. Vaarandi, R.: 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), pp. 119–126 (2003)
42. Wang, X., et al.: SPINE: a scalable log parser with feedback guidance. In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1198–1208 (2022). <https://doi.org/10.1145/3540250.3549176>
43. Zheng, Z., et al.: Co-analysis of RAS log and job log on Blue Gene/P. In: 2011 IEEE International Parallel & Distributed Processing Symposium, pp. 840–851 (2011)
44. Meng, W., et al.: Summarizing unstructured logs in online services (2020)
45. Chuah, E., Jhumka, A., Narasimhamurthy, S., Hammond, J., Browne, J., Barth, B.: Linking resource usage anomalies with system failures from cluster log data. In: 2013 IEEE 32nd International Symposium on Reliable Distributed Systems, pp. 111–120 (2013)

46. Pi, A., Chen, W., Zhou, X., Ji, M.: Profiling distributed systems in lightweight virtualized environments with logs and resource metrics. In: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, pp. 168–179 (2018)
47. Ren, Z., Liu, C., Xiao, X., Jiang, H., Xie, T.: Root cause localization for unreproducible builds via causality analysis over system call tracing. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 527–538 (2019)
48. Kimura, T., et al.: Spatio-temporal factorization of log data for understanding network events. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 610–618 (2014)
49. Steinle, M., Aberer, K., Girdzijauskas, S., Lovis, C.: Mapping moving landscapes by mining mountains of logs: novel techniques for dependency model generation. *VLDB* **6**, 1093–1102 (2006)
50. Lou, J., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: 2010 USENIX Annual Technical Conference (USENIX ATC 2010) (2010)
51. Kc, K., Gu, X.: ELT: efficient log-based troubleshooting system for cloud computing infrastructures. In: 2011 IEEE 30th International Symposium on Reliable Distributed Systems, pp. 11–20 (2011)
52. Awad, M., Menascé, D.: Performance model derivation of operational systems through log analysis. In: 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MAS-COTS), pp. 159–168 (2016)
53. Tan, J., Kavulya, S., Gandhi, R., Narasimhan, P.: Visual, log-based causal tracing for performance debugging of MapReduce systems. In: 2010 IEEE 30th International Conference on Distributed Computing Systems, pp. 795–806 (2010)
54. Beschastnikh, I., Brun, Y., Ernst, M., Krishnamurthy, A.: Inferring models of concurrent systems from logs of their behavior with CSight. In: Proceedings of the 36th International Conference on Software Engineering, pp. 468–479 (2014)
55. Mariani, L., Pastore, F.: Automated identification of failure causes in system logs. In: 2008 19th International Symposium on Software Reliability Engineering (ISSRE), pp. 117–126 (2008)
56. Ulrich, A., Hallal, H., Petrenko, A., Boroday, S.: Verifying trustworthiness requirements in distributed systems with formal log-file analysis. In: 2003 Proceedings of the 36th Annual Hawaii International Conference on System Sciences, p. 10 (2003)
57. Oprea, A., Li, Z., Yen, T., Chin, S., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 45–56 (2015)
58. Balzarotti, D., Stolfo, S., Cova, M.: *Research in Attacks, Intrusions and Defenses*. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-642-33338-5>
59. Yoon, E., Squicciarini, A.: Toward detecting compromised MapReduce workers through log analysis. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 41–50 (2014)
60. Yen, T., et al.: Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 199–208 (2013)
61. Gonçalves, D., Bota, J., Correia, M.: Big data analytics for detecting host misbehavior in large logs. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 238–245 (2015)

62. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298 (2017)
63. Bertero, C., Roy, M., Sauvanaud, C., Trédan, G.: Experience report: log mining using natural language processing and application to anomaly detection. In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), pp. 351–360 (2017)
64. Meng, W., et al.: LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI 2019, pp. 4739–4745 (2019)
65. Zhang, X., et al.: Robust log-based anomaly detection on unstable log data. In: 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)
66. He, S., Zhu, J., He, P., Lyu, M.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207–218 (2016)
67. Cândido, J., Aniche, M., Deursen, A.: Log-based software monitoring. *PeerJ Comput. Sci.* **7**, e489 (2021)
68. Tang, D., Iyer, R.: Analysis of the VAX/VMS error logs in multicomputer environments—a case study of software dependability. In: Proceedings Third International Symposium on Software Reliability Engineering, pp. 216–217 (1992)
69. Lim, C., Singh, N., Yajnik, S.: A log mining approach to failure analysis of enterprise telephony systems. In: 2008 IEEE International Conference on Dependable Systems and Networks with FTCS and DCC (DSN), pp. 398–403 (2008)
70. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 117–132 (2009)
71. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.: Online system problem detection by mining patterns of console logs. In: 2009 Ninth IEEE International Conference on Data Mining, pp. 588–597 (2009)
72. Gandomi, A., Haider, M.: Beyond the hype: big data concepts, methods, and analytics. *Int. J. Inf. Manag.* **35**, 137–144 (2015). <https://www.sciencedirect.com/science/article/pii/S0268401214001066>
73. Chang, Y., et al.: A survey on evaluation of large language models (2023)
74. Chang, M., Ratinov, L., Roth, D., Srikumar, V.: Importance of semantic representation: dataless classification. In: Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 2, pp. 830–835 (2008)
75. Larochelle, H., Erhan, D., Bengio, Y.: Zero-data learning of new tasks. In: Proceedings of the 23rd National Conference on Artificial Intelligence, vol. 2, pp. 646–651 (2008)
76. Palatucci, M., Pomerleau, D., Hinton, G., Mitchell, T.: Zero-shot learning with semantic output codes. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems, pp. 1410–1418 (2009)
77. Lampert, C., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 951–958 (2009). <https://api.semanticscholar.org/CorpusID:10301835>
78. Miller, E., Matsakis, N., Viola, P.: Learning from one example through shared densities on transforms. In: Proceedings of the IEEE Conference on Computer

- Vision and Pattern Recognition, CVPR 2000 (Cat. No. PR00662), vol. 1, pp. 464–471 (2000)
79. Scao, T., et al.: BLOOM: a 176B-parameter open-access multilingual language model (2023)
  80. Fu, X., Ren, R., Zhan, J., Zhou, W., Jia, Z., Lu, G.: LogMaster: mining event correlations in logs of large-scale cluster systems. In: 2012 IEEE 31st Symposium on Reliable Distributed Systems, pp. 71–80 (2012)
  81. Tama, B., Comuzzi, M.: An empirical comparison of classification techniques for next event prediction using business process event logs. *Expert Syst. Appl.* **129**, 233–245 (2019). <https://www.sciencedirect.com/science/article/pii/S0957417419302465>
  82. Shmueli, G., Koppius, O.: Predictive analytics in information systems research. *MIS Q.* **35**, 553–572 (2011). <http://www.jstor.org/stable/23042796>
  83. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: from the past to present and future. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 38–52. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13094-6\\_5](https://doi.org/10.1007/978-3-642-13094-6_5)
  84. Leiter, C., et al.: ChatGPT: a meta-analysis after 2.5 months (2023)
  85. Pettinato, M., Gil, J., Galeas, P., Russo, B.: Log mining to re-construct system behavior: an exploratory study on a large telescope system. *Inf. Softw. Technol.* **114**, 121–136 (2019). <https://www.sciencedirect.com/science/article/pii/S0950584919301429>
  86. Mao, R., Chen, G., Zhang, X., Guerin, F., Cambria, E.: GPTEval: a survey on assessments of ChatGPT and GPT-4 (2023)
  87. Li, B., et al.: Evaluating ChatGPT’s information extraction capabilities: an assessment of performance, explainability, calibration, and faithfulness (2023)
  88. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Matthias, J., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-07881-6\\_31](https://doi.org/10.1007/978-3-319-07881-6_31)
  89. Francescomarino, C., Ghidini, C., Maggi, F., Petrucci, G., Yeshchenko, A.: An eye into the future: leveraging a-priori knowledge in predictive business process monitoring. In: International Conference on Business Process Management (2017). <https://api.semanticscholar.org/CorpusID:206703657>
  90. Verenich, I., Dumas, M., Rosa, M., Maggi, F., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *ACM Trans. Intell. Syst. Technol.* **10**(4), 1–34 (2019)
  91. Evermann, J., Rehse, J., Fettke, P.: Predicting process behaviour using deep learning. *Decis. Support Syst.* **100**, 129–140 (2017)
  92. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59536-8\\_30](https://doi.org/10.1007/978-3-319-59536-8_30)
  93. ChatGPT: Optimizing language models for dialogue. <https://openai.com/blog/chatgpt>
  94. ChatGPT: UI. <https://chat.openai.com/>
  95. ChatGPT: ChatGPT 3.5. <https://platform.openai.com/docs/models/gpt-3-5>
  96. LogAI: A Library for Log Analytics and Intelligence. <https://blog.salesforceairesearch.com/logai/>
  97. LogAI: A Library for Log Analytics and Intelligence. <https://www.gartner.com/smarterwithgartner/how-to-get-started-with-aiops>

98. How to Get Started with AIOps. <https://www.zebrium.com/blog/part-1-machine-learning-for-logs>
99. Observability. <https://en.wikipedia.org/wiki/Observability>
100. Zhu, J., He, S., He, P., Liu, J., Lyu, M.: Loghub: a large collection of system log datasets for ai-driven log analytics. In: IEEE International Symposium on Software Reliability Engineering (ISSRE) (2023)