arXiv:2505.03100v1 [cs.CR] 6 May 2025

Towards a standardized methodology and dataset for evaluating LLM-based digital forensic timeline analysis

Hudan Studiawan[a,*], Frank Breitinger[b], Mark Scanlon[c]

[a]Department of Informatics, Institut Teknologi Sepuluh Nopember, Indonesia
[b]Institute of Computer Science, University of Augsburg, Augsburg, Germany
[c]Forensics and Security Research Group, School of Computer Science, University College Dublin, Ireland

Abstract

Large language models (LLMs) have seen widespread adoption in many domains including digital forensics. While prior research has largely centered on case studies and examples demonstrating how LLMs can assist forensic investigations, deeper explorations remain limited, i.e., a standardized approach for precise performance evaluations is lacking. Inspired by the NIST Computer Forensic Tool Testing Program, this paper proposes a standardized methodology to quantitatively evaluate the application of LLMs for digital forensic tasks, specifically in timeline analysis. The paper describes the components of the methodology, including the dataset, timeline generation, and ground truth development. Additionally, the paper recommends using BLEU and ROUGE metrics for the quantitative evaluation of LLMs through case studies or tasks involving timeline analysis. Experimental results using ChatGPT demonstrate that the proposed methodology can effectively evaluate LLM-based forensic timeline analysis. Finally, we discuss the limitations of applying LLMs to forensic timeline analysis.

Keywords: LLM evaluation, Forensic timeline, Large language models, ChatGPT, log2timeline/Plaso

1. Introduction

Forensic investigations often require the reconstruction of a timeline of events and activities related to a digital device or users (Hargreaves and Patterson, 2012). Such timelines can provide valuable insights into various criminal activities, including malware, brute-force attacks, or attacker post-exploitation activities. The timeline analysis process is complex and time-consuming, particularly when dealing with large amounts of digital data from multiple sources. Traditional methods for timeline analysis are based on manual analysis, which can be subjective and prone to errors, and can lead to missing critical information (Studiawan et al., 2020).

The development of large language models (LLMs), such as OpenAI's GPT-3 (Brown et al., 2020) has opened up many possibilities, including in digital forensic research. The model has been implemented in the ChatGPT application and instantly gained many users (Buchholz, 2023). LLMs have shown remarkable performance in various natural language processing tasks, including language generation and question-answering. Leveraging these capabilities, natural language processing techniques can be applied to digital data sources to analyze temporal information and investigate timelines of events. Other studies also suggest that artificial intelligence should provide more assistance in forensic investigation (Hall et al., 2022; Studiawan et al., 2019).

An editorial article by Scanlon et al. (2023b) discusses the increasing demand for expert digital forensic analysts and the potential use of LLMs such as ChatGPT in this domain. They emphasize the importance of maintaining the "AI-assisted investigation" and "human-in-the-loop" mantras when using LLMs in digital forensics. The article suggests that LLMs could lead to a new career specialization of digital forensic prompt engineers. Wickramasekara et al. (2025) provides a comprehensive overview of where LLMs may assist digital forensics.

In addition, various studies explored the application of LLMs in digital forensics. For instance, (Scanlon et al., 2023a) assessed ChatGPT's impact on tasks such as understanding artifacts, evidence searching, and anomaly detection. Although ChatGPT shows promise in several low-risk forensic applications, concerns arise about evidence security and the model's occasional inaccuracies. Experts must exercise caution and have a deep understanding of the subject to effectively use ChatGPT in forensic scenarios. Furthermore, ChatGPT has been explored for digital evidence investigations (Henseler and van Beek, 2023), virtual forensic assistants (Dinis-Oliveira and Azevedo, 2023), and report writing (Michelet and Breitinger, 2024). Based on our literature review, existing work in this area has not discussed standardized evaluation for LLM-based digital investigation.

Contribution. The contributions of this paper are as follows:

1. This paper proposes a standardized methodology to quantitatively evaluate the performance of LLMs in forensic timeline analysis tasks, such as event summarization.
2. This study presents a case study of forensic timeline analysis using LLM, e.g., ChatGPT.

---

[*] Corresponding author

Email addresses: `hudan@its.ac.id` (Hudan Studiawan), `frank.breitinger@uni-a.de` (Frank Breitinger), `mark.scanlon@ucd.ie` (Mark Scanlon)

3. We created forensic timeline datasets and ground truth from Windows 11 using Plaso and these are publicly available[1] for research and education purposes.

The remainder of the paper is organized as follows: Sec. 2 provides related research. Sec. 3 describes the proposed approach for standard methodology and quantitative evaluation for LLM-based timeline analysis. Sec. 4 presents the case study that demonstrates the application of the proposed method and a discussion of the results. Finally, Sec. 5 concludes this study.

2. Related work

2.1. Forensic tool testing and validation

To effectively validate digital forensic tools and methods, a proper validation test plan should include laboratory use in the real world, controlled internal tests based on scientific principles, and peer review. Brunty (2023) provides an overview of the foundational scientific aspects of forensic validations and describes the recommended steps to conduct a forensically sound validation method.

The Computer Forensics Tool Testing (CFTT) Program at NIST aims to establish a methodology for testing computer forensic tools, including developing specifications, test procedures, and criteria (NIST, 2019). The program helps to ensure the reliability of forensic software tools, helping tool makers, users, and interested parties. CFTT methodology involves breaking down forensic tasks into discrete functions and creating test methodologies for each.

Hughes and Karabiyik (2020) discuss the need for rigorous validation practices in digital forensics to establish accuracy and reliability. They highlight challenges in developing statistical confidence for forensic tools, such as the lack of reference data, validation methods, and precise definitions of measurement. The authors propose a method for generated data procedures, virtual machine-based validation, and empirical models to guide the analysis.

Another study discusses the challenges of scientifically validating digital forensic evidence (Arshad et al., 2018). The authors emphasize the lack of standard datasets, formal testing procedures, and established error rates. Horsman (2019) examines the challenges of ensuring reliability in digital forensic tools. The paper discusses the lack of standardized validation methods and the issues of transparency from software vendors. A survey of practitioners reveals widespread concerns about tool reliability and a need for improved testing standards and error rate reporting.

The related study on tool testing and validation shows a research gap where we need a method to evaluate and validate LLMs as tools in digital forensics. This paper aims to fulfill this need specifically for the task of forensic timeline analysis.

2.2. Forensic timeline analysis

Forensic timeline analysis involves reconstructing the sequence of events and activities related to a user or a system. Therefore, a variety of artifacts, such as browsing history, log files, or file metadata, are being parsed, and relevant information is extracted (Palmbach and Breitinger, 2020). The analysis of the timeline analysis is then conducted using tools and data visualization techniques (Inglot and Liu, 2014). If tools do not yield expected results, a manual examination of data sources may be required. However, this approach can be time-consuming, labor-intensive, and prone to errors.

Timeline generation tools, such as log2timeline/Plaso, Autopsy, and Magnet AXIOM, can automate the timeline analysis process to some extent by extracting relevant temporal information from digital data sources. However, these tools are limited by the quality of the extracted data and may not be able to capture all relevant events and activities from acquired artifacts (Studiawan et al., 2022a).

The approach by Hargreaves and Patterson (2012) can automatically reconstruct or summarize high-level events from low-level events. Previous techniques focus on extracting times from a disk image into a timeline, which can produce several million "low-level" events (e.g., file modification or Registry key update) for a single disk. In contrast, this approach can automatically reconstruct high-level events (e.g. connection of a USB stick) from this set of low-level events. The knowledge representation model presented in Chabot et al. (2014) allows a semantically rich representation of events related to the incident. It includes the identification of correlated events that can highlight valuable information for the investigators.

The construction of a timeline array using time information from web browser log files is one way to perform forensic timeline analysis (Nalawade et al., 2016). Different data types of timelines that can be constructed from web browser artifacts such as web history, cache, cookie, download history, and search term timelines. Furthermore, Bhandari and Jusas (2020) propose an abstraction-based approach to reconstruct a timeline of events and artifacts. The method enhances the relevance of the timeline by reconstructing it into four levels of depth, from general to specific, to reduce complexity and extract information.

The use of deep learning techniques, e.g., autoencoders, improves anomaly detection in a forensic timeline by establishing a baseline for normal activities (Studiawan and Sohel, 2021). Another tool, namely DroneTimeline, constructs a timeline from a drone device and considers time extracted not only from file metadata, but also from various source artifacts of a drone or its control devices (Studiawan et al., 2022b).

2.3. LLMs for digital forensics

In the case of LLM application for digital forensics, Henseler and van Beek (2023) discuss how ChatGPT can assist investigators by writing structured queries, summarizing and evaluating large volumes of communication data, and analyzing search results. The authors highlight that ChatGPT can transform natural language queries into structured formats, summarize and

---

[1] `https://zenodo.org/blinded_for_review`

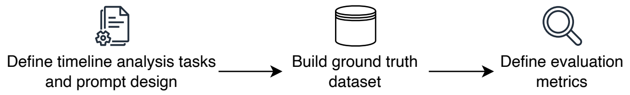

Define timeline analysis tasks and prompt design
Build ground truth dataset
Define evaluation metrics

Figure 1: The proposed methodology for quantitative evaluation of LLM-based timeline analysis

visualize chat logs to reveal key relationships. The study notes limitations, such as hallucinations and the need for expert guidance. Another work explores the potential of using LLMs, e.g., ChatGPT and Llama, to assist in the generation of forensic reports in digital investigations (Michelet and Breitinger, 2024). The authors assess the ability of LLMs to automate parts of the report writing process, focusing on sections such as the introduction, items received, methodology, and results. They found that while ChatGPT performs well and generates relatively accurate drafts, Llama struggles with accuracy and completeness. The results show that LLM outputs still require proofreading and corrections.

Dinis-Oliveira and Azevedo (2023) also explore the potential and challenges of using ChatGPT in forensic sciences. The authors highlight the advantages of ChatGPT, such as assisting forensic professionals in drafting reports, analyzing forensic data, performing literature searches, and serving as a virtual forensic assistant. However, the paper also raises concerns about the ethical and legal challenges associated with using AI in this field, such as credibility issues, inaccuracies, plagiarism, and the risk of overreliance on AI in judicial decisions.

Finally, Scanlon et al. (2023a) describes the potential applications of ChatGPT and LLMs in digital forensics. The authors assess how ChatGPT can assist in various forensic tasks, such as identifying digital artifacts, generating code for forensic activities, and detecting anomalies in logs. LLMs present challenges including issues with hallucinations, inaccuracies, and limitations when dealing with sensitive data. The study shows that ChatGPT can be a useful tool for investigators when used with caution, but human expertise remains essential to ensure reliability in forensic investigations.

The application of LLMs in digital forensics has the potential to enhance investigators' capabilities to handle digital evidence and help solve cases with greater accuracy. However, it is important to remember that LLMs are not a replacement for human expertise, but rather a valuable tool that complements and assists forensic professionals. Therefore, we need a methodology and a dataset to evaluate LLMs as a forensic tool, particularly for timeline analysis, as discussed in this paper.

3. Proposed methodology

To assess the performance of an LLM for timeline analysis, several aspects are important as depicted in Fig. 1. We must define one or more tasks (Sec. 3.2) that we expect the LLM to perform. This involves designing a prompt to interact with the system, such as summarizing events into high-level insights or identifying indicators of compromise. In addition, a ground-truth dataset is needed that can be used to assess the outcome of an LLM (Sec. 3.3). Lastly, evaluation metrics are required that allow us to compare the ground truth with LLM output (Sec. 3.1). While starting with the tasks may seem natural, we recommend beginning with the evaluation metric, as it defines the required output, which in turn influences the task and prompt.

3.1. Evaluation metrics

For the evaluation, we decided to use BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation). They were selected due to their widespread acceptance and established methodologies in machine translation and summarization. These metrics provide a way to quantify the quality of generated text and allow for comparisons across different models and tasks.

3.1.1. BLEU – Bilingual Evaluation Understudy

BLEU assesses the quality of machine-generated outputs by comparing them to human-curated reference texts (ground truth) (Papineni et al., 2002). The score focuses on how accurately and completely the machine or LLM has replicated the human ground truth. It is calculated as follows:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (1)$$

where p_n is the precision for each n-gram, w_n is the weight for each n-gram, and BP is the brevity penalty (BP). BP is designed to penalize generated text that is too short. The idea is that shorter text might artificially increase precision, but may not capture the full meaning of the original text. The brevity penalty is calculated as:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \quad (2)$$

where c is the length of the candidate (machine) translation and r is the reference length.

3.1.2. ROUGE – Recall-Oriented Understudy for Gisting Evaluation

ROUGE is a collection of metrics designed to evaluate automatic summarization and machine translation systems (Lin, 2004). It primarily focuses on the quality of the output generated by these systems. In our case, the essence of ROUGE is to provide a quantitative measurement of the quality of an automatically generated text from an LLM by comparing it with reference data or ground truth created by humans.

ROUGE includes several metrics, each serving a unique purpose in evaluating text. Two of the key metrics are ROUGE-N and ROUGE-L. ROUGE-N assesses the overlap of n-grams between the machine-generated text and the reference, where n is 1 and 2 in our experiments. ROUGE-L focuses on the longest common subsequence (LCS) between the LLM-generated n and the reference.

ROUGE-N is based on the n-gram overlap between the machine-generated text and the reference as follows:

$$\text{ROUGE-N} = \frac{\sum_{s \in \{\text{Reference}\}} \sum_{n\text{-gram} \in s} \text{Count}_{\text{match}}(n\text{-gram})}{\sum_{s \in \{\text{Reference}\}} \sum_{n\text{-gram} \in s} \text{Count}(n\text{-gram})} \quad (3)$$

where $\text{Count}_{\text{match}}(n\text{-gram})$ is the count of n-grams in the machine-generated text that matches the ground truth. $\text{Count}(n\text{-gram})$ is the count of n-grams in the ground truth. On the other hand, ROUGE-L evaluates the LCS between the machine-generated text and the reference as follows:

$$\text{ROUGE-L} = \frac{\sum_{s \in \{\text{Reference}\}} \text{LCS}(s, \text{Machine})}{\sum_{s \in \{\text{Reference}\}} \text{Length}(s)} \quad (4)$$

where $\text{LCS}(s, \text{Machine})$ refers to the length of the LCS between the system-generated text and the reference s. Finally, $\text{Length}(s)$ is the length of the reference text. For both BLEU and ROUGE, the higher the score, the better. We implemented both metrics using HuggingFace `evaluate` library (Hugging Face, 2024a,b). Note that we evaluate the text from LLM's answer that is generated in a downloadable file, not from the text-based responses.

3.1.3. Considerations

Achieving high BLEU and ROUGE scores requires a significant overlap between the LLM's output and our ground truth data, where 'overlap' means identical wording. These metrics do not assess meaning but only textual similarity. For example, the sentences 'He is 25' and 'He was born in 2000' would yield low scores despite conveying the same information. Therefore, we must ensure the LLM returns data in a specific format, which we also use in our ground truth. To achieve this, we designed tasks that are largely deterministic (solvable by traditional software) and provided examples within the prompt to guide the LLM. In the future, we plan to explore fine-tuning an LLM, which could enhance user experience. However, this study focuses on feasibility, and fine-tuning is beyond its scope.

3.2. Common tasks for forensic timeline analysis

Given the considerations and in order to quantitatively evaluate the capabilities of an LLM, we selected the following four tasks:

1. Running `grep` for specific terms, i.e., assess how well the LLM handles a straightforward task such as running `grep`.
2. Rule-based anomaly detection, i.e., looking for patterns that could also be identified using rules, such as multiple failed login attempts, could mean a brute-force attack.
3. Event summarization, i.e., combining several low-level events into a more meaningful event, such as if events A and B are found (low-level), this means a new user was created (meaningful event).
4. Exploratory data analysis.

Tasks have been carefully chosen to be realistic but also allow for validation, e.g., for running grep we can develop our own grep expression. Note that only the first three tasks require a ground truth. With respect to the prompts, we follow the prompt style of Scanlon et al. (2023b) and the OpenAI prompt engineering guides (OpenAI, 2024b). More details are provided in the subsequent sections.

3.2.1. Prompts for running grep of specific terms

This task simulates a simple `grep` command to ensure that it can handle basic tasks without making critical errors. The example prompts are shown below:

1. "I am a forensic investigator. I need to find these terms: `\b[A-Za-z0-9_\\:.]+\.exe\b` in the given CSV file to get all entries related to executable files (.exe). The CSV file is a forensic timeline generated from the log2timeline/Plaso tool."
2. "For your references, the grep command is: grep -E "`\b[A-Za-z0-9_\\:.]+\.exe\b`" timeline.csv."
3. "Do not include the first line of the file containing column names. Include all columns in the results, not only the message column. Export the results into plain text."

The prompt asks an LLM to replicate the functionality of a `grep` command, which is commonly used to search for patterns in the text. The goal is to search the CSV file for all entries that contain executable files with the `.exe` extension. In total, five terms need to be found, the system is expected to identify these entries and save the results in plain text format. In addition, we ask the system to exclude the header row and include all column values in the results. This task checks whether the LLM can effectively search through the forensic timeline using a regular expression to filter out relevant entries.

3.2.2. Prompts for rule-based anomaly detection

The goal of this task is to enable more natural queries against the timeline. This simulates providing a timeline and then asking about specific aspects, such as 'Have there been failed login attempts?' or 'Was registry.exe executed?' Rather than posing these queries one by one, we opted to include multiple elements of interest in a file (keyword list), which the user uploads. This approach effectively cross-references a keyword list with the CSV-based timeline.

Specifically, we provide the following prompt: 'I am a forensic investigator. Read this list of keywords to find suspicious events.' The user uploads a keyword list, allowing the system to focus on specific patterns or terms that may indicate abnormal or anomalous behavior within the timeline.

As we require the output in a specific format, the uploaded file is in reality a JSON file which includes elements of the prompt (`event`) as well as what to look for (`keyword`). This helps the LLM to detect suspicious events in a timeline CSV file. Note, the keyword is extracted from the message column from the timeline data, i.e., it exists in the timeline CSV. The event is our own creation.

```
{
    "event": "Registry launch with prefetch file",
    "keyword": "Prefetch [REGEDIT.EXE] was executed"
}
```

The LLM is expected to return a JSON-formatted response that includes the timestamp of the detected event (datetime), the name of the matched event (event), the keyword that triggered the match (keyword), and the full log message (message) from the timeline. This structured format facilitates automated comparison with ground truth data and supports downstream forensic analysis. This format also maintains consistency and interpretability to allow for an accurate evaluation using BLEU and ROUGE. An example output structure is shown below:

```
[{
    "datetime": "2023-12-27T00:37:14.609465+00:00",
    "event": "Registry launch with prefetch file",
    "keyword": "Prefetch [REGEDIT.EXE] was executed",
    "message": "Prefetch [REGEDIT.EXE] was executed -
    run count 3 path hints: \\WINDOWS\\REGEDIT.EXE
    hash: 0x246AC210 volume: 1 [serial number:
    0x5CE1DF5A  device path: \\VOLUME{01da182ce1985a
    64-5ce1df5a}]"
}]
```

3.2.3. *Prompts for event summarization*

A user action (high-level event) of causes many entries in a timeline (low-level events). This tasks looks at the possibility to summarize low-level tasks to high-level tasks. To solve this task without fine-tuning, we provide a code (a python library) that can be used (executed) by the LLM.

The interaction between the user and ChatGPT is outlined in Fig. 2. We provide a persona, such as stating a role (e.g., forensic investigator), including detailed information about the task (e.g., event type or data format), and offer additional tools to improve accuracy. These steps help the system to manage responses more accurately. The prompt uses a space delimiter to provide suitable spacing to separate key pieces of information.

In the third-to-last box (“Specify steps to run an event summarization”), the user outlines the exact procedure for summarizing events. This involves uploading the CSV file, selecting the type of event (such as “last-shutdown”), and executing the summarization using the given libraries. The expected return value for this task is as follows:

```
{"0": {
 "id": 1002,
 "date_time_min": "2023-12-26 00:34:47.890403+00:00",
 "date_time_max": "2023-12-26 00:34:47.890403+00:00",
 "evidence_source": "[9707 / 0x25eb] Provider identifier:
   {...} Source Name: Microsoft-Windows-Shell-Core Strings:
   ['msedge.exe\" --no-startup-window --win-session-start']
   Computer Name: WinDev2311Eval Record Number: 2249
   Event Level: 4",
 "type": "Process Creation",
 "description": "Process creation of 'msedge.exe'",
 "category": "Windows",
 "plugin": "EVT-WinEVTX-winevtx",
 "files": "NTFS:\\Windows\\System32\\winevt\\Logs\\
   Microsoft-Windows-Shell-Core%4Operational.evtx",
 "keys": {
   "Windows Event ID": "9707",
   "Windows Event ID (hex)": "0x25eb",
   "Executable name": "msedge.exe"
 },
 "supporting": { ... },
 "trigger": { ... }
} ... }
```

3.2.4. *Prompts for exploratory data analysis*

Lastly, we explore the potential of LLMs for exploratory data analysis (EDA) which allows gaining valuable insights into the dataset as a whole. For instance, EDA may help investigators grasp the structure, distribution, and key features. It may also enable the identification of patterns and relationships between events, such as how user behaviors might be interconnected. In addition, it facilitates the visualization of temporal data, which is an important aspect of timeline analysis. Using diagrams such as histograms and heatmaps, investigators can acquire a clearer understanding of trends and cycles in the data. These visualizations pinpoint periods of interest and aid in the identification of suspicious activities for further investigation.

The example EDA prompt is: “Explore patterns of event occurrences based on the datetime field per second (e.g., busiest times, significant gaps), use a bar chart. Write the hour:minute:second in the x axis”. An LLM will generate a Python code to create the bar chart, and we can download the chart as a PNG file.

3.3. *Ground truth*

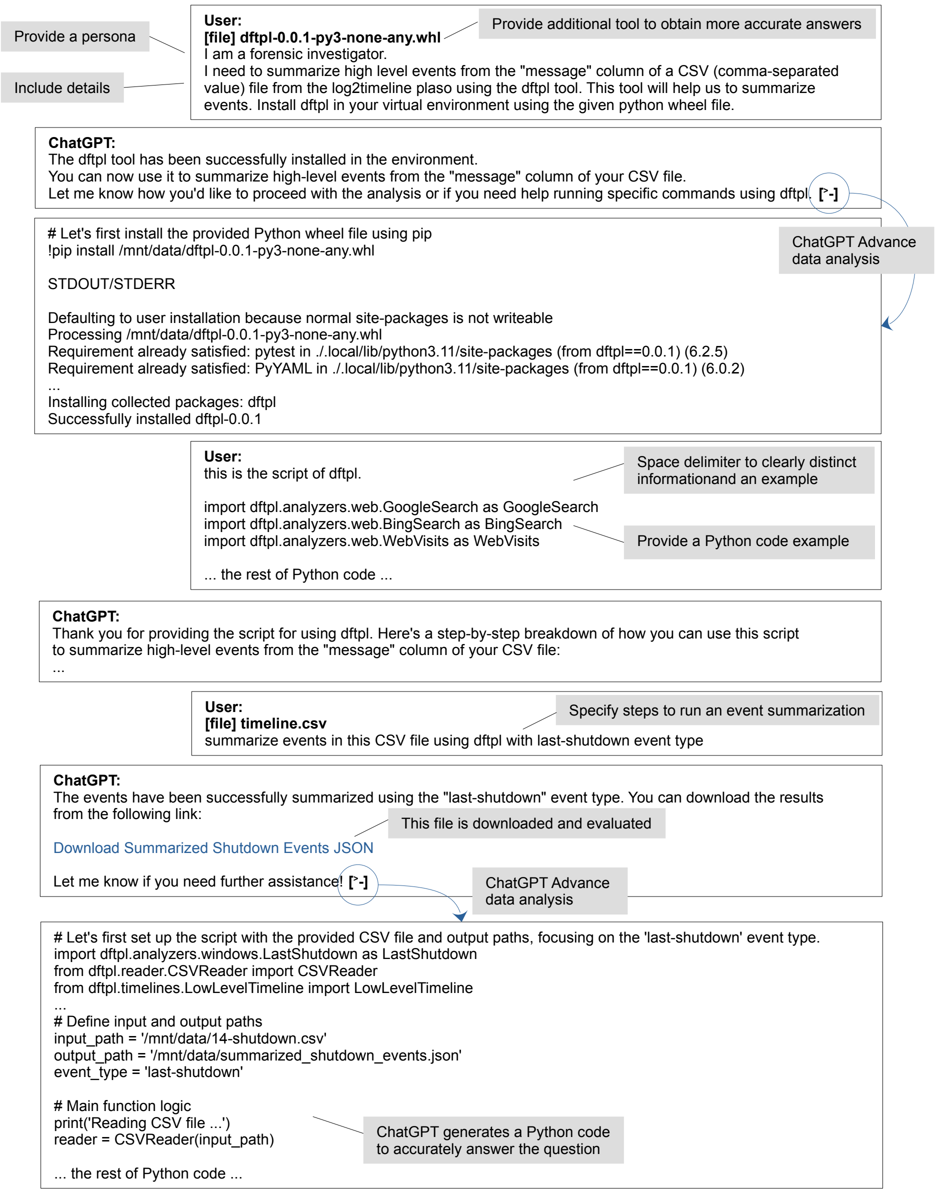
To assess the quality of output (LLM response), we require a ground truth dataset, i.e., documentation of the underlying dataset (Göbel et al., 2023; Breitinger and Jotterand, 2023). A peculiarity in our scenario is that we need the ground truth in a specific format so that it is comparable with the output of an LLM (automated). Specifically, there is no easy way to compare a disk image or its corresponding timeline against the LLM output. Consequently, the underlying dataset must be converted into a text-based format (ground truth), allowing and automated comparison with the LLM output.

To accomplish this, we first must create a dataset (Sec. 3.3.1) where the creation process is documented or recorded. Next, we generate a timeline of the disk image (Sec. 3.3.2) which serves as an input for the LLM. Lastly, using the documentation and timeline, we manually create the *expected outcome* which represents our ground truth (Sec. 3.3.5 to 3.3.3).

3.3.1. *Scenario and dataset generation*

The first step was to create a dataset as no appropriate dataset was available. The procedure is illustrated in Fig. 3 and the dataset is shared through Zenodo. Our test bed was a Windows 11 machine within a virtual environment simulating regular computer usage. All activities were recorded using screen capture (video) and therefore are documented (written notes).

The scenario follows a sequence of opening applications, downloading software, and accessing websites. The user begins by opening the Edge browser and then navigates to Bing. They perform a search query for “Mozilla Firefox download” on Bing and visit Mozilla’s official website to download the Firefox browser. After that, the user opens the File Explorer to navigate the downloaded installer. The user runs the Firefox installer and opens the newly installed Firefox browser. Afterward, they navigate to Google, perform a search related to SQL injection, and open a tutorial on the W3Schools website. The session ends with a system shutdown, indicating that the user has completed all activities.


Provide a persona
Include details
Provide additional tool to obtain more accurate answers
ChatGPT Advance data analysis
Space delimiter to clearly distinct informationand an example
Provide a Python code example
Specify steps to run an event summarization
This file is downloaded and evaluated
ChatGPT Advance data analysis
ChatGPT generates a Python code to accurately answer the question

User:
[file] dftpl-0.0.1-py3-none-any.whl
I am a forensic investigator.
I need to summarize high level events from the "message" column of a CSV (comma-separated value) file from the log2timeline plaso using the dftpl tool. This tool will help us to summarize events. Install dftpl in your virtual environment using the given python wheel file.

ChatGPT:
The dftpl tool has been successfully installed in the environment.
You can now use it to summarize high-level events from the "message" column of your CSV file.
Let me know how you'd like to proceed with the analysis or if you need help running specific commands using dftpl. [>-]

```
# Let's first install the provided Python wheel file using pip
!pip install /mnt/data/dftpl-0.0.1-py3-none-any.whl

STDOUT/STDERR

Defaulting to user installation because normal site-packages is not writeable
Processing /mnt/data/dftpl-0.0.1-py3-none-any.whl
Requirement already satisfied: pytest in ./.local/lib/python3.11/site-packages (from dftpl==0.0.1) (6.2.5)
Requirement already satisfied: PyYAML in ./.local/lib/python3.11/site-packages (from dftpl==0.0.1) (6.0.2)
...
Installing collected packages: dftpl
Successfully installed dftpl-0.0.1
```

User:
this is the script of dftpl.

```
import dftpl.analyzers.web.GoogleSearch as GoogleSearch
import dftpl.analyzers.web.BingSearch as BingSearch
import dftpl.analyzers.web.WebVisits as WebVisits

... the rest of Python code ...
```

ChatGPT:
Thank you for providing the script for using dftpl. Here's a step-by-step breakdown of how you can use this script to summarize high-level events from the "message" column of your CSV file:
...

User:
[file] timeline.csv
summarize events in this CSV file using dftpl with last-shutdown event type

ChatGPT:
The events have been successfully summarized using the "last-shutdown" event type. You can download the results from the following link:

Download Summarized Shutdown Events JSON

Let me know if you need further assistance! [>-]

```
# Let's first set up the script with the provided CSV file and output paths, focusing on the 'last-shutdown' event type.
import dftpl.analyzers.windows.LastShutdown as LastShutdown
from dftpl.reader.CSVReader import CSVReader
from dftpl.timelines.LowLevelTimeline import LowLevelTimeline
...
# Define input and output paths
input_path = '/mnt/data/14-shutdown.csv'
output_path = '/mnt/data/summarized_shutdown_events.json'
event_type = 'last-shutdown'

# Main function logic
print('Reading CSV file ...')
reader = CSVReader(input_path)

... the rest of Python code ...
```

Figure 2: A conversation sample between an investigator and ChatGPT. Note that the prompt should be accurate following the OpenAI prompt engineering guides

3.3.2. Timeline generation

To generate the timeline, we ran log2timeline/Plaso (Metz et al., 2024) on the `vmdk` file. The tool (Plaso) analyzes all known artifacts[2] and compiles them into a single unified timeline. By default, the tool processes all partitions from a `vmdk` file and generates a Plaso storage file (`*.plaso`, a database file) containing the forensic timeline. To convert `plaso` file to a CSV timeline file, we ran `psort`.

3.3.3. Ground truth for Task 1: Running `grep` for specific terms

Building the ground truth is straightforward as we only have to manually run `grep` on the dataset and take note of the output.

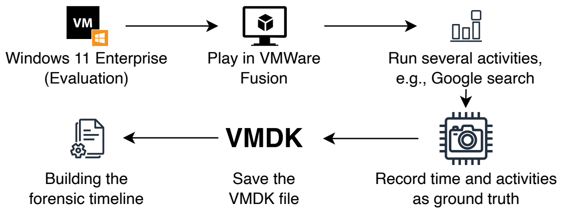

VM
Windows 11 Enterprise (Evaluation)
Play in VMWare Fusion
Run several activities, e.g., Google search
Record time and activities as ground truth
VMDK
Save the VMDK file
Building the forensic timeline

Figure 3: Building ground truth for LLM evaluation

This was done for the following five keywords:

1. `RegisteredApplications`: obtaining events related registered applications in Windows registry.
2. `(OneDrive|OneDrive\.exe)`: finding events related to Microsoft OneDrive application.

---

[2] Plaso consists of various parsers for different artifacts. Artifacts unknown to Plaso are ignored.

3. `\b[A-Za-z0-9_\\:.]+\.exe\b`: looking for all entries related to executable files (.exe).
4. `4616 /`: finding Windows event ID 4616 which related to system time change without regex.
5. `\[4616 / 0x1208\].*Microsoft-Windows-Security -Auditing.*svchost.exe`: finding Windows event ID 4616 with regex.

The command to generate this ground truth is `grep -E keyword timeline.csv`, where `-E` indicates that extended regular expressions are being used with the `grep` command.

3.3.4. Ground truth for Task 2: Rule-based anomaly detection

The second ground truth requires matching keywords (or phrases) with events. We create the keywords as a rule-based approach by first checking the date and time of the event we performed earlier in the Windows test-bed. Next, we manually look for related entries in the timeline CSV file. Once we find the relevant entry, such as registry launch, we extract the keywords linked to the event. Finally, we format these keywords into a JSON format as shown below:

```
{
    "event": "Registry launch with prefetch file",
    "keyword": "Prefetch [REGEDIT.EXE] was executed"
}
```

In the evaluation, we can ask questions in natural language because the event and the keyword the LLM searches for are already defined. Unlike an event summarization task, no script or library is provided, and the LLM handles the matching on its own. These keywords collected are a useful technique to identify suspicious events in the forensic timeline. There are seven keywords in total and the full list of keywords in JSON format is available on Zenodo.

3.3.5. Ground truth for Task 3: Event summarization

Event summarization aims at combining low-level events to obtain high-level events based as proposed by Hargreaves and Patterson (2012). Forming the ground truth was accomplished by implementing the `dftpl` tool[3] as described by the authors. Given a CSV timeline, our prototype can extract certain high-level events and return a JSON file. There are eight predefined events, grouped into three categories:

1. Web: Google search, Bing search, and web visit
2. Windows: last shutdown, process creation, and program opened
3. User activity: file download, and recent file access

We chose JSON because it is human-readable, making it easier for investigators to interpret and manually validate results. JSON also facilitates straightforward comparison with evaluation metrics due to its structured nature for efficient parsing. Moreover, its compatibility with various programming languages and tools further supports automation and quantitative evaluation in forensic analysis workflows.

To create the high-level events, we ran the `dftpl` command as follows: `dftpl -i timeline-input.csv -o summarization-output.json -t last-shutdown`, where `-i` is a Plaso CSV file, `-o` is the output (in JSON), and `-t` specifies the event of interest. The `-t` option can be omitted to summarize multiple events. The list of high-level events was then manually validated that it was correct.

A sample output is provided in Fig. 4 and includes the following high-level activities:

1. *id*: A unique identifier for the event, which is a number that differentiates this event from others.
2. *date_time_min*: The earliest possible timestamp for when the event could have occurred.
3. *date_time_max*: The latest possible timestamp for when the event could have occurred.
4. *evidence_source*: Refers to the Plaso message that provides information about the event.
5. *type*: The nature of the event, such as Google Search, File Download, or any other high-level event type.
6. *description*: A human-readable explanation or summary of the event.
7. *category*: A higher-level classification or tag for filtering or organizing events.
8. *plugin*: Identifies the Plaso plugin used to parse the source file from which the event was extracted.
9. *files*: Refers to the file(s) related to the event, such as the log file, binary file, or any other data source.
10. *keys*: Stores additional key-value pairs related to the event, such as specific attributes or metadata.
11. *supporting*: Stores a list of five low-level events before and after the main event for context.
12. *trigger*: Refers to the reasoning artifact or piece of evidence that caused the event to be recognized.

4. Experimental results and analyses

This section details the experimental settings, along with the analysis, results, and discussion of our case study.

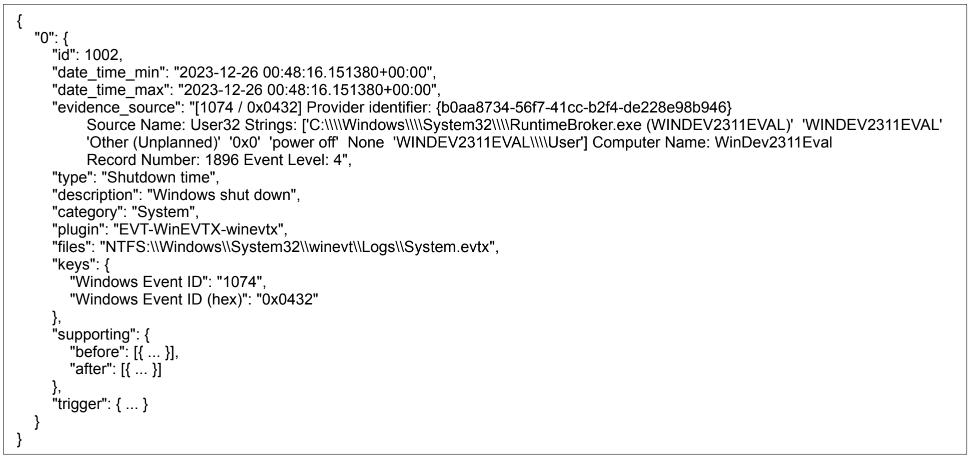
4.1. Experimental settings

We used the version of log2timeline/Plaso which was the Docker image version 20230717. The target operating system was Microsoft Windows 11 Enterprise. The OS was sourced from the Microsoft Developer Network, specifically the evaluation virtual machine (VM) version 2311 (Microsoft Developer, 2024). For virtualization, we opted for VMWare Fusion 13.5.0. For the LLM, we selected ChatGPT-4o, one of the most advanced models available at the time of writing this paper. To facilitate containerized environments, we use Docker Desktop version 4.22.1 (118664).

The extracted full timeline is too large to be handled by ChatGPT due to token limitations. Consequently, we only provided ChatGPT with about 2000 lines of Plaso entries as a timeline of interest. We have experimented with different sizes (e.g., 1000, 2000, 3000 lines) and found 2000 lines to be a manageable amount that balances input size and processing efficiency.

---

[3] https://github.com/studiawan/dftpl



```
{
  "0": {
    "id": 1002,
    "date_time_min": "2023-12-26 00:48:16.151380+00:00",
    "date_time_max": "2023-12-26 00:48:16.151380+00:00",
    "evidence_source": "[1074 / 0x0432] Provider identifier: {b0aa8734-56f7-41cc-b2f4-de228e98b946}
          Source Name: User32 Strings: ['C:\\\\Windows\\\\System32\\\\RuntimeBroker.exe (WINDEV2311EVAL)'  'WINDEV2311EVAL'
          'Other (Unplanned)'  '0x0'  'power off'  None  'WINDEV2311EVAL\\\\User'] Computer Name: WinDev2311Eval
          Record Number: 1896 Event Level: 4",
    "type": "Shutdown time",
    "description": "Windows shut down",
    "category": "System",
    "plugin": "EVT-WinEVTX-winevtx",
    "files": "NTFS:\\Windows\\System32\\winevt\\Logs\\System.evtx",
    "keys": {
      "Windows Event ID": "1074",
      "Windows Event ID (hex)": "0x0432"
    },
    "supporting": {
      "before": [{ ... }],
      "after": [{ ... }]
    },
    "trigger": { ... }
  }
}
```

Figure 4: An example of a ground truth for the event summarization task in JSON format

4.2. Timeline analysis with ChatGPT

The Advanced Data Analysis feature of ChatGPT, previously called Code Interpreter, is now integrated into ChatGPT versions 4 and 4o (OpenAI, 2024a). This feature allows users to analyze data and interpret code directly within the platform. This enhances the user experience by supporting data uploads, where users can write, test, and execute code seamlessly. The supported file formats include text, image files, PDFs and Word documents, code or other data files, as well as audio and video. In this study, we used the CSV file generated by Plaso. Once the data is uploaded, we can use the prompts to instruct ChatGPT to read or analyze the timeline.

We employ ChatGPT in two scenarios: with and without additional knowledge. In the first scenario, we provided ChatGPT with specific information related to the task, such as a library for event summarization (Sec. 3.3.5) or a list of keywords to detect suspicious activities (Sec. 3.3.4). In the latter scenario, we did not provide any additional information and relied solely on ChatGPT's existing language model to analyze the timeline.

4.3. Results and analysis

To quantitatively evaluate ChatGPT for forensic timeline analysis, we developed four tasks, including ground-truth data. For example, the event summarization task has 14 event types, the rule-based anomaly detection task has seven rules, and the search task for specific terms has five keywords. Note that the exploratory data analysis task does not have evaluation metrics because there is no ground truth data for this task.

A sample result of the given prompts and the ChatGPT answers is depicted in Fig. 2. The evaluation results for the used datasets are shown in Table 1 where the metric values represent the mean values for each task.

4.3.1. Results of running grep for specific terms

It is important to note that when asked to search for specific terms, ChatGPT does not run the `grep` command. Instead, it generates Python code to perform the search. The results of this task are shown in Table 1. The results indicate that ChatGPT performs this task effectively, especially when provided with additional knowledge, i.e., the corresponding `grep` command. Without additional knowledge, the BLEU score is 0.847, and both ROUGE-1 and ROUGE-L are 1.000. The results suggest that the system accurately identifies specific terms most of the time, but with minor variations that affect the BLEU score. With additional knowledge, the BLEU, ROUGE-1, and ROUGE-L scores all reach 1.000 and they demonstrate that the model can perfectly match the specific terms when it has more context or knowledge about the data. These findings imply that the performance of ChatGPT in conducting targeted searches is enhanced when it is given relevant prior information. Therefore, it produces consistent and fully accurate results.

ChatGPT can detect all entries correctly when provided with additional knowledge or information. However, the `grep` output from ChatGPT does not contain commas, whereas the ground truth does, as the timeline is a comma-separated file. Additionally, the model's output has extra spaces that are not present in the original data.

Furthermore, it gives inconsistent output when no additional knowledge is provided. In several cases, it only produces incomplete results, displaying only the "message" column without including all other columns. In other instances, it provides the correct values for all columns of the CSV file. When we obtained inconsistent responses, we clicked the "Refresh" button and it would generate the correct ones.

4.3.2. Results of rule-based anomaly detection

As mentioned in Sec. 4.2, there are two scenarios: one with additional knowledge and one without. In the case without additional knowledge, the prompt is slightly different because it does not include instructions to read the uploaded keywords file. In this task, we can instruct ChatGPT to format the answers in a specific format, such as JSON. The prompt would be "Format your answer using this JSON format:" and we can give an example format as follows:

```
{
  "datetime": "datetime_here",
  "event": "event_name_here",
  "keyword": "keyword_here",
```

Table 1: Evaluation results of various tasks given to ChatGPT for forensic timeline analysis

Task	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	Mean score
Without additional knowledge					
Event summarization (single)	0.077	0.192	0.129	0.136	0.134
Event summarization (multiple)	0.001	0.171	0.120	0.132	0.106
Rule-based anomaly detection	0.147	0.144	0.075	0.141	0.127
Run grep for specific terms	0.847	1.000	1.000	1.000	0.962
With additional knowledge					
Event summarization (single)	0.999	1.000	1.000	1.000	1.000
Event summarization (multiple)	0.743	0.786	0.786	0.786	0.775
Rule-based and anomaly detection	0.945	0.997	0.996	0.997	0.984
Run grep for specific terms	0.847	1.000	1.000	1.000	0.962

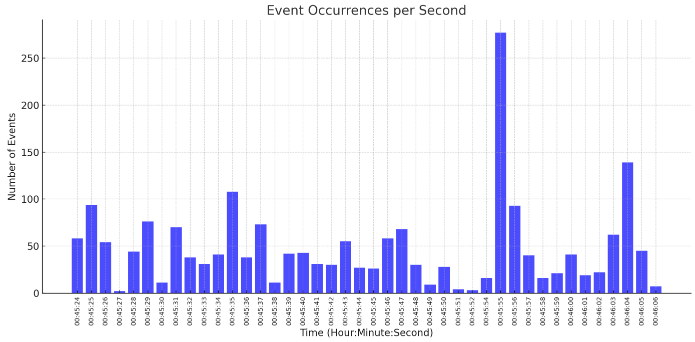


Figure 5: A bar chart generated by ChatGPT in exploratory data analysis task.

```
  "message": "message_from_logs_here"
}
```

Moreover, we instruct the system to export all results to a downloadable file, with “I need all entries of suspicious entries. Export to a JSON file for all of the results”.

In the task of rule-based anomaly detection without additional knowledge, the performance was poor: The BLEU score is 0.147, and the ROUGE scores range from 0.141 to 0.192, indicating that the model’s output is different significantly from the expected output. The keywords generated by ChatGPT are as follows: ‘delete’, ‘clear’, ‘wipe’, ‘remove’, ‘malware’, and ‘unauthorized’. These low scores reflect minimal overlap between the system’s output and the expected results, both in terms of individual words and word sequences. However, it is important to note that these evaluation metrics are based on word matching and do not account for semantic similarity. Although the wording used by ChatGPT may differ from the predefined ground truth, the underlying interpretation or intent of the result may still be forensically relevant or correct.

In contrast, the results improve when additional knowledge is provided. Specifically, the BLEU score rises to 0.945 and the ROUGE scores increase to nearly perfect values (ranging from 0.996 to 0.997). This means that generated outputs closely match the expected results. This highlights the importance of providing context or specialized knowledge to improve performance in more complex forensic analysis tasks.

Even with additional information or knowledge, ChatGPT can still make mistakes. The errors are mainly due to differences in how characters are escaped. For example, the ground truth uses two backslashes to escape regular expressions (regex), while ChatGPT’s output uses four backslashes to escape the “\” character.

4.3.3. *Results of event summarization*

Event summarization comprises two scenarios: summarizing a single event or multiple events. Summarizing a single event means the method extracts one specific event from the provided timeline, such as a Google search (full list see Sec. 3.3.5). Consequently, multiple events mean the LLM is tasked with summarizing all defined events.

Our research indicates that ChatGPT uses a virtual environment to run Python code when responding to user prompts. This means we can install the `dftpl` Python wheel installer within that virtual environment. To respond to the user prompts, ChatGPT generates Python code as shown in Fig. 2. For example, if the parser example is designed to work for all supported events, ChatGPT can summarize a specific event, such as the last shutdown event on Windows. One can click the ‘[>_]’-button to view the generated Python source code. Thus, experienced investigators may validate the code and with it the answer. Finally, the results can be downloaded in a JSON format and this file will be quantitatively evaluated based on the ground truth from Sec. 3.3.5.

The result of event summarization on single and multiple events without additional knowledge shows a low performance, with a BLEU score of 0.077 indicating limited precision in generating a summarization that closely matches the expected events. The ROUGE-1 score of 0.192 suggests that around 19.2% of single words in the generated output matched the reference, while the ROUGE-2 score of 0.129 shows even lower overlap in bigrams (two-word sequences). The ROUGE-L score of 0.136 reflects a moderate match in terms of the longest sequence of matching words. However, we conclude that without additional knowledge, the system cannot accurately summarize events.

In contrast, the result for a single event with additional knowledge, i.e., using the `dftpl` library, shows near-perfect performance, with a BLEU score of 0.999 and ROUGE-1, ROUGE-2, and ROUGE-L scores, all at 1.000. This indicates that the ChatGPT output almost exactly matched the reference in terms of precision, word overlap, and sequence structure.

The high scores suggest that, with additional knowledge, the system was able to mimic the expected results. The reason is that we gave a Python library that can summarize events based on the method described in Hargreaves and Patterson (2012) to ChatGPT (Fig. 2). Although we did not explicitly instruct ChatGPT to follow a particular order, the ground truth output produced by the `dftpl` library is chronologically ordered by timestamp. For the multiple event summarization task, the evaluation scores were lower because ChatGPT generated the correct events but in a different order than the ground truth. The beginning of the file displays timestamps that increase or remain the same, indicating a mostly sorted order. Similarly, the end of the file follows a chronological pattern. However, the middle sections break this order, with some events appearing earlier than preceding ones. This discrepancy in ordering affected the BLEU and ROUGE scores, which are sensitive to the sequence of words or structures. Importantly, while the order differed, the extracted content was sometimes semantically correct and forensically valid. Future work may include implementing order-invariant evaluation metrics or normalizing the output order before comparison to address this issue.

4.3.4. Results of exploratory data analysis

This section aims to explore how ChatGPT can assist forensic investigators in identifying patterns or anomalies within large timelines through exploratory data analysis (EDA). Specifically, we evaluate the model's ability to generate useful visualizations that support investigative tasks. The example of a generated bar chart is shown in Fig. 5. The chart shows the number of events occurrences per second within a specific time range, where each bar corresponds to a second in the format: *hour:minute:second*. The data reveal variability in event activity, with most seconds seeing between 50 and 150 events. However, there is a noticeable spike at 00:45:55, where the event count exceeds 250 which indicates a sudden surge in activity during that particular second. The concentration of events at specific seconds may point to important actions or incidents that require further investigation, especially during periods of relatively low activity that are punctuated by intense bursts (Studiawan and Sohel, 2021).

Another chart generated by ChatGPT using Python is a heatmap shown in Fig. 6. The heatmap illustrates the flow of the event sequence, showing the transitions between various types of events based on their timestamps. The rows represent the current event types, while the columns represent the next event types, with each cell indicating how often a specific event type is followed by another. The color intensity, as shown by the legend, reflects the frequency of these transitions, with darker shades showing more frequent sequences. The heatmap highlights common flows in the event timeline and provides valuable insight into which events tend to trigger others. Therefore, it can help to understand the sequences of events within the forensic timeline analysis.

Key patterns can be observed in this visualization. For example, 'Metadata Modification Time' transitions into itself 1079 times, suggesting that it frequently repeats or is followed by itself in the sequence of events. There are also noticeable transitions from 'Creation Time' to 'Metadata Modification Time' (118 times) and from 'Last Access Time' to 'Metadata Modification Time' (98 times).

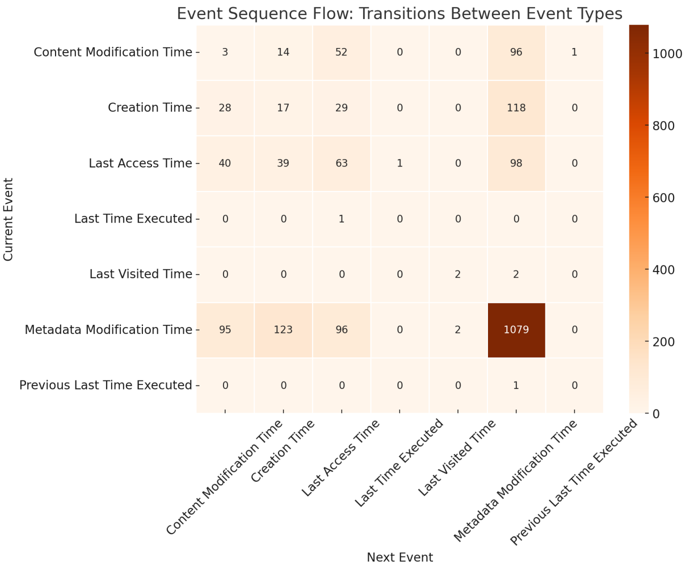


Figure 6: A heatmap generated by ChatGPT for event sequence flow

The heatmap reveals typical patterns in the event timeline by showing how certain events frequently follow others. This visualization helps investigators better understand the sequence and relationship between events during forensic timeline analysis. In short, EDA can be done by a human investigator, but using ChatGPT can help speed up this manual work.

4.4. Discussion

Overall quantitative evaluation. Without additional knowledge, tasks such as 'Event summarization (single)' and 'rule-based anomaly detection' have mean scores of 0.134 and 0.127, respectively, indicating limited accuracy. However, 'run grep for specific terms' achieves a much higher mean score of 0.962, suggesting that ChatGPT can handle search for specific terms relatively well even without prior information. With additional knowledge, mean scores improve across all tasks. Single event summarization tasks achieved a perfect mean score of 1.000, while the multiple one obtained 0.775. The results demonstrate inconsistent accuracy scores, even when provided with relevant context. The mean score for the rule-based anomaly detection task also increases to 0.984. The consistent mean score of 0.962 for "run grep for specific terms" shows that the task is already handled effectively regardless of additional knowledge. In the `grep` task, providing prior information does not lead to further improvement.

CSV file size of a forensic timeline. While ChatGPT is advertised as being capable of handling CSV files up to 50MB in size[4], we found that in practice, it struggles to process files of

---

[4] https://help.openai.com/en/articles/8983719-what-are-the-file-upload-size-restrictions

that size. Throughout our experiments, we observed that ChatGPT could successfully analyze smaller CSV files, but when attempting to work with larger files (10MB or more), the model often encountered errors or failed to provide results. This discrepancy suggests that, despite the claims in the documentation, there are practical limitations when analyzing larger datasets, likely due to resource constraints or the tokens complexity involved in processing such large volumes of data.

5. Conclusion and future work

The proposed methodology and dataset have demonstrated its potential for quantitative evaluation of timeline analysis using LLMs. Using the proposed standardized methodology and dataset, researchers can apply and expand the test and evaluation of LLM-based forensic timeline analysis. By employing the advantages of natural language processing on LLMs, e.g., ChatGPT, the presented case studies show that it can assist in analyzing events and temporal information from a forensic timeline. It also provides valuable information for forensic investigators, particularly in the task of exploratory data analysis. However, based on the quantitative evaluation, ChatGPT performs worse than a rule-based approach or a regular expression-based approach accompanied by a human investigator.

For future work, we plan to undertake the task of developing custom LLMs specifically trained on digital forensic data and associated tasks. In addition, we will explore the use of other commercial LLM services such as Google Gemini and Claude to evaluate the robustness of our approach. In addition, to address concerns about the confidentiality of digital evidence, we plan to deploy open-source LLMs, such as LLaMA (Touvron et al., 2023) and Mixtral (Jiang et al., 2024) on a local device. By keeping the forensic timeline on the local computer, we aim to avoid the need to upload sensitive data to cloud-based LLM services, thus ensuring the privacy of the investigation.

Acknowledgments

We would like to thank Christopher Hargreaves for his valuable comments and feedback.

References

Arshad, H., Jantan, A.B., Abiodun, O.I., 2018. Digital forensics: Review of issues in scientific validation of digital evidence. Journal of Information Processing Systems 14, 346–376.

Bhandari, S., Jusas, V., 2020. An abstraction based approach for reconstruction of timeline in digital forensics. Symmetry 12, 104.

Breitinger, F., Jotterand, A., 2023. Sharing datasets for digital forensic: A novel taxonomy and legal concerns. Forensic Science International: Digital Investigation 45, 301562.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. Advances in Neural Information Processing Systems 33, 1877–1901.

Brunty, J., 2023. Validation of forensic tools and methods: A primer for the digital forensics examiner. Wiley Interdisciplinary Reviews: Forensic Science 5, e1474.

Buchholz, K., 2023. One million users: Threads shoots past one million user mark at lightning speed. `https://www.statista.com/chart/29174/time-to-one-million-users/`.

Chabot, Y., Bertaux, A., Nicolle, C., Kechadi, M.T., 2014. A complete formalized knowledge representation model for advanced digital forensics timeline analysis. Digital Investigation 11, S95–S105.

Dinis-Oliveira, R.J., Azevedo, R.M., 2023. ChatGPT in forensic sciences: A new Pandora's box with advantages and challenges to pay attention. Forensic Sciences Research 8, 275–279.

Göbel, T., Baier, H., Breitinger, F., 2023. Data for digital forensics: Why a discussion on "how realistic is synthetic data" is dispensable. Digital Threats: Research and Practice 4, 1–18.

Hall, S.W., Sakzad, A., Choo, K.K.R., 2022. Explainable artificial intelligence for digital forensics. Wiley Interdisciplinary Reviews: Forensic Science 4, e1434.

Hargreaves, C., Patterson, J., 2012. An automated timeline reconstruction approach for digital forensic investigations. Digital Investigation 9, Supplem, S69–S79.

Henseler, H., van Beek, H., 2023. ChatGPT as a copilot for investigating digital evidence, in: Proceedings of the Third International Workshop on Artificial Intelligence and Intelligent Assistance for Legal Professionals in the Digital Workplace, pp. 58–69.

Horsman, G., 2019. Tool testing and reliability issues in the field of digital forensics. Digital Investigation 28, 163–175.

Hugging Face, 2024a. Metric: bleu. `https://huggingface.co/spaces/evaluate-metric/bleu`.

Hugging Face, 2024b. Metric: rouge. `https://huggingface.co/spaces/evaluate-metric/rouge`.

Hughes, N., Karabiyik, U., 2020. Towards reliable digital forensics investigations through measurement science. Wiley Interdisciplinary Reviews: Forensic Science 2, e1367.

Inglot, B., Liu, L., 2014. Enhanced timeline analysis for digital forensic investigations. Information Security Journal: A Global Perspective 23, 32–44.

Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.d.l., Hanna, E.B., Bressand, F., et al., 2024. Mixtral of experts. arXiv:2401.04088 .

Lin, C.Y., 2004. ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, pp. 74–81.

Metz, J., Gudjonsson, K., White, D., et al., 2024. log2timeline Plaso: Super timeline all the things. `https://github.com/log2timeline/plaso`.

Michelet, G., Breitinger, F., 2024. ChatGPT, Llama, can you write my report? An experiment on assisted digital forensics reports written using (local) large language models. Forensic Science International: Digital Investigation 48, 301683.

Microsoft Developer, 2024. Get a Windows 11 development environment. `https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/`.

Nalawade, A., Bharne, S., Mane, V., 2016. Forensic analysis and evidence collection for web browser activity, in: 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), pp. 518–522.

NIST, 2019. Computer Forensics Tool Testing Program (CFTT). `https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt`.

OpenAI, 2024a. Data analysis with ChatGPT. `https://help.openai.com/en/articles/8437071-data-analysis-with-chatgpt`.

OpenAI, 2024b. Prompt engineering. `https://platform.openai.com/docs/guides/prompt-engineering`.

Palmbach, D., Breitinger, F., 2020. Artifacts for detecting timestamp manipulation in ntfs on windows and their reliability. Forensic Science International: Digital Investigation 32, 300920.

Papineni, K., Roukos, S., Ward, T., Zhu, W.J., 2002. BLEU: A method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318.

Scanlon, M., Breitinger, F., Hargreaves, C., Hilgert, J.N., Sheppard, J., 2023a. ChatGPT for digital forensic investigation: The good, the bad, and the unknown. Forensic Science International: Digital Investigation 46, 301609.

Scanlon, M., Nikkel, B., Geradts, Z., 2023b. Digital forensic investigation in the age of ChatGPT. Forensic Science International: Digital Investigation 44, 301543.

Studiawan, H., Ahmad, T., Santoso, B.J., Pratomo, B.A., 2022a. Forensic timeline analysis of iOS devices, in: 2022 International Conference on Engineering and Emerging Technologies (ICEET), pp. 1–5.

Studiawan, H., Ahmad, T., Santoso, B.J., Shiddiqi, A.M., Pratomo, B.A., 2022b. DroneTimeline: Forensic timeline analysis for drones. SoftwareX 20, 101255.

Studiawan, H., Sohel, F., 2021. Anomaly detection in a forensic timeline with deep autoencoders. Journal of Information Security and Applications 63, 103002.

Studiawan, H., Sohel, F., Payne, C., 2019. A survey on forensic investigation of operating system logs. Digital Investigation 29, 1–20.

Studiawan, H., Sohel, F., Payne, C., 2020. Sentiment analysis in a forensic timeline with deep learning. IEEE Access 8, 60664–60675.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al., 2023. Llama: Open and efficient foundation language models. arXiv:2302.13971 .

Wickramasekara, A., Breitinger, F., Scanlon, M., 2025. Exploring the potential of large language models for improving digital forensic investigation efficiency. Forensic Science International: Digital Investigation 52, 301859.