Analyst-In-Loop LLM Systems for Forensic Timeline Analysis Assistance

Oscar J. Rodriguez Ruiz
Master in Computer Science
Advisor: Jeffrey Duffany, Ph.D.
Polytechnic University of Puerto Rico
Graduate Project EXPO February 2026

Abstract *— The increasing volume and complexity of digital evidence in digital forensic investigations of today have made manual timeline analysis not only inefficient but a reckless waste of resources and effort. Tools such as Plaso (Log2timeline) have shown to be highly effective at creating "super-timelines" that gather information from various sources. Creating datasets spanning thousands of events, which can be actively considered "noise" for the forensics examiner. However, within the past years, we have made great strides in the field of artificial intelligence. These allow for the utilization of the processing power of the neural networks to assist with the process of detecting anomalies and filtering for essential parts of an investigation. Through extracting general outputs with Plaso as a CSV file, we can utilize said outputs to highlight the role that private AI models will begin to take within the field after considerable training with the available forensic training data.*

Keywords *— Digital Forensics, Event Reconstruction, LLM, Plaso Supertimeline.*

INTRODUCTION

Throughout the years, technology has evolved at a worldwide scale, integrating itself into every aspect of our current lives. Whether it is to look up the time, play a simple game, or look up information on the internet, these devices that help facilitate our current lives have become crucial pieces of information that store data on everything we utilize them for. Our current use of digital forensic technology focuses around eight aspects: basic theory and methods, physical equipment and forensic methods, image forgery identification, file recovery and data extraction, smartphones and social network forensics, case-based forensics and crime forensics, automatic identification technology and tools, cloud computing and cloud forensics [1][2]. Each one of these fields has advanced since the beginning of their conception, and alongside them, our ability to comb and analyze data at all levels has grown drastically. We now rely on computers with the ability to learn and adapt new strategies and understanding. The ability to comb through hundreds of thousands of points of data in record time is in our grasp. People throughout the world are now using it to establish more efficient and active practices than ever. Digital forensics and Incident Response have thus started to adapt their tools in order to keep up with the changes. Investigations on the possible utilization of LLMs have surged as to whether these tools could become a practical part of our current forensic systems [3]. The current work that is being presented in this report aims to showcase an LLM-based Analyst-In-The-Loop system that is meant to facilitate the understanding of supertimelines developed with tools such as Log2Timeline. Previously, this has been facilitated through the integration of UI elements with tools such as Autopsy and Timeline Explorer. This project decides to take our currently available system and implement an Analyst-In-Loop Process that would facilitate this understanding in a similar fashion to the available UI elements.

BACKGROUND

Digital forensics has evolved from a supplementary measure in investigations to a critical, multi-faceted discipline that covers areas from storage media and mobile devices to cloud computing. As the complexity of digital crimes increases alongside growing storage capacities, the need for automated event reconstruction has become paramount.

Timeline analysis is a fundamental task of any investigation involving a digital forensic examiner.

It has become essential to determine precisely what events took place within a system. You can always look over your file system for the Modified, Accessed, and Created Timelines, which are what systems such as the Sleuth Kit and EnCase attempt to create. However, more specialized systems starting in 2009 with Cyber Forensic Time Lab or CTFL by Olsson and Boldt also started gathering metadata from locations such as EXIF data, and Windows Registry [4]. One of these systems is log2timeline, invented by Kristinn Guðjónsson back in 2010 [5]. These systems have advanced and become staples of the digital forensic investigator's arsenal. However in the attempt to make understandable systems that do not miss crucial pieces of information we have also created systems that take in every single piece of information and can take days to filter through one disk only for us as investigators to have to look over the information from the timeline and for new investigators to be completely lost with the amount of events that they have to understand. At the time of this software's invention, data storage was not as advanced as it is now. Back in 2010, the majority of computers had smaller SSDs that held up to 256 GB at the high end. Today, regular laptops you can find on the open floor of a convenience store tend to have that much storage at the low end. Most home computers sport drives of around 2 or 4 TB, and that is if they aren't using multiple drives. Fifteen (15) years ago, it was much more manageable for a forensic investigator to comb through events manually. However, today's technology environment has expanded the amount of data that is more commonly available.

PROBLEM

The primary challenge currently facing the field of digital forensics is the staggering volume of data that must be analyzed during an investigation. This "data explosion" is driven by increasing storage capacities, a higher number of personal digital devices, and the inherent complexity of modern operating systems. As a result, forensic bureaus are facing significant investigation backlogs. In the case of a basic disk image today holding terabytes of data, it can be quite labor-intensive.

To address these challenges, investigators often utilize "super timelines"—comprehensive records that combine all available log file information and system events into a single chronological view. While tools like **Log2Timeline (Plaso)** are exceptionally powerful at extracting these artifacts, they often produce several million "low-level" events for a single disk. While these events are correctly placed within the systems, they tend to lack context for the investigation. This main problem defines the three major gaps that can be seen with our current investigative methods, which can be greatly helped through the use of AI systems. The first of these gaps is the human scalability problem, as the number of events on a timeline often exceeds what we can fully analyze; we require the ability to effectively reduce data to only the essential pieces. The second gap is the semantic gap, which, with more understanding and clear language, can help newer and less technical analysts understand the files they are working on and how these pertain to the investigation or system as a whole. The third gap has been worked on in various cases, and that is the tooling gap. People utilize and have developed many different systems for viewing and clearly explaining the info available for Forensic timelines. Tools like Autopsy, Timeline2GUI, and Timeline Explorer have made it easier to explore timelines than ever before due to their ability to give you a simple user interface that is easy to read through based on the forensic image you are utilizing [6].

EQUIPMENT AND MATERIALS

The equipment and material used for this project are substitutable; therefore, in order to focus on the relevance of the project's scope, we will focus upon the software components utilized for this project. However, the main workstation came with 32 GB of RAM, an Intel Core i9-10900k CPU,

and an AMD Radeon RX 6900 XT, which accounts for 16 GB of VRAM. All the software that was utilized can be used within multiple operating systems to achieve similar results. This project specifically utilized Ubuntu 24.04.03 in order to run Log2timeline(Plaso) from the command line for creating supertimelines and the csv. These timelines were then processed utilizing scripts that involved the LLM API. The LLM utilized is OpenAI's, which utilizes the "gpt 4.1 mini" model, as its high throughput and lower latency allow for a better system in the case of this project. The system was also set to utilize low creativity to account for the factual nature required from forensic investigations. Other API can be utilized to a similar effect. My recommendation is Gemini's API utilizing the "Gemini-2.5-flash" model, which is comparable to the "gpt 4.1" model and was run comparatively during the initial project until ultimately choosing OpenAI due to their rates. In order to implement these into scripts, Python 3.10.11 was used within the Visual Studio Code environment. This facilitated scripting and the utilization of any API, libraries, or dependencies, including pandas and OpenAI. The primary data sources for this project were obtained from the Digital Corpora websites. More specifically, the utilized sources include the "nps-2009-casper-rw" NPS Test Disk Images [7]. This image was picked due to being designed for the testing of Forensic Tools and its lack of Personal Identifiable Information.

METHODOLOGY

This project involved a pipeline designed to ingest forensic data from an E01 file and output summaries based on that forensic data. The process was divided into data acquisition, pre-processing, and LLM analysis. When it comes to a regular E01 file, the base structure of it can be analyzed and separated based on the data structure found. As seen in the program Autopsy here in Figure 1, we can find one of our test images (nps-2009-casper-rw), which, for the record, is about 300 MB, has quite a lot of files, and within them, we can find that they have all been split based on their current standing within the image. Autopsy includes all sorts of information from web history and searches to deleted and recovered files.

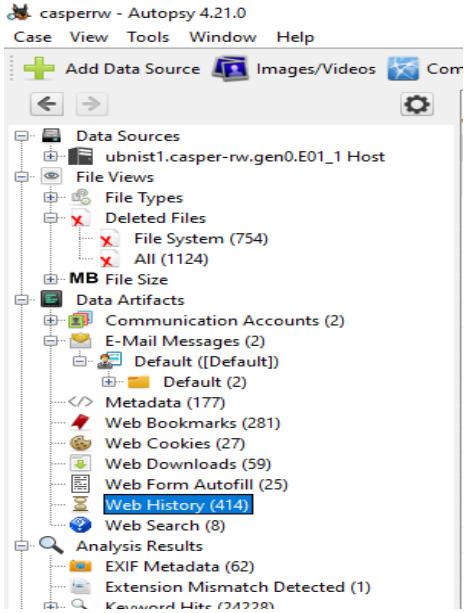

Figure 1
Autopsy Casper-rw Case Directory

However, what this project focuses on is creating a read/able timeline assistance file. Autopsy does a good job of showcasing a timeline that you can edit and look through by yourself, but without proper knowledge, you would not be able to understand what it means. The visualization of the timeline of the same disk image seen in Figure 1 is showcased here in Figure 2. This timeline holds information about what events happened and can even be specified to the day and date. You can even specify events and use different viewing methods to look over and parse through the disk.

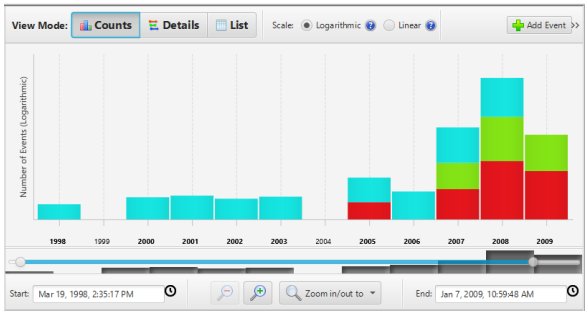

Figure 2
Autopsy Casper-rw Timeline

For a skilled forensic interpreter who understands the scope of his case, this tends to work quite well, as he can look over the type of events and details that he is tasked with searching for. An untrained mind in the field of forensics would find this an exhaustive search that not only messes with their head but also wastes time going over multiple unneeded years of information, since this disk goes back to 1998.

Autopsy actually uses Plaso/Log2timeline to create this timeline of the disk. For data acquisition, this project utilizes the same system from the command line to process the E01 file using Log2Timeline's psteal.py command to generate a much more parsable csv file. In the field, it would make more sense to utilize filters and parsers available from the program; we want to simulate a basic amount of knowledge. This command allows for the direct creation of a csv file highlighting all the events and actions taken within the image, similarly to the timeline in Figure 2. This leaves us with a csv similar to the one found below in Figure 3. Figure 3 includes multiple fields; however, usually this would span thousands of lines due to the amount of events available. These create a base for the parsing and understanding power of the LLM.

datetime	timestamp_desc	source	source_long	message	parser	display_name	tag
2008-12-28	Content Modificat	FILE	File stat	EXT:/lost+fo	filestat	EXT:/lost+found	-
2008-12-28	Last Access Time	FILE	File stat	EXT:/lost+fo	filestat	EXT:/lost+found	-
2008-12-28	Metadata Modifica	FILE	File stat	EXT:/lost+fo	filestat	EXT:/lost+found	-

Figure 3
CSV File from Casper-rw.gen0 Image

This csv is then moved over to the same directory as the ForensicAssistant.py scripts. Figure 4 below showcases the directory structure within Visual Studio. The .env file holds the API keys used to send prompts to the LLM. The Forensic Assistant file processes the information. The txt files are where the LLM's answer is placed after careful analysis. As you can see below, this can be done for multiple LLMs; however, the restrictions those LLMs have indicate how limited your understanding of the data could be. For this project, OpenAI was the main focus. There was limited testing done with Gemini that showcased it could be utilized within their current structure limits the ability of the system due to the limited amount of RPD or requests per day their models employ.

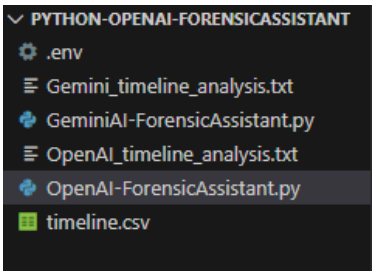

Figure 4
Visual Studio Directory

The script in this project is but one of the many steps that are needed in order to feed information to an LLM. We need one more thing when communicating with an LLM, and that is the construction of the query. In order to achieve the results we desire, we must be specific with the available query to make sure that the system understands its role when looking through this information. The structure of the query utilized for the AI system is demonstrated in Figure 5. These scripts can and should be edited for the purposes of the investigator. More specifically, different investigations will require the usage of different prompts and rulings based on the defined structure of the files and investigation. Due to the nature of the nsp-casper-rw image, the prompt utilized focused on setting criteria based on the user activities, web artifacts, and web history. The criteria section showcasing this can be seen in Figure 5 below. This section highlights important things that I, as the user, wanted to prioritize when it came to the analysis. This actively plays into the input of the LLM program, giving it the base idea for what to analyze and parse the csv for.

```
CRITERIA = """
- Focus on explicit user-initiated actions
- Prioritize WEBHIST, OLECF, and FILE artifacts
- Emphasize activity in user directories (e.g., /home/ubuntu)
- Ignore baseline system artifacts unless user-accessed
- Ignore entries marked as 'Not a time'
- Highlight file downloads from browsers
"""
```

Figure 5
Criteria of the LLM Prompt

When feeding information to AI, it is necessary that we separate the information into smaller, more digestible sections. In order to achieve this, the Python script used in Figure 6 takes the csv file and, utilizing the pandas library, determines how many chunks of 500-line digestible pieces can be made from it. This information is then used to determine how many chunks will be sent to the LLM. This information from the CSV tends to be quite extensive for certain images, as many low-priority events appear in the csv files. Thus, this helps stay within the API usage restrictions.

```
df = pd.read_csv(CSV_PATH)
total_rows = len(df)
total_chunks = math.ceil(total_rows / CHUNK_SIZE)

print(f"[+] Loaded CSV with {total_rows} rows")
print(f"[+] Processing in {total_chunks} chunks of {CHUNK_SIZE} rows")
```

Figure 6
CSV Chunks

Utilizing a shortened example file that I made based on the "nps-2009-casper-rw" file, here we can see an expected output of the OpenAI-Forensicassistant.py in Figure 7. Here we can see that during 2007-06-01 to 2007-06-15, the user of the computer had done some web browsing, specifically utilizing the browser Firefox and bookmarking the site used. However, even if there is a strong correlation to what we are looking for, it still requires further investigation because, like any ethical assistive system, it should not make assumptions and come up with results for the investigator.

```
### 2007-06-01 to 2007-06-15: User Web Browsing

#### Firefox Browsing and Bookmark Activity (User
- **2007-06-01 08:45:06 UTC**: The Firefox histor
  - *Significance*: This may indicate browsing a
  - *Confidence*: High
  - *Corroboration*: Further review of the datab

- **2007-06-06 11:38:27 UTC**: Bookmarks for `ht
  - *Significance*: These entries are consistent
  - *Confidence*: High
```

Figure 7
Output File Text Example

RESULTS & DISCUSSION

The project experimentation revealed that the system's ability to ingest and filter the forensic timelines and their generated data was astonishing and helpful. What normally would have taken hours of time was summarized with minor effort in anywhere from 1 to 3 minutes per chunk of 500 lines. While the original disk image can be parsed through with Autopsy, this tool has a better way to explain the possibly important sections of each chunk and how they can relate to your investigation. As well, the original timeline, which was around 60,000 lines of information, was cut down to a tenth of the total amount of lines and set up as natural language, which is easier to understand.

The implementation of the Python middleware utilizing pandas to reduce the volume of the supertimeline into digestible and important information allowed for a more formal system that mediated token consumption on the API (OpenAI). The system excelled at maintaining a chronological order across the chunks of log entries fed to it and was able to establish summarized instructions and understanding of the happenings within the CSV. OpenAI's GPT 4.1 mini was able to provide interpretable answers that highlighted the differences within each entry and allowed for a better understanding of the information given as input. This system category takes these large files of data and converts them into a bridge for those who are not as technically adept and are still attempting to understand the scope of the information in question, as well as allowing them to understand tools like autopsy better through the use of natural language.

The system did excel at handling the large file base, and while it did take some time to fully process the entirety of the images, the 2GB USB image took around 5 minutes to fully process. It was able to understand and follow along with the context even after multiple chunks. However, as the files get larger the risk of losing context between chunks increases, as well as the risk of

hallucinations. Future iterations of this project should explore the utilization of Retrieval-Augmented Generation, which would allow the AI to query the dataset rather than reading it linearly. The chunk system, while easier to implement, lacks a certain complexity that would be more suitable for a system targeted towards forensic use.

While these systems seemed reliable and very useful, they were, however, still prone to hallucinations. Cutting the creativity of the AI actively kept the structure of the outputs similar to an expected analytical take. However, the possibility of hallucinations is the main marker for creating an Analyst-in-Loop system and keeping the creativity of the system to a minimum. These summaries created by the AI can be reliable leads in an investigation, but they cannot be treated as absolute truths, as one of the more important parts of maintaining forensic integrity is to maintain provenance of the data. This system can therefore be utilized alongside Autopsy to pinpoint areas to investigate within the disk based on our stated criteria. In the case of the nsp-casper-rw, we can narrow down the investigation of the disk to look at the Web history of the user and work from there. We can confirm data from the summaries that can be very important towards the investigation, such as major downloads, dates, and time as well as web access if necessary. When viewing the information within the files, one of the most clear showcases of this is when looking for information on downloaded files. In the case of the investigation, the Python script was able to correctly pinpoint sources where various US websites, including the SSA, FCC, and NIST, among others, had been visited and information from those websites was downloaded, including multiple important pictures, PDFs, and docs. Other important government websites can be seen within the visited and marked sections as well. While Autopsy can easily and accessibly set up the viewing window for this information, if I had no information on the case, the utilization of the Forensic Assistance LLM would allow for a clearer understanding based on the natural language. A great example of this can be seen within Chunk 26, showcasing multiple downloads from a government website (fcc), shown in Figure 8.

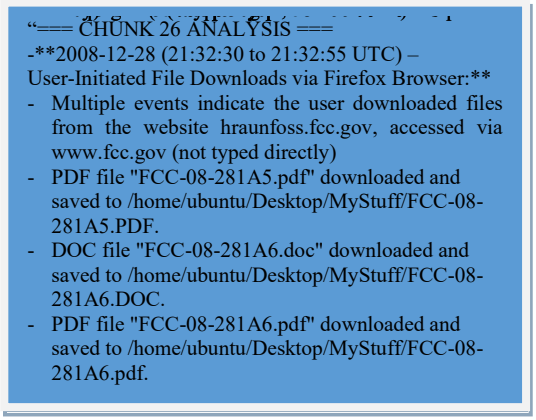

Figure 8
Chunk 26 Output

If opened within Autopsy, you would be able to confirm both the date that this data was downloaded as well as the downloads themselves within the download tab, which would showcase what can be seen in Figure 9. The exact downloads that the chunk analysis describes were done during 2008-12-28.

Figure 9
Chunk 26 Output

While the lack of natural language in Figure 9 might not be a problem for most veterans of the field, most students starting to use these systems find Chunk 26 more accessible, and understanding when it comes to forensic artifacts can dictate the difference between a rightful judgment and a butchered court case.

One part that is lacking is the entirely ethical implementation of the system. A system such as this sends information across the internet to LLMs, which can be catastrophic for the sensitive information withheld in most forensic cases today. There is a major risk of confidential information being stolen or extracted through these calls. This ultimately demotes the tool substantially, which is clear from the disclaimers presented when utilizing

the tool. This disclaimer is clearly stated within Figure 10 and showcases that a tool of this caliber would require more proprietary information and cautiously prepared measures before being utilized on its own within the field.

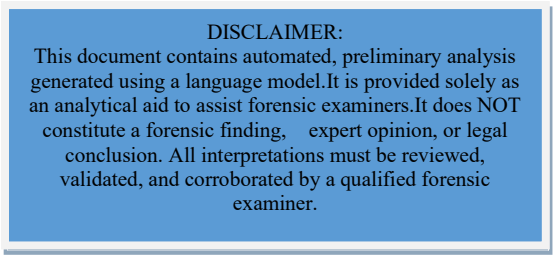

Figure 10
Disclaimer

CONCLUSION

This project demonstrated the significant potential of leveraging Large Language Models to address the persistent gaps that exist within digital forensic timeline analysis today. While tools are exceptional at extracting important and noteworthy low-level artifacts, the sheer volume of data that is stored and can be extracted can be as blinding as a sudden flashing light. By developing a Python-based middleware integrating popular LLMs (such as Gemini or OpenAI), this study successfully automates the transformation of Disk Images into a natural language format that can facilitate the work of an analyst. This also establishes a process for utilizing LLMs as an assistive tool for either studying or practicing supertimeline analysis.

While the use of LLMs within this investigation is far from actually replacing an expert in the field, they can actively serve as a powerful force multiplier. By reducing cognitive load, these tools may allow users to make more high-level decisions and have faster response times during incidents. Larger context windows, more sophisticated reasoning, and their continuous integration into systems across the globe will become an essential part of modern digital evidence extraction.

FUTURE WORK

The current implementation shows a feasible framework for utilizing a Large Language Model to synthesize a simplified version of complex supertimeline outputs. However, there are still areas of investigation that need to be explored and enhanced when it comes to this field, especially when it comes to establishing reliable, scalable systems for deeper analysis. Noise still continues to be a problem when it comes to the standard supertimeline. One valid way to remedy this is by exploring techniques for Forensic Data Reduction. Whitelisting things like data telemetry could save on token costs in the long run, and Dynamic Feature selection could be used to optimize the model's focus when looking for specific data.

While we can successfully identify patterns and include certain insights within the investigation, this automation is still based upon systems that are prone to hallucinations when given too much data. As LLMs continue to advance and become more refined, this problem will become less prominent in future cases. There is also the possibility of creating specially trained systems in-house that can actively learn and work together with the forensic networks available.

The comparisons made within this study are more lenient and specific towards the LLMs, not taking into account the observations that a human being made upon the same data. Future work could aim to evaluate under those criteria future LLMs with systems like ROGUE and BLEU to verify the system's understanding against that of a human analyst [8].

The chunk system implemented in this project feeds the entire CSV worth of raw data to the API within multiple prompts, which ultimately takes time and is inefficient if we want to use more natural language within the query. A solution to this system may utilize Retrieval-Augmented Generation, also known as RAG, to create an external trusted data source based not only on the CSV used but also on other Digital Forensic material, which can provide up-to-date and context-

aware responses that will be more well-informed due to the data to back up the system's claims. Other projects have also gone about implementing RAG, and others have even housed their own LLM system based on Llama [9]. Either of these methods will help avoid the dreaded hallucinations that are common when using LLMs.

REFERENCES

[1] Atlam, H. F. (2025). *LLMs in Cyber Security: bridging practice and education. Big Data and Cognitive Computing*, 9(7), 184. [Online] Available: https://doi.org/10.3390/bdcc9070184.

[2] Jiang, G., & Li, C. (2019). *A Scientometric Review of Research Evolution in Digital Forensics.* Proceedings of the 3rd International Conference on Computer Science and Application Engineering, 1–11. [Online] Available: https://doi.org/10.1145/3331453.3362055.

[3] 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. [Online] Available: https://doi.org/10.1016/j.fsidi.2024.301859.

[4] Olsson, J., & Boldt, M. (2009). *Computer forensic timeline visualization tool.* Digital Investigation, 6, S78–S87. [Online] Available: https://doi.org/10.1016/j.diin.2009.06.008.

[5] Hargreaves, C., & Patterson, J. (2012). *An automated timeline reconstruction approach for digital forensic investigations.* Digital Investigation, 9, S69–S79. [Online] Available: https://doi.org/10.1016/j.diin.2012.05.006.

[6] Debinski, M., Breitinger, F., & Mohan, P. (2018). *Timeline2GUI: A Log2Timeline CSV parser and training scenarios.* Digital Investigation, 28, 34–43. [Online] Available: https://doi.org/10.1016/j.diin.2018.12.004.

[7] Disk Images – Digital Corpora. (n.d.). [Online] Available: https://digitalcorpora.org/corpora/disk-images/.

[8] Studiawan, H., Breitinger, F., & Scanlon, M. (2025). *Towards a standardized methodology and dataset for evaluating LLM-based digital forensic timeline analysis.* Forensic Science International Digital Investigation, 54, 301982. [Online] Available: https://doi.org/10.1016/j.fsidi.2025.301982.

[9] Sharma, B., Ghawaly, J., McCleary, K., Webb, A. M., & Baggili, I. (2025). *ForensicLLM: A local large language model for digital forensics.* Forensic Science International Digital Investigation, 52, 301872. [Online] Available: https://doi.org/10.1016/j.fsidi.2025.301872.