Forensic Science International: Digital Investigation 56 (2026) 302053

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS EU 2026 - Selected Papers from the 13th Annual Digital Forensics Research Conference Europe

Addressing the dataset gap problem with Generative AI: Towards LLM-driven forensic scenarios for dataset generation

Michael Plankl [a], Thomas Göbel [b,*], Harald Baier [b,1]

[a] Friedrich-Alexander University, 91058, Erlangen, Germany
[b] Research Institute CODE, University of the Bundeswehr Munich, 81739, Munich, Germany

ARTICLE INFO

Keywords:
Data synthesis
Data generation
Data synthesis framework
Synthetic data
Dataset gap problem
Generative AI
GenAI
Large language model
LLM
GPT
Gemini

ABSTRACT

The increasing amount of incriminating data to be analysed on the one hand and the limited availability of forensic datasets on the other hand complicate forensic research as well as the development and validation of forensic tools. This challenge is often referred to as the *dataset gap problem*. A novel and promising approach to solve the dataset gap problem is the generation of synthetic, forensic scenarios through the application of Generative AI (GenAI) approaches like Large Language Models (LLMs). In this paper, we demonstrate how to use popular, general-purpose foundation models to generate various forensic artefacts. While emphasising the benefits of an LLM-driven dataset generation, we also address in detail inherent risks that can impair data synthesis using LLMs (e.g., hallucinations, limited explainability, stochastic model behaviour) and show how to compensate for these limitations (e.g., skilful use of prompt engineering and architectural patterns such as function calling and AI agents). In addition, we prove the practicability of our approach by enhancing a recent data synthesis framework with LLM capabilities and a user-friendly interface. Consequently, we are able to use GenAI to automatically generate configuration files for various forensically coherent scenarios and the resulting datasets. Our implementation thus demonstrates the potential of an automated, prompt-driven scenario generation process, thereby presenting a scalable solution to the shortage of forensic dataset availability.

1. Introduction

The use of Generative AI (GenAI) and Large Language Models (LLMs) opens up new possibilities in digital forensics (DF) by automating the investigative process, accelerating and optimising the search for evidence and facilitating the creation of forensic reports (Scanlon et al., 2023; Michelet and Breitinger, 2024). This is particularly true since most phases in DF still involve manual processes that require significant human effort and time and are often subject to changing requirements. Despite these advantages, according to Breitinger and Jotterand (2023), the use of synthetic data generated by artificial intelligence (AI) is not yet widespread in the DF community, where synthetic data refers to generation of traces using software with a certain degree of autonomy.

However, there is a consensus in the forensic community that both real-world *and* synthetic data are needed in the DF, as real-world data is usually rarely available due to data protection and law enforcement restrictions (Göbel et al., 2023). Since datasets are urgently needed in DF (e.g., for education and practitioner training, for tool testing, and in research (Göbel et al., 2025, 2023; Horsman and Lyle, 2021)), it is likely that we will soon see many more artificially generated traces and artefacts in DF. The generation of synthetic datasets is already being automated using data synthesis frameworks (Göbel and Baier, 2025; Göbel et al., 2022, 2024). However, without AI, the narrative depicting the concrete action sequences of the individual scenarios within a forensic image must still be devised and constructed manually, which is a time-consuming process and requires a great deal of manual creativity.

Initial research efforts show that AI can be used to generate background activities for forensic scenarios that are in line with the interests of the user (Voigt et al., 2024), who make use of ChatGPT's creative writing abilities to generate an individual personality profile of a user and background activities that match the user's interests. However, the work of Voigt et al. (2024) is limited to the use of an LLM in a semi-automated test setup, and the functional scope of the scenario is restricted to interest-related Google search queries and the creation of text documents.

Our work builds on the current state of research and revisits some of

* Corresponding author.https://www.unibw.de/digfor
E-mail addresses: m-plankl@web.de (M. Plankl), thomas.goebel@unibw.de (T. Göbel), harald.baier@unibw.de (H. Baier).
[1] URL: https://www.unibw.de/digfor

https://doi.org/10.1016/j.fsidi.2026.302053

its efforts. We explore the full potential of an LLM beyond its typical use as a graphical user interface, i.e., we make use of the respective LLM API. In addition, we extend the LLM usage to advanced forensically relevant artefacts such as scenario descriptions. As a final step, we demonstrate how an agent-based workflow leveraging *GPT-4o-mini* and *GPT-5-mini* can interface with the data synthesis framework `ForTrace` to generate forensic datasets. In detail the main contributions of this paper are as follows.

- We provide a concept and an initial evaluation of how LLMs can be used for the automatic generation of sample forensically relevant scenarios.
- We assess, which LLM and framework for data synthesis in the forensic field are suitable for creating synthetic datasets, respectively.
- We integrate the selected LLM into the selected data synthesis framework.
- We discuss the benefits, issues, and limitations of using an LLM for data synthesis compared to a manual data creation approach.

The rest of the paper is organised as follows. In Sec. 2, we deal with the current state of research in the context of LLMs in the DF domain. Sec. 3 provides the theoretical foundations of LLMs in our scope and optimisation strategies and techniques for LLMs as used in our proof of concept (PoC). Sec. 4 describes how a suitable LLM is selected for our work and defines a structured procedure to generate forensic artefacts with this model. Sec. 5 describes our experimental setup used in order to generate forensic scenario descriptions and different relevant artefacts as well as attempts to qualitatively or quantitatively assess the model output. In Sec. 6, we show how the selected LLM can be integrated into a DF data synthesis framework. While Sec. 7 discusses our actual results, Sec. 8 concludes our work and provides an outlook on possible future research.

2. Related work

In this section, we review related work in the scope of forensic dataset generation and AI-assisted forensic investigations.

2.1. Data synthesis/generation in DF

Publicly available data (sets) in DF and thus in the field of law enforcement are rather rare, thus the forensic community argues that synthetically generated data is relevant (Göbel et al., 2023) and proposes various principles and characteristics that should be taken into account when creating and processing a dataset in order to produce high-quality datasets (Göbel et al., 2025).

Recently, more and more data synthesis frameworks have been published in the field of DF. They try to simulate real scenarios and automate the generation of associated forensic artefacts. A scenario is a constructed situation that specifically generates traces in a controlled environment and often resembles typical real cases. In their work, Göbel et al. (2022) provide an overview of existing data synthesis frameworks in the DF context. These kinds of data generation are mostly based on the common idea of using a virtualised environment in which an agent communicates with the host system to generate forensic artefacts. The approaches differ in terms of the technologies used (e.g., agent-based vs. agentless synthesis approaches (Wolf et al., 2024)), availability, capabilities, supported operating systems, and types of artefacts that can be generated.

2.2. Applications of LLMs in DF

GenAI and LLMs have become prominent topics of global discussion, prompting researchers to intensify their investigations utilising their capabilities. Also in law enforcement, it becomes increasingly important to understand the role of GenAI and recognise why it is a game-changing technology in order to integrate AI into the daily tasks of DFIR.

A comprehensive literature review by Wickramasekara et al. (2025) shows the increasing attention of academia, industry, and research, as well as the particularly great potential of LLMs for improving the efficiency of DF investigations. Michelet and Breitinger (2024) investigated the extent to which the writing of forensic reports can be automated using LLMs, however, there is no relation to data synthesis.

Sakshi and Wadhwa (2023) developed a model of how ChatGPT could be integrated into the existing investigative process to help investigators identify and analyse digital evidence. However, they emphasise that human intervention is still required to control model decisions and ensure the reliability of the results. Scanlon et al. (2023) investigated the possible applications of ChatGPT in DF and performed exploratory experiments in individual forensic subdisciplines. Their results show that ChatGPT can be a useful tool for creative and standardised processes. Relevant for our work is that the authors see great potential in generating creative forensic scenarios for educational purposes that do not require strict correctness criteria, i.e., using ChatGPT to generate overall storyboards for cases such as intellectual property theft or stalking including innocuous activity, as well as actions related to the scenario.

Voigt et al. (2024) used the previous findings and integrated them into a semi-automated process flow to generate synthetic images. They generate user profiles and coherent background activities with ChatGPT. Using the GUI automation tool *pyautoqemu*[2] by Schmidt et al. (2023), they execute activities in a VM and thereby create synthetic datasets. However, their concept does not fully utilise the functional scope of modern LLMs as their approach still requires manual interaction with the language model via its web interface.

Studiawan et al. (2025) propose a standardised methodology and dataset for evaluating the application of LLMs for DF tasks, specifically in timeline analysis. Using exemplary DF tasks like pattern matching, anomaly detection, and event summarisation, they demonstrate that LLM's performance improves significantly when provided with additional knowledge and supporting tools. A key limitation of their BLEU-based evaluation is that it measures only textual overlap, not semantic correctness, potentially undervaluing accurate but differently worded outputs.

Pawlaszczyk et al. (2025) propose a novel method for the automated generation of synthetic datasets in mobile forensics. Their method employs an LLM to create “storyboards” defining sequences of predefined user activities using a custom markdown syntax, which are then executed on physical devices. However, because users must manually develop detailed forensic narratives from scratch, dataset generation remains time-intensive despite the technical automation.

3. On optimising and extending LLMs

In this section we present strategies used to optimise the output of LLMs if used for data generation. Various optimisation methods exist in the literature to systematically influence the interaction between the input context and the resulting model output. These key strategies are summarised under the term *prompt engineering*. With additional architectural patterns and external tools like *Retrieval-Augmented Generation* and *function calling*, the inherent functions of an LLM can be expanded and its weaknesses compensated. We also present an *AI agent-based approach* to automate the model access step by step, thus increasing both the efficiency and robustness of the interaction. Finally we point to some risks and limitations.

[2] https://wiwi-gitlab.uni-muenster.de/itsecurity/pyautoqemu (no longer available as of September 2025).

3.1. Prompt engineering

A prompt is a task-specific instruction or contextual information in natural language that causes an LLM to produce appropriate output without adjusting the model parameters (Liu et al., 2023). A prompt typically consists of (1) the instruction describing the specific task that the model is to perform, (2) the context providing additional information or external knowledge to guide the model's responses, (3) the input data presenting the core question or describing the problem that the model is supposed to solve, and (4) the expense indicator specifying the desired output format (e.g., a short response, a long text or a structured output) (Giray, 2023).

Prompt engineering refers to the systematic process of constructing such prompts to improve the model output in a targeted manner, it is an active research field including a variety of different techniques such as:

- *Zero-shot Prompting* instructs a pre-trained language model to solve a task without having been specifically trained for it.
- *Few-shot Prompting* (often referred to as in-context learning) conditions a language model by providing multiple examples in the prompt so that it can better understand the context and nuances of the problem.
- *Prompt Chaining* links multiple prompts. The output of a previous prompt serves as input for the next prompt to process a complex or multi-step task.
- *Meta Prompting* uses an LLM to create and improve prompts.
- *Chain-of-Thought (CoT) Prompting* is a method that forces LLMs to follow a series of steps, resulting in more structured, transparent, and accurate output.

3.2. Retrieval-Augmented Generation (RAG)

The knowledge of LLMs is static. It is essentially limited to information gathered during training. New knowledge cannot be automatically integrated. Therefore, LLMs tend to hallucinate when processing domain-specific or knowledge-intensive tasks that go beyond their training data or require up-to-date information (Lewis et al., 2020). RAG links an LLM with an external knowledge source to generate more detailed, factual, and precise answers.

3.3. Function calling

The knowledge of LLMs is limited as it is determined by the timeliness and scope of the data used to train the model. The language model cannot access information and events outside the training period. Function Calling attempts to compensate for these limitations. It grants a language model access to external functions and APIs to retrieve additional information or perform specialised calculations and tasks.

3.4. AI agents

LLMs typically follow a rigid dialogue-oriented control flow. This is unsuitable for the generation of complex forensic scenarios requiring collaborative information exchange between several systems. An agent-based workflow (agentic AI) offers a dynamic approach instead. Wickramasekara and Scanlon (2024) present a framework utilising AI agents and LLMs to perform tasks articulated in natural language by a human agent. Typically, the agent uses the language model for strategy and decision-making and for processing the input. At the same time, the language model depends on the support of the agent and its external modules since it has no direct interfaces to the outside world or can carry out independent actions.

3.5. Risks and limitations

Neural networks tend to store training data. This tendency intensifies with increasing model size. On the one hand, this model property enables the acquisition of in-depth knowledge of the world, on the other hand, it can reinforce misinformation and social prejudices from the training data (Huang et al., 2025). Hallucinations refer to generating convincingly formulated but inaccurate or factually false content. This misinformation arises when the model is based on uncertain or inadequate training data, misinterprets contextual relationships, or creates non-existent information due to probabilistic generation processes (Huang et al., 2025). Possible countermeasures to reduce hallucinations include providing high-quality training datasets and using credible external sources of information and tools (Peng et al., 2023).

4. Conceptualisation of data synthesis using LLMs

First, in Sec. 4.1, we describe our methodology for selecting the LLM used for data synthesis. Through a structured analysis, we define requirements and evaluation criteria for the model to be selected. Then, in Sec. 4.2, we decide which data synthesis framework is suitable for an LLM integration. Last, in Sec. 4.3, we present our uniform, multi-stage procedure for artefact generation, which we later apply to all trace types to ensure a systematic and comparable approach.

4.1. Selection of a suitable LLM

In order to evaluate our LLM-based scenario generation, we selected a suitable LLM for scenario generation through a structured two-step process, starting with a *small*, fixed-size pre-selection of ten recent and popular candidates based on a widely accepted community preference ranking by users as collected by the *Chatbot Arena* platform.[3] We excluded models that were not in the top 10, were classified as experimental, or received fewer than 10,000 votes, as we considered them insufficiently tested or under-represented for our purposes. As of September 2024, when we had to select the LLM for this work, the chosen candidates came from Google (different Gemini-1.5 models), OpenAI (different GPT-4 models), Anthropic (different Claude 3 models), and Alibaba (Qwen2-VL-72B).

Open-weight models such as Llama were excluded due to the significantly higher operational overhead of self-hosting with our limited hardware resources. Consequently, we focus solely on what is technically achievable with state-of-the-art foundation models and do not evaluate potential quality trade-offs of using smaller, locally deployable alternatives.

Our first criteria for the second step and hence the final choice comprises mandatory API availability for autonomous integration of the LLM in the data synthesis framework. This requirement is a key issue for automation and hence scalability of our approach. Our second requirement is compatibility with *LangChain*,[4] a software framework to integrate LLMs into applications (i.e., a data synthesis framework). Our third requirement constitutes a critical analysis of cost-efficiency, particularly per token, given the expected iterative experiments. Although our preselection includes various performance tiers, including models aiming for the highest prediction accuracies (like *GPT-4o*, *Claude 3 Opus*, and *Gemini Pro 1.5*), we did not choose them due to their significantly higher cost per token, which would have substantially increased the financial overhead for our expected high number of iterative experiments. However, our approach can be used with these models, too. Finally, older model versions, such as *GPT-4-Turbo* and *Claude 3 Opus* (relative to *Claude 3.5 Sonnet*), were not selected as newer variants typically offer better performance or lower costs, as indicated by updated Arena rankings and pricing structures.

Our analysis led to the selection of *GPT-4o-mini* from OpenAI, chosen for its favourable price-performance ratio, best suited for our iterative

[3] https://lmarena.ai (last accessed on 2025-10-07).
[4] https://langchain.com (last accessed on 2025-10-07).

experimental setup, meaning that models with less optimal cost-efficiency relative to their performance were not ultimately selected despite meeting baseline technical and functional requirements.

Since our initial evaluation, newer and superior models have been released. Rather than deferring them to future work, we integrated them into our iterative experiments whenever our primary model choice limited the required forensic scenario complexity. Specifically, *GPT-5-mini* was subsequently employed in Sec. 6.2 to address coherence constraints that could not be reliably enforced with *GPT-4o-mini*, and *Gemini 2.5* was examined to assess cross-model transferability of our workflow in Sec. 7.2.

4.2. Selection of a suitable data synthesis framework

Several prerequisites are necessary for integrating an LLM into an existing framework. Most importantly, the source code of the project must be freely available so that it can be expanded accordingly. The tools `Forensig2` (Moch and Freiling, 2009, 2012), `EviPlant` (Scanlon et al., 2017) and `TraceGen` (Du et al., 2021) are not suitable for this as their source code is either not available or has never been published. `ForGeOSI`,[5] and `ForGe` (Visti et al., 2015) are based on Python 2 and do not offer any compatibility with libraries of current LLMs that require at least Python 3. Porting the existing code base to a newer Python version is not within the scope of this work. `VMPOP` (Park, 2018) only serves as a PoC and does not pursue a holistic data synthesis approach (i.e., synthesis of traces on different abstraction levels such as hard disk, memory, and network layer). The framework `hystck` (Göbel et al., 2020) also does not offer a holistic approach and has been replaced by its successors `ForTrace` (Göbel et al., 2022) and `ForTrace++` (Wolf et al., 2024), respectively. Both `ForTrace` (agent-based version) and `ForTrace++` (agentless version) meet the necessary requirements for integrating an LLM, are written in Python 3, are actively being developed, and pursue a comprehensive and holistic data synthesis approach. For this paper, we make use of `ForTrace`.[6]

4.3. Procedure for scenario generation

The implementation of our concept makes use of three key components, that is *Jupyter Notebook*[7] (using Python 3.12) to edit and maintain our source code, the OpenAI API for our chosen LLM *GPT-4o-mini* to communicate to the LLM, and the *LangChain* framework to integrate the LLM into the data synthesis framework `ForTrace` (as explained in Sec. 6). *LangChain* allows us to integrate and interact with various LLMs using generically implemented chat interfaces, ensuring high adaptability without vendor-specific libraries. This framework also provides abstractions for common design patterns like Chains, Agents and RAG to retrieve and use updated information with the LLM, simplifying the implementation of complex workflows. Accessing the OpenAI API requires a commercial developer account. To maintain result consistency, we set the 'temperature' hyperparameter to 0, leaving other parameters at their default values, thus reducing the randomness of the GPT outputs.

5. Forensic artefact creation using GenAI

This section presents our LLM-based creation of forensically relevant scenario storylines as well as LLM-based generation of sample traces like browser activities and associated databases. Our used prompts are available via GitHub: https://github.com/lawly/prompts.

5.1. Prompts to generate scenario descriptions

The automatic creation of text content is particularly necessary for describing the exact sequence of events in a forensically relevant scenario (alongside the typical AI-based creation of content, e.g., in the form of text files). However, generating a complete story from simple textual inputs, such as a topic or prompt, can lead to repetitions and incoherence. In the following section, we discuss how elements such as title, plot, prompt, or characters can shape and control the generation process.

Fig. 1 illustrates the four steps of our workflow to automatically create a storyline of a forensic scenario with several actors in different roles using an LLM and prompt chaining. We deliberately begin by creating personas (as clearly defined actors serve as hard constraints for all subsequent steps) so that the plot, relationships, and system activities can be generated in a causally consistent and forensically coherent manner. In what follows we explain each of the four steps and the used prompts, more details can be found via our prompt GitHub (see start of Sec. 5.1).

The prompt $f_{\text{generate\_personas}}(\textit{count})$[8] creates profiles for the actors involved, i.e. a suspect, an external accomplice, and a user-defined number of innocents. Each profile contains attributes such as name, age, username, email, interests, and case role (e.g., suspect, innocent, external). It uses zero-shot prompting with a clear, detailed task description instead of examples, and CoT prompting to guide step-by-step planning, development, and validation.

The prompt $f_{\text{generate\_plot}}(\textit{personas})$[8] creates a short plot for a fictional forensic scenario based on previously generated person profiles, forming the foundation for subsequent events and actions. It employs few-shot prompting with an example of the desired plot format, role prompting to establish decision-making context, and CoT prompting to enable explicit scenario planning.

The prompt $f_{\text{generate\_relations}}(\textit{personas}, \textit{plot})$[8] generates a relationship network among the actors based on previously created characters and the plot. Each relationship reflects logical personal or professional connections consistent with the story and the actors' interests. An example relationship diagram is available on GitHub,[8] illustrating that the suspect Ethan Chase maintains only one potentially risky, case-relevant connection—with his external accomplice, Derek Henson.

Finally $f_{\text{generate\_activities}}(\textit{personas}, \textit{plot}, \textit{relations})$[8] generates a table containing system activities that were created on the shared computer and should be consistent with the previously generated components of the scenario. This allows the model to define specifications for system activities. Synthesis frameworks such as `ForTrace` implement a finite set of possible actions.

5.2. Prompts for generating additional artefacts

In the scope of this work, we were able to generate further forensic artefacts, however, we cannot discuss them in detail here due to space constraints. For example, a similar prompt-driven workflow as in Sec. 5.1 was used to generate synthetic browser artefacts.[8] We invoked the Google Search API via function calling to produce non-hallucinated, interest-based URLs (Voigt et al., 2024). Database manipulation was achieved by combining Text-to-SQL with function calling to generate schema-aware SQL commands that programmatically inserted coherent browsing history entries into the `places.sqlite` database of Firefox. Evaluation against manually produced histories showed the approach yields valid URLs, accurate page titles, and plausible navigation patterns, though SQLite-specific character masking and Firefox-specific fields limited fully automated, forensically reliable reproduction.

[5] https://github.com/maxfragg/ForGeOSI/, (last accessed on 2025-05-12).
[6] https://github.com/dasec/ForTrace (last accessed on 2025-12-16).
[7] https://github.com/jupyter/notebook (last accessed on 2025-10-07).
[8] https://github.com/lawly/prompts/blob/main/chapter-5-prompts.md (last access 2025-12-16).

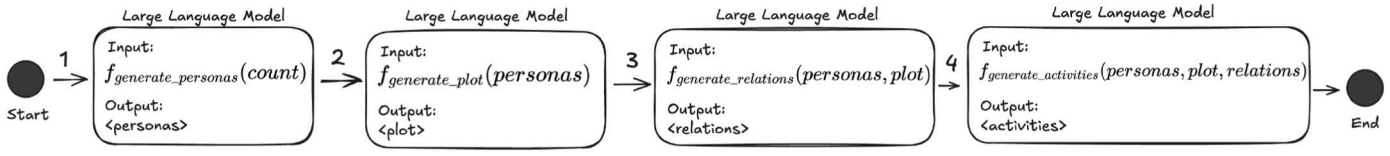

Fig. 1. Our workflow to create a forensic scenario.

Weak passwords[9] mimicking real-world patterns (for instance from the "RockYou" list) were generated using statistical rules from *Pipal*. While 80 % matched insecure trends, prompt sensitivity affected consistency.

Next diffusion models (e.g., *DALL-E 3*, *FLUX.1-dev*, *Recraft V3*) were used to generate images[9] relevant to forensic scenarios, such as simulated crime scenes or drug-related content. While *FLUX.1-dev* produced unfiltered outputs, proprietary models like *DALL-E 3* blocked the generation of sensitive content such as CSAM. Challenges included precise adherence to the format requirements of text and the variability in the safety filters of the models.

6. Integration of an LLM into the data synthesis framework ForTrace

To integrate an LLM into `ForTrace` in the best possible way, we first analyse its programme architecture. It is necessary to examine where the language model can support the synthesis process while also being smoothly integrated into existing workflows. Subsequently, it is analysed which structural and architectural adjustments need to be made to the existing code base.

`ForTrace` offers two different approaches for creating forensic scenarios - either by directly programming Python scripts or by using YAML configuration files. Our results in Sec. 5 show that an LLM can interpret and generate well-formed data structures (such as the YAML configuration files of `ForTrace`). The evaluation in Sec. 5.1 also showed that an LLM can construct forensic scenarios with a limited portfolio of user actions and complex relationships between different actors. An RAG approach can compensate for missing domain knowledge. Therefore, as a PoC, we will demonstrate how the scenario creation in `ForTrace` can be improved using the framework's YAML configuration files in conjunction with an LLM. The code of our prototype implementation is available via GitHub: https://github.com/lawly/fortrace-yaml-generator.

6.1. Architecture of the LLM integration

Fig. 2 shows the architecture of our PoC implementation. The modular system interacts with a web application, a vector database, and external APIs via defined interfaces. The workflow comprises the following six steps:

1. The user communicates with the server component via a *Streamlit* web interface.
2. The web application communicates with the server, which uses the *LangChain* framework to interact with an LLM and reduces development effort. The backend runs within a *Docker* container.
3. *GPT-4o-mini* is used as the language model. The abstraction layers provided in *LangChain* enable future replacement of the language model with a simple configuration change.
4. Naturally, the knowledge base of the LLM is insufficient to understand the declarative module descriptions actually implemented for the YAML configuration files of `ForTrace`. Missing domain knowledge is provided to the model through an RAG approach. Therefore, a *FAISS* vector database is used via a shared folder within the *Docker* container.
5. Typically, complex tasks are solved using a prompt chaining approach with several subtasks. However, a multi-stage decision-making process, failed API calls, or incorrect intermediate results can make it challenging to identify failure points. Structured logging with *LangSmith* is designed to support the development process and provide insights into internal processing steps.
6. Finally, the YAML configuration file generated by the AI agent-based approach is returned to the user via the web interface.

6.2. Practical integration of an LLM

The YAML configuration files for `ForTrace` follow a modular structure. Each module is identified by a namespace, e.g., *fortrace.usermanagement* or *fortrace.browser*, and defines specific actions like `create` or `browse_to`. Each action includes mandatory and optional parameters that control its context and execution. A naive few-shot prompting approach is unsuitable since this structure is not inherently known to the LLM. It would require transmitting the full YAML specification in each prompt, leading to excessive token usage and higher costs. Moreover, the variability in required and optional parameters across modules would necessitate exhaustive examples, complicating the differentiation between necessary and optional elements.

6.2.1. Structure of the vector store

We use a vector store to compensate for the lack of prior knowledge of the LLM about the structure and functionality of `ForTrace`'s modules. Using individual usage examples of the synthesis actions proved to be imprecise during the testing phase. Without additional contextual guidance, the model could not properly distinguish between mandatory and optional parameters of the different actions. We decided to create a JSON schema based on the existing source code documentation for each `ForTrace` module to provide more contextual information to the LLM. Each schema can be enriched with a detailed description of all parameters and different usage examples. We determined that the existing function descriptions did not fully utilise the notation scope of *Sphinx*. Furthermore, it was noted that not all functionalities were described consistently.

Fig. 3 illustrates the procedure for optimising and standardising the documentation of existing functions. Our goal is to create consistent and understandable documentation that ensures both syntactic and semantic clarity. The steps are:

1. A Python script extracts the relevant functions and documentation from the source code of `ForTrace`.
2. We use $f_{\text{enhance\_docstring}}(docstring, code)$[10] to generate new function documentations from existing ones and the associated program code. The new documentation was generated and completed using clear and uniformly structured descriptions of individual actions, parameters, supported types, default values, and return types according to the formatting requirements of *Sphinx*.

[9] https://github.com/lawly/prompts/blob/main/chapter-5-prompts.md (last accessed on 2025-10-06).

[10] https://github.com/lawly/prompts/blob/main/chapter-6-prompts.md (last accessed on 2025-12-16).

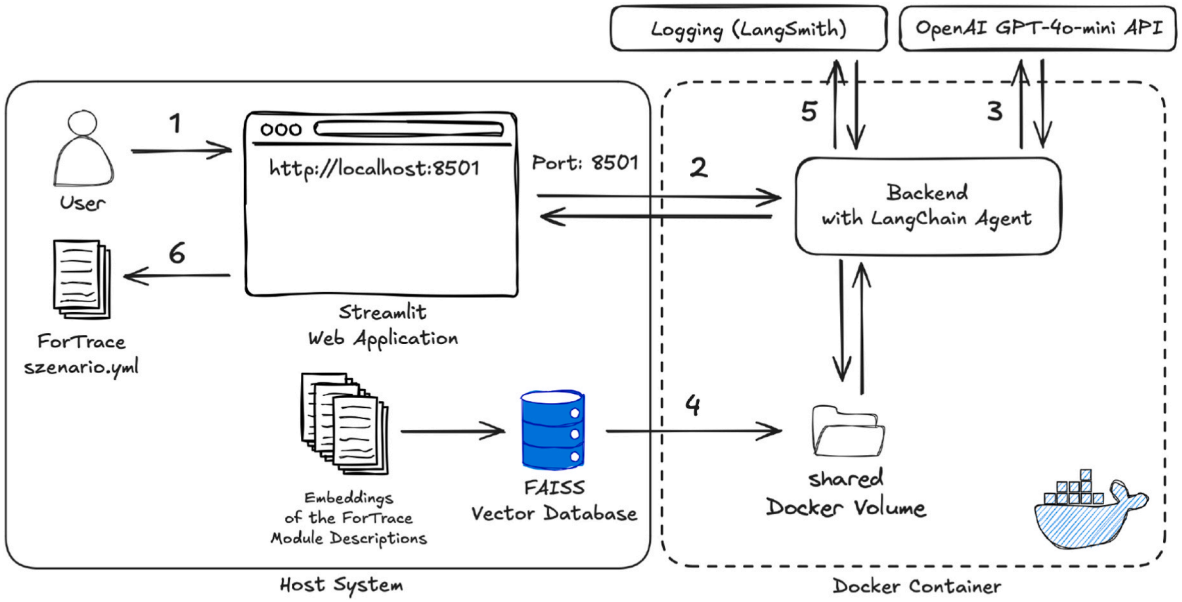

Fig. 2. Architecture of the LLM-driven generation of YAML configuration files for use with the data synthesis framework `ForTrace`.

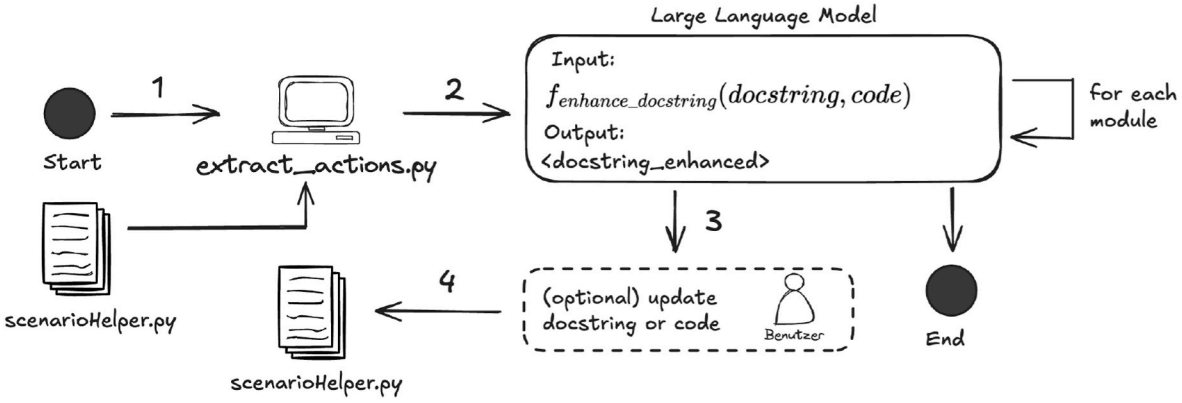

Fig. 3. Flowchart for the automated improvement and standardisation of the existing *Sphinx* function documentation in `ForTrace`.

3. We check the correctness of the model's output manually. During this work, we noticed that individual actions were occasionally incorrectly named or missing in the original function description. This was corrected manually in this step.

The revised documentation is used to create the structured JSON schemas. Fig. 4 illustrates the procedure for creating the vector store.

1. $f_{\text{create\_rag\_schema}}(docstring\_enhanced, yaml\_examples)$[10] is used to transform the revised documentation into structured JSON schemas. Additionally, each schema is enriched with usage examples.
2. We converted each schema to *LangChain's Document* abstraction to be able to leverage a semantic retrieval via the vector store later.
3. We generate word embeddings from these documents using OpenAI's *text-embedding-3-small* model, and store them in a local FAISS vector store.
4. The computed embeddings are persisted in the local vector store on the host system.

6.2.2. Creation of a synthetic ForTrace configuration file

Fig. 5 illustrates the workflow for generating a synthetic YAML configuration file for `ForTrace`. The individual steps are as follows.

- The prompt $f_{\text{generate\_personas}'}(count)$[10] is similar to Sec. 5.1. During testing, several mail providers suffered hallucinations and produced incorrect mail settings. To ensure reliability, a local SQLite database is prepared that stores mail providers and valid connection settings that are proven to work seamlessly with `ForTrace` to enforce strict control over the set of allowed providers.
- $f_{\text{generate\_plot}'}(personas)$[10] follows the methodology for generating a plot as described in Sec. 5.1. It contains a summary of the available synthesis functions to ensure that the storyline can be mapped to `ForTrace` actions.
- The creation of relationships between actors remains unchanged.[10]
- $f_{\text{generate\_activities}'}(personas, relations, plot)$[10] is generating a tabular representation of the users' actions. In addition, a set of logical and content-related constraints is introduced. This rule set is designed to ensure that synthetically generated system activities are reliable and free from logical errors, e.g., the web browser is opened before a website can be visited.
- Personas, plot, relationships, and activities are consolidated into structured YAML via using the prompt $f_{\text{generate\_yaml}}(personas, relations, plot, activities)$,[10] it is reflecting `ForTrace`'s required format, too. A FAISS retrieval approach compensates for the LLM's limited `ForTrace` module knowledge by incorporating retrieved schemata as additional prompt context.

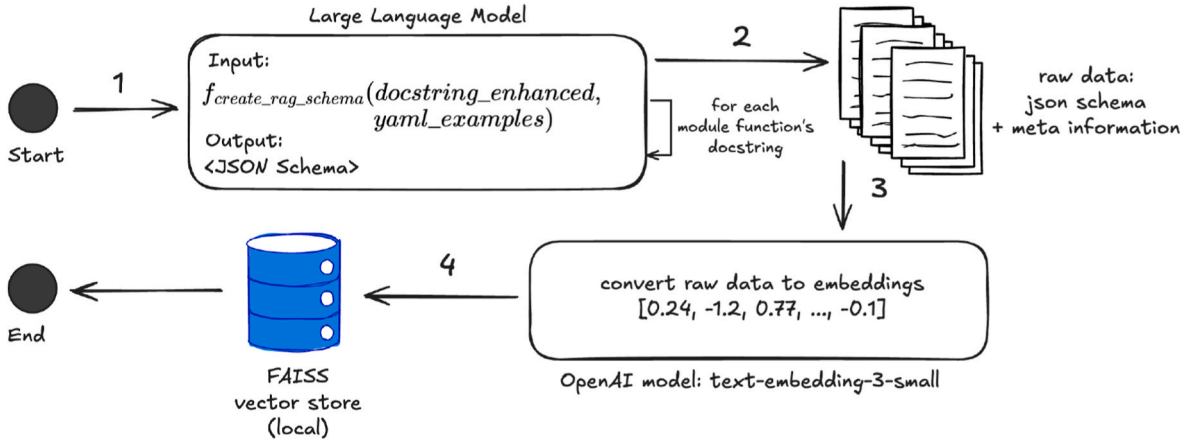

Fig. 4. Automated process for building a FAISS vector store. We use an LLM to translate each function documentation into a JSON schema. Word embeddings can be created from these schemata and are then persisted in a vector.

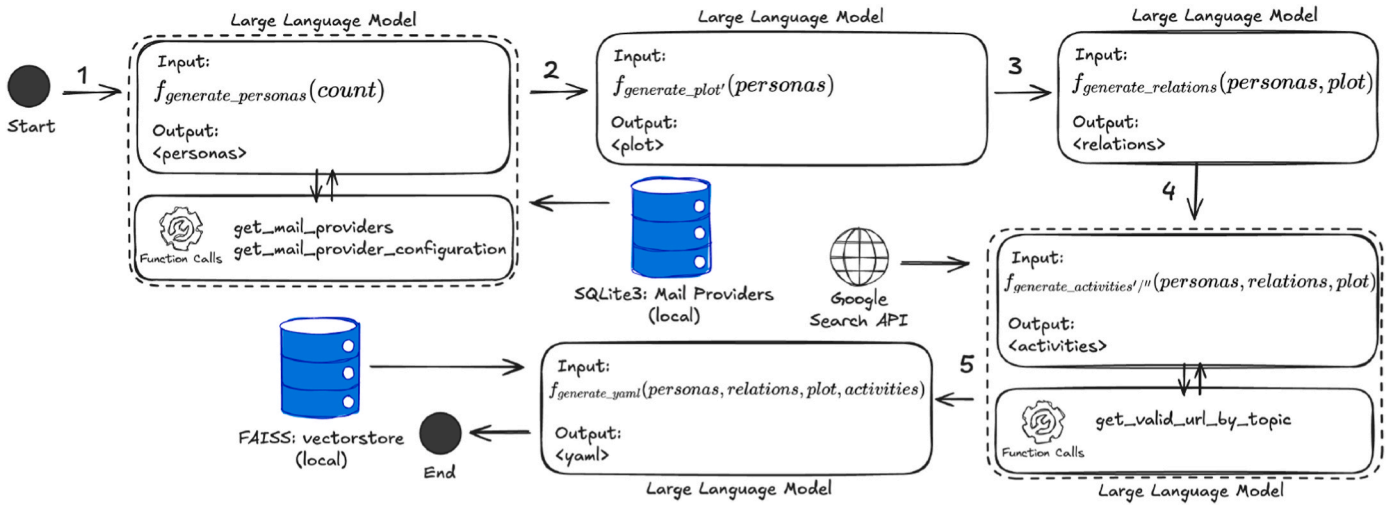

Fig. 5. Prompt chaining workflow to automatically generate a `ForTrace` YAML configuration based on the proposed LLM integration.

Using $f_{\text{generate\_activities}'}(personas, relations, plot)$ did not fully enforce the specified coherence rules, as initial experiments revealed. The generated system activities often lacked explicit user transitions via `switch_to`, and file operations were sometimes executed out of order, with files used before creation. Across the conducted experiments, no variations of the prompt was identified that reliably enforced all coherence rules. It appears that under the given conditions, *GPT-4o-mini* cannot fully handle the diversity and complexity of requirements specified in the prompt.

However, we could satisfy the coherence rules using the reasoning model *GPT-5-mini*. Reasoning models improve the capabilities of LLMs by embedding a native CoT process for structured problem solving. As a result, explicit activation phrases such as "*Think step-by-step*" and predefined reasoning structures are no longer necessary. We therefore removed the manual thought structure from the prompt and assumed that the reasoning model could formulate a superior planning phase. Apart from relocating the rule set for the synthetic activities to the requirements section of the prompt (i.e., the section enumerating mandatory attributes and constraints for each generated artefact), no further changes were required.

6.2.3. Provision of a user interface

The web interface of our application, built with the Python framework *Streamlit*, consists of two main sections. The interactive configuration section lets users control each prompt and model parameter in the prompt chain, enabling customization of prompts, LLMs, and parameters without programming expertise. Settings can be saved in a configuration file, and the generation process is triggered via a dedicated button. The visualization section on the right-hand side of the web interface sequentially displays model outputs for each prompt, including character profiles, the plot, relationship networks, system activities, and the corresponding YAML configuration for `ForTrace`.

6.3. Evaluation of a synthetic scenario generated by GPT

We generated key components of a synthetic forensic scenario using our workflow including individual personas, their relations, the plot, and the tabular system activities.[11] We then consolidated into an automatically generated `ForTrace` YAML configuration.[12] To evaluate the correctness and forensic soundness of the automatically generated scenario, the LLM-generated `ForTrace` YAML configuration was executed to produce a Windows disk image, which was subsequently analysed using `Autopsy`. Due to space limitations, the complete forensic evaluation, including screenshots and detailed artefact validation, is provided

[11] https://github.com/lawly/prompts/blob/main/chapter-6-prompts.md (last accessed on 2025-12-16).

[12] https://github.com/lawly/prompts/blob/main/scenario.yaml (last accessed on 2025-12-16).

in our public GitHub repository.[13]

The analysis confirms that the generated disk image is internally consistent and accurately reflects the personas, roles, and activity constraints defined in the synthetic scenario description. All specified user accounts, files, browser artefacts, and email configurations, as well as sent emails, were created as intended, with plausible timestamps and coherent sequence of events. Background activity generated the innocent user is clearly distinguishable from case-relevant artefacts attributed to the suspect, demonstrating correct role separation and semantic alignment between the scenario description and the resulting system traces.

Testing $f_{\text{generate\_activities}'}(\textit{personas}, \textit{relations}, \textit{plot})$ with different LLMs revealed that timestamps vary by model and temperature settings: both *GPT-4o-mini* and *Gemini 2.5 Flash* consistently produce 2023 timestamps (which can vary with temperature adjustments), while *GPT-5-mini* consistently outputs 2025 timestamps. We suspect that the model reflects its training data timeframe (so-called knowledge cutoff) and the lack of dedicated temporal prompt specification, which suggests a tendency to default to familiar periods when not explicitly guided. This behaviour should further be researched in subsequent work.

Overall, the results show that the LLM-generated configuration can be executed without manual correction and produces a forensically plausible and verifiable dataset. This confirms the practical feasibility of the proposed LLM-driven scenario generation and execution workflow.

7. Discussion and limitations

In this section we critically discuss our results, including methodological limitations and open questions, to provide a well-founded interpretation of the findings. The insights gained offer new perspectives and highlight areas for future research efforts.

7.1. Risks of LLMs in artefact generation

We first discuss the recurring risks and limitations associated with the use of LLMs on the quality, consistency, and reliability of synthetically generated artefacts covered in Sec. 5.2 as well as forensic scenarios addressed in Sec. 6.2.

7.1.1. Lack of explainability

The black-box nature and lack of explainability of LLMs present a key challenge in DF, impacting traceability and legal admissibility (Scanlon et al., 2023). This issue is comparable to the challenges in mobile forensics, where proprietary and obscure methods for data extraction are often used (Ottmann et al., 2021). Despite their lack of transparency, these techniques are used because of their high forensic value, necessitating continuous evaluation and validation to preserve the legal admissibility of the evidence they produce. Analogously, this underscores the need for a thorough understanding and methodical approach when working with LLMs. Our work specifically explored methods to increase transparency into the model's decision-making process to address this need for understanding and validation. We found that CoT Prompting effectively improves traceability by externalising the model's reasoning steps, proving valuable for scenario generation and error analysis. Reasoning models that natively integrate CoT showed potential for enhanced consistency and simplified prompting.

7.1.2. Timeliness of the knowledge base

To address the limitation of LLMs' static domain knowledge in specific forensic tasks, we investigated an RAG approach. Our results showed its significant potential: integrating an external knowledge base successfully compensated for missing knowledge, eliminating hallucinations and enabling the generation of accurate, contextually relevant forensic data. However, our practical implementation encountered issues with the FAISS vector store configuration.

7.1.3. Stochastic behaviour

A key challenge with LLMs is their non-deterministic and unpredictable behaviour, complicating the reproducibility typically required in DF. While beneficial for generating diverse synthetic data, their inherent stochastic nature makes them unreliable for tasks demanding exact deterministic results. It also makes model behaviour often unpredictable, with minor prompt changes leading to significantly different outputs. We view this nuanced behaviour as beneficial for generating diverse synthetic datasets, addressing data scarcity, but challenging for tasks demanding exact, deterministic results like Base64 decoding or encryption. To address unreliability in exact tasks, we propose delegating deterministic operations to external functions via function calling. For complex rule-based tasks with semantic context, our results suggest that these should be handled by reasoning models, which demonstrated improved rule adherence.

7.2. Changing the LLM in use

Given the rapid evolution of LLMs, numerous new models have been released since we created the workflow in Sec. 6.2. During our experiments, the necessity to switch from *GPT-4o-mini* to *GPT-5-mini* for handling more complex prompts further underscored the importance of examining the transferability and model-agnostic robustness of our workflow across different foundation models. To thoroughly investigate this, we repeated the scenario generation process with Google's *Gemini 2.5 Flash* and *Gemini 2.5 Pro* models.

Switching between different OpenAI models was technically straightforward. It only required specifying a different model name within the existing *LangChain* abstraction. Transitioning from GPT to a Gemini requires replacing the *LangChain* interface `ChatOpenAI` abstraction with `ChatGoogleGenerativeAI` and supplying the corresponding API key.

However, the prompts initially tailored for OpenAI's models proved to be only partially compatible with the models from Google. While both models correctly understood the forensic context and produced coherent narratives, they often failed to comply with the strict structural and formatting requirements necessary for automated processing. A critical observation across the test runs was their inconsistent omission of mandatory XML tags intended to delimit the required output from the rest of the response. This critical failure to maintain the specified structural formatting prevented the programmatic extraction and transmission of data to the next agent in the pipeline and repeatedly broke the automated workflow. Another common error was an invalid Mermaid syntax that disrupted visualization of the actors' relations. The strict rate limit of *Gemini 2.5 Pro*'s free tier restricted to five API calls per minute[14] renders it impractical for our multi-stage workflow, while the paid tier is over seven times more expensive than *GPT-5-mini*, making large-scale scenario generation economically and operationally infeasible. In contrast, *Gemini 2.5 Flash* operated with a less strict rate limitations but showed weaker functional integration, failing to invoke function calls for mail configuration retrieval, resulting in hallucinated data. To ensure stable execution under *Gemini 2.5 Flash*, the prompt set required targeted adaptations. All updated prompts can be found on GitHub.[15]

[13] https://github.com/lawly/prompts/blob/main/evaluation.md (last accessed on 2025-12-16).

[14] https://ai.google.dev/gemini-api/docs/rate-limits#free-tier (last accessed on 2025-10-07).

[15] https://github.com/lawly/prompts/blob/main/chapter-7-prompts.md (last accessed on 2025-12-16).

These findings underscore the sensitivity of structured prompting to model-specific interpretation and highlight the need for adaptive prompt tuning and validation to ensure reliable workflow transfer across foundation models.

7.3. Recap of our contributions

This section shortly reviews our contributions. First we presented our methodology for selecting the LLM used for data synthesis in Sec. 4.1. Through a structured analysis, we defined requirements and evaluation criteria for the model to be selected. Our analysis led to the selection of *GPT-4o-mini* from OpenAI, chosen for its favourable price-performance ratio, best suited for our iterative experimental setup. In addition, `ForTrace` (and `ForTrace++`) were the only framework meeting all requirements.

In Sec. 5 we conducted several experiments to demonstrate that LLMs can automatically generate forensically relevant scenarios and artefacts. The results show that an LLM can generate a wide range of synthetic artefacts. In our experiments, no single, uniform method proved equally suitable for creating all types of forensic traces. The approach proved to be trace-dependent. The diversity of generated traces depends on both the access method to the LLM and the chosen architectural approach. While direct use of ChatGPT via the web interface may be primarily restricted to textual artefacts according to Scanlon et al. (2023), Studiawan et al. (2025) demonstrated that the use of OpenAI's integrated *Code Interpreter* can expand its capabilities. However, an API-driven approach allows an even more flexible utilisation of the model. It enables local file manipulation and can be integrated into architectural design patterns like RAG for a broader spectrum of synthetic artefacts. The use of prompt chaining has proven to be a particularly practical architectural approach.

A demo application was developed in Sec. 6 to integrate the selected LLM into the chosen data synthesis framework. The initial review identified several potential integration points. While the integration offered automation benefits, it also introduced technical challenges. A vector store proved useful for providing domain knowledge to the LLM, but it increased maintenance overhead. The introduction of the new modules in `ForTrace` must be promptly reflected in the database. As shown in Sec. 6.2, even this step can be simplified with an LLM. However, query optimisation in the RAG system remains an open issue. The modular setup in Sec. 5.1 using prompt chaining reduced the integration effort. Some prompts required minimal changes, while others could be reused without any additional changes.

We analysed the benefits, issues, and limitations of using an LLM for data synthesis compared to a manual approach, based on the results in Sec. 5 and Sec. 6. While a manual creation of curated synthetic datasets offers fine-grained control over traces, it is time-consuming, error-prone, and poorly scalable (Göbel et al., 2023; Woods et al., 2011). Our experiments demonstrate that LLMs are able to significantly automate and scale data synthesis. Using tailored, chained prompts, we developed automated workflows capable of generating diverse synthetic artefacts with minimal user interaction, as demonstrated by our practical implementation in Sec. 6. However, designing these workflows in the first place is complex, requiring iterative decomposition of problems and advanced prompt engineering for high-quality, coherent, and accurate output while avoiding hallucinations. Therefore, the right choice between manual synthesis and LLM-based automation is context-dependent. Manual approaches are preferable for small datasets where precise control is needed. In contrast, LLM-based synthesis is advantageous when large-scale or variant-rich data is required.

8. Conclusion and future work

Public forensic datasets remain essential for research and education, yet high-quality DF datasets are limited and often fail to meet specialised requirements. This work has shown that diverse, forensically relevant artefacts can be synthetically generated using LLMs. It has also highlighted how risks such as hallucinations, limited explainability, and stochastic behaviour affect data synthesis, and how prompt engineering and modern architectural patterns—such as function calling—can extend LLM capabilities and mitigate these limitations.

As a proof of concept, we integrated an LLM into the `ForTrace` data synthesis framework. Within an agent-based control flow, the model generated configuration files for various forensic scenarios and storylines without requiring further user interaction or knowledge of the framework's technical details. This prototype illustrates the feasibility of automatically producing coherent scenarios from natural language and offers a scalable approach to addressing DF dataset scarcity. In future, it will be necessary to generate other important types of data as part of the scenario, e.g., chat conversations, which are well within the capabilities of an LLM.

Further research should explore the potential of fine-tuning open-weight models like *gpt-oss*[16] or *DeepSeek-R1*[17] with domain-specific knowledge to determine whether this can reduce reliance on extensive and potentially complex prompt engineering techniques, leading to more accurate, relevant, and efficient artefact generation. In addition to multimodal models, Vision Language Models (VLMs) may be used to analyse visual data such as screenshots captured during the data synthesis process using frameworks such as `ForTrace`, in order to then automatically interact with the VM and, in particular, track and resolve interruptions directly during the synthesis process (Wolf and Baier, 2025).

Additional future work may address to which extent experimental concepts such as *Computer Use*[18] or *Computer-Using Agent*[19] are suitable for a data synthesis framework in DF (Hu et al., 2024). This involves answering the question of whether an LLM can remotely control a virtual environment to generate synthetic artefacts. Furthermore the possibility of replacing `ForTrace`'s guest agent by a Model Context Protocol (MCP) server should be explored to assess if this could lead to a more streamlined and simplified scenario generation process (Singh et al., 2025; Hilgert et al., 2025). Beyond architectural considerations, future work should focus on ways to explicitly control time in synthetic scenarios. This includes methods to guide or restrict timestamps so that the limitations, uncertainties, and risks of time-based scenario generation can be systematically evaluated.

Acknowledgements

We would like to thank Rene Groß from FAU for his inspiring preliminary work and the reviewers to improve our work.

References

Breitinger, F., Jotterand, A., 2023. Sharing datasets for digital forensic: a novel taxonomy and legal concerns. Forensic Sci. Int.: Digit. Invest. 45, 301562. https://doi.org/10.1016/j.fsidi.2023.301562.

Du, X., Hargreaves, C., Sheppard, J., Scanlon, M., 2021. TraceGen: user activity emulation for digital forensic test image generation. Forensic Sci. Int.: Digit. Invest. 38, 301133. https://doi.org/10.1016/j.fsidi.2021.301133.

Giray, L., 2023. Prompt engineering with ChatGPT: a guide for academic writers. Ann. Biomed. Eng. 51, 2629–2633. https://doi.org/10.1007/s10439-023-03272-4.

Göbel, T., Baier, H., 2025. From IaC to IoC—Using infrastructure as code (IaC) to generate synthetic datasets of compromised (IoC) Linux systems for use in digital forensics. Digit. Threats 6. https://doi.org/10.1145/3748268.

[16] https://huggingface.co/openai/gpt-oss-120b (last accessed on 2025-12-03).

[17] https://huggingface.co/deepseek-ai/DeepSeek-R1 (last accessed on 2025-12-03).

[18] https://www.anthropic.com/news/3-5-models-and-computer-use (last accessed on 2025-10-07).

[19] https://openai.com/index/computer-using-agent (last accessed on 2025-10-07).

Göbel, T., Baier, H., Breitinger, F., 2023. Data for digital forensics: why a discussion on "How Realistic is Synthetic Data" is dispensable. Digit. Threats: Res. Pract. 4. https://doi.org/10.1145/3609863.

Göbel, T., Schäfer, T., Hachenberger, J., Türr, J., Baier, H., 2020. A novel approach for generating synthetic datasets for digital forensics. In: Advances in Digital Forensics XVI, 16Th. Springer, pp. 73–93. https://doi.org/10.1007/978-3-030-56223-6_5.

Göbel, T., Baier, H., Wolf, D., 2024. Scenario-based data set generation for use in digital forensics: a case study. In: INFORMATIK 2024. Bonn, pp. 355–370. https://doi.org/10.18420/inf2024_25.

Göbel, T., Breitinger, F., Baier, H., 2025. Optimising data set creation in the cybersecurity landscape with a special focus on digital forensics: principles, characteristics, and use cases. Forensic Sci. Int.: Digit. Invest. 52, 301882. https://doi.org/10.1016/j.fsidi.2025.301882.

Göbel, T., Maltan, S., Türr, J., Baier, H., Mann, F., 2022. ForTrace - a holistic forensic data set synthesis framework. Forensic Sci. Int.: Digit. Invest. 40, 301344. https://doi.org/10.1016/j.fsidi.2022.301344.

Hilgert, J.N., Jakobs, C., Külper, M., Lambertz, M., Mahr, A., Padilla, E., 2025. Chances and challenges of the model context protocol in digital forensics and incident response. arXiv:2506.00274. https://doi.org/10.48550/arXiv.2506.00274.

Horsman, G., Lyle, J.R., 2021. Dataset construction challenges for digital forensics. Forensic Sci. Int.: Digit. Invest. 38, 301264. https://doi.org/10.1016/j.fsidi.2021.301264.

Hu, S., Ouyang, M., Gao, D., Shou, M.Z., 2024. The dawn of GUI agent: a preliminary case study with claude 3.5 computer use. arXiv preprint arXiv:2411.10323. https://doi.org/10.48550/arXiv.2411.10323.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., Liu, T., 2025. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. ACM Trans. Inf. Syst. 43. https://doi.org/10.48550/arXiv.2311.05232.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems. Curran Associates, Inc, pp. 9459–9474. https://doi.org/10.48550/arXiv.2005.11401.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G., 2023. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput. Surv. 55. https://doi.org/10.48550/arXiv.2107.13586.

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 Sci. Int.: Digit. Invest. 48, 301683. https://doi.org/10.1016/j.fsidi.2023.301683.

Moch, C., Freiling, F.C., 2009. The forensic image generator generator (forensig2). In: 2009 Fifth International Conference on IT Security Incident Management and IT Forensics. In: IEEE, pp. 78–93. https://doi.org/10.1109/IMF.2009.8.

Moch, C., Freiling, F.C., 2012. Evaluating the forensic image generator generator. In: Gladyshev, P., Rogers, M.K. (Eds.), Digital Forensics and Cyber Crime. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 238–252. https://doi.org/10.1007/978-3-642-35515-8_20.

Ottmann, J., Pollach, J., Scheler, N., Schneider, J., Rückert, C., Freiling, F., 2021. Zur Blackbox-Problematik im Bereich Mobilfunkforensik. Datensch. Datensicherheit-DuD 45, 546–552. https://doi.org/10.60882/cispa.27087646.

Park, J., 2018. TREDE and VMPOP: cultivating multi-purpose datasets for digital forensics–A windows registry corpus as an example. Digit. Invest. 26, 3–18. https://doi.org/10.1016/j.diin.2018.04.025.

Pawlaszczyk, D., Engler, P., Bodach, R., Hummert, C., 2025. AI-driven dataset creation in mobile forensics using LLM-based storyboards. ScienceOpen Preprints. https://doi.org/10.1016/j.fsidi.2025.302002.

Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., Huang, Q., Liden, L., Yu, Z., Chen, W., Gao, J., 2023. Check your facts and try again: improving large language models with external knowledge and automated feedback. arXiv:2302.12813. https://doi.org/10.48550/arXiv.2302.12813.

Sakshi, S.E., Wadhwa, M.. Enhancing digital investigation: leveraging ChatGPT for evidence identification and analysis in digital forensics. https://doi.org/10.1109/ICCCIS60361.2023.10425000.

Scanlon, M., Breitinger, F., Hargreaves, C., Hilgert, J.N., Sheppard, J., 2023. ChatGPT for digital forensic investigation: the good, the bad, and the unknown. Forensic Sci. Int.: Digit. Invest. 46, 301609. https://doi.org/10.1016/j.fsidi.2023.301609.

Scanlon, M., Du, X., Lillis, D., 2017. EviPlant: an efficient digital forensic challenge creation, manipulation and distribution solution. Digit. Invest. 20, S29–S36. https://doi.org/10.1016/j.diin.2017.01.010.

Schmidt, L., Kortmann, S., Hupperich, T., 2023. Improving trace synthesis by utilizing computer vision for user action emulation. Forensic Sci. Int.: Digit. Invest. 45, 301557. https://doi.org/10.1016/j.fsidi.2023.301557.

Singh, A., Ehtesham, A., Kumar, S., Khoei, T.T., 2025. A survey of the model context protocol (MCP): standardizing context to enhance large language models. . https://doi.org/10.48550/arXiv.2505.02279.

Studiawan, H., Breitinger, F., Scanlon, M., 2025. Towards a standardized methodology and dataset for evaluating LLM-based digital forensic timeline analysis. arXiv preprint arXiv:2505.03100. https://doi.org/10.1016/j.fsidi.2025.301982.

Visti, H., Tohill, S., Douglas, P., 2015. In: Automatic creation of computer forensic test images. Springer, pp. 163–175. https://doi.org/10.1007/978-3-319-20125-2_14.

Voigt, L.L., Freiling, F., Hargreaves, C.J., 2024. Re-imagen: generating coherent background activity in synthetic scenario-based forensic datasets using large language models. Forensic Sci. Int.: Digit. Invest. 50, 301805. https://doi.org/10.1016/j.fsidi.2024.301805.

Wickramasekara, A., Breitinger, F., Scanlon, M., 2025. Exploring the potential of large language models for improving digital forensic investigation efficiency. Forensic Sci. Int.: Digit. Invest. 52, 301859. https://doi.org/10.48550/arXiv.2402.19366.

Wickramasekara, A., Scanlon, M.. A framework for integrated digital forensic investigation employing AutoGen AI agents. https://doi.org/10.1109/ISDFS60797.2024.10527235.

Wolf, D., Baier, H., 2025. Bringing AI into ForTrace++–A framework for automatic data synthesis. In: 15th SPRING Graduate Workshop, p. 29.

Wolf, D., Göbel, T., Baier, H., 2024. Hypervisor-based data synthesis: on its potential to tackle the curse of client-side agent remnants in forensic image generation. Forensic Sci. Int.: Digit. Invest. 48, 301690. https://doi.org/10.1016/j.fsidi.2023.301690.

Woods, K., Lee, C.A., Garfinkel, S., Dittrich, D., Russell, A., Kearton, K., 2011. Creating Realistic Corpora for Security and Forensic Education. AFDSL.