

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**MASTER OF SCIENCE IN SIGNAL THEORY
AND COMMUNICATIONS**

**BEHAVIORAL ANALYSIS IN SECURITY
CONTEXTS BASED ON LOCATION DATA
AND MOVEMENT AND LANGUAGE
MODELS**

LORENA QUESADA ILMERT

2025

MASTER OF SCIENCE IN SIGNAL THEORY AND COMMUNICATIONS

MASTER'S THESIS

Title: Behavioral analysis in security contexts based on location data and movement and language models

Author: Lorena Quesada Ilmert

Technical Supervisor: Ana María Bernardos Barbolla

Academic Supervisor: Ana María Bernardos Barbolla

Department: Signals, Systems and Radiocommunications

MEMBERS OF THE EXAMINATION COMMITTEE

President:

Member:

Secretary:

Substitute:

The above-named members of the panel agree to grant the qualification of:

Madrid, a de de 20...

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**MASTER OF SCIENCE IN SIGNAL THEORY
AND COMMUNICATIONS**

**BEHAVIORAL ANALYSIS IN SECURITY
CONTEXTS BASED ON LOCATION DATA
AND MOVEMENT AND LANGUAGE MODELS**

**LORENA QUESADA ILMERT
2025**

SUMMARY

The growth in location-based services has contributed to the increase of georeferenced data generation and applications. In the scope of the Bachelor's thesis, a system was proposed to leverage this data through two of the main applications of interest: behavioral modeling and anomaly detection. The approach of behavioral modeling in georeferenced data aims at extracting patterns of interest that help explain the underlying preferences and conducts that explain movement. On the other hand, the approach of anomaly detection on georeferenced data aims at identifying deviations of interest from established behavioral patterns that might signify relevant changes in the environment or conduct of an individual. Meanwhile, another trend has emerged in the literature tied to the growth in popularity of language-based solutions. Natural Language Processing (NLP) techniques have evolved into complex models and architectures that enable the automation of tasks which used to require human intervention. Large Language Models (LLMs) have established themselves as the forefront of development in the area of natural language processing, generation, and interpretation, with most applications in the field designed to exploit and leverage the capabilities of these models through their integration in complex architectures, like retrieval-augmented generation (RAG), and complex systems, like AI agentic systems.

In this context, this Master's thesis proposes a framework to analyze, integrate, and interpret georeferenced data against a series of contextual sources to derive useful and understandable conclusions in a security context. The framework integrates the developed behavior analysis and anomaly detection system designed as part of the Bachelor's thesis with additional processing modules that implement language-based solutions to contextualize and give meaning to the data in a broader context. Firstly, a data contextualization and interpretation module is proposed, which takes the conclusions reached by the previously developed system and situates them within a broader police investigation setting by providing significance to them through both external and internal contextual information. External information is gathered through an automated goal-based AI agent system, which extracts relevant context in historic external knowledge, while internal information is leveraged through a retrieval-augmented generation architecture, which engineers prompts for the language model that carries out the interpretation task to consider the relevant context. Secondly, a data generation and movement prediction module is also proposed, which integrates a movement simulator developed in the Bachelor's thesis with language models to predict future movements based on the patterns mentioned and conclusions reached by the previous module. Finally, a police report simulation module is proposed to tackle the limitations of available police-relevant information and documentation through the generation of synthetic reports with the use of LLMs.

Furthermore, a novel evaluation framework is designed to overcome the lack of standardized validation approaches for language-based systems. Under this framework, the results obtained are evaluated through a domain-relevant evaluation scenario. It is concluded that the modules in the framework can properly correlate the data given and draw accurate and well-founded conclusions, especially as more information is available. The results obtained allow us to affirm that the framework could be implemented in real scenarios.

RESUMEN

El crecimiento de los servicios basados en la localización ha contribuido al aumento de la generación y las aplicaciones de datos georreferenciados. En el marco del TFG, se propuso un sistema para aprovechar estos datos a través de dos de las principales aplicaciones de interés: el modelado del comportamiento y la detección de anomalías. El enfoque del modelado del comportamiento en datos georreferenciados tiene como objetivo extraer patrones de interés que ayuden a explicar las preferencias y conductas subyacentes que explican el movimiento. Por otro lado, el enfoque de la detección de anomalías en datos georreferenciados tiene como objetivo identificar desviaciones de interés de los patrones de comportamiento establecidos que podrían significar cambios relevantes en el entorno o la conducta de un individuo. Mientras tanto, ha surgido otra tendencia en la literatura relacionada con el crecimiento en popularidad de las soluciones basadas en el lenguaje. Las técnicas de procesamiento del lenguaje natural (NLP) han evolucionado hasta convertirse en modelos y arquitecturas complejos que permiten la automatización de tareas que antes requerían la intervención humana. Los modelos de lenguaje grandes (LLM) se han consolidado como la vanguardia del desarrollo en el área del procesamiento, la generación y la interpretación del lenguaje natural, con la mayoría de las aplicaciones en este campo diseñadas para explotar y aprovechar las capacidades de estos modelos mediante su integración en arquitecturas complejas, como la generación aumentada por recuperación (RAG), y sistemas complejos, como los sistemas agenticos de IA.

En este contexto, este Trabajo de Fin de Máster propone un marco para analizar, integrar e interpretar datos georreferenciados en relación con una serie de fuentes contextuales con el fin de obtener conclusiones útiles y comprensibles en un contexto de seguridad. El marco integra el sistema de análisis de comportamiento y detección de anomalías desarrollado en el TFG con módulos de procesamiento adicionales que implementan soluciones basadas en el lenguaje para contextualizar y dar significado a los datos en un contexto más amplio. En primer lugar, se propone un módulo de contextualización e interpretación de datos, que toma las conclusiones alcanzadas por el sistema desarrollado anteriormente y las sitúa en un entorno de investigación policial concreto, dotándolas de significado a través de información contextual tanto externa como interna. La información externa se recopila a través de un sistema automatizado de agentes de IA basado en objetivos, que extrae el contexto relevante del conocimiento externo histórico, mientras que la información interna se aprovecha a través de una arquitectura de generación aumentada por recuperación, que diseña indicaciones para el modelo de lenguaje que lleva a cabo la tarea de interpretación para tener en cuenta el contexto relevante. En segundo lugar, también se propone un módulo de generación de datos y predicción de movimientos, que integra un simulador de movimientos desarrollado en el TFG con modelos de lenguaje para predecir movimientos futuros basándose en los patrones mencionados y las conclusiones alcanzadas por el módulo anterior. Por último, se propone un módulo de simulación de informes policiales para abordar las limitaciones de la información y la documentación disponibles relacionadas con la policía mediante la generación de informes sintéticos con el uso de LLMs.

Además, se ha diseñado un nuevo marco de evaluación para abordar la falta de enfoques de validación estandarizados para los sistemas basados en el lenguaje. En este marco, los resultados obtenidos se evalúan mediante un escenario de evaluación relevante en el dominio de aplicación. Se concluye que los módulos del marco pueden correlacionar adecuadamente los datos proporcionados y extraer conclusiones precisas y bien fundadas, especialmente cuando se dispone de más información. Los resultados obtenidos nos permiten afirmar que el marco podría implementarse en escenarios reales.

KEYWORDS

Large language models (LLMs), retrieval-augmented generation (RAG) architectures, AI agentic systems, contextualization, data integration, data simulation, georeferenced data, anomaly detection, behavioral model, security

PALABRAS CLAVE

Modelos de lenguaje de gran tamaño (LLMs), arquitecturas de generación aumentada por recuperación (RAG), sistemas agenticos de IA, contextualización, integración de datos, simulación de datos, datos georreferenciados, detección de anomalías, modelo de comportamiento, seguridad

AGRADECIMIENTOS

A mi familia, como siempre, por todo el apoyo y amor incondicional que me dais. Por haberme ayudado a llegar hasta aquí, y por apoyarme en mis decisiones futuras. Por animarme y cuidarme cuando lo necesitaba, y por siempre confiar en mí. Gracias a vosotros tengo la confianza de seguir mi camino y de seguir aprendiendo. Espero poder seguir demostrando lo mucho que os quiero y aprecio, y que este trabajo sea una pequeña representación de todos los recursos, amor y el cariño que me habéis dado toda mi vida.

A mi tutora Ana Bernardos, por todo lo que me has enseñado, por la paciencia que me das, y la confianza que pones en mí. Gracias por haber hecho posible este trabajo y por siempre empujarme a mejorar. Gracias por permitirme crecer estos últimos años, y gracias por darme la oportunidad de seguir haciéndolo bajo tu tutela. Miro a los siguientes años con entusiasmo gracias a los últimos tres trabajando contigo.

A José Ramón, por no solo haberme dado la oportunidad de estar aquí, sino por permitirme continuar cumpliendo mis metas en este grupo. Gracias por ofrecerme esta nueva oportunidad y por haberme guiado hasta ella. Y gracias por haberme dado la oportunidad de formar parte de este grupo hace tres años, he crecido y aprendido mucho aquí, y espero poder seguir haciéndolo en los siguientes años.

A mis compañeros del GPDS, con los que estoy muy contenta de empezar esta nueva etapa de mi carrera profesional. Gracias por los comentarios y las risas, por los cafés a los que casi siempre faltó, y las salidas después del trabajo. Espero que los siguientes años juntos sean igual, y estoy emocionada por compartíroslos con vosotros.

A todas las personas me siguen apoyando día a día. Gracias a Lucía, Miriam, y Yenny. Habéis estado ahí desde el principio, soy quién soy gracias a vosotras y no hay nada que me hace más feliz que sigamos creciendo juntas. Gracias a María Pilar, por compartir esta experiencia del máster conmigo, por celebrar las buenas notas, y sufrir las madrugadas juntas. Gracias a Pablo, por tratar con sus comentarios que el estrés del máster no pudiese conmigo, incluso si no siempre funcionaba. Gracias a todos mis profesores, que tuvieron paciencia enseñando y me han permitido aprender todo lo que se a día de hoy. Gracias a mis compañeros de la delegación, se cierra una etapa para mí, pero habéis marcado mi carrera universitaria de principio a fin, y siempre lo recordaré con cariño.

Y finalmente, gracias a Álvaro. Gracias por confiar en mí y en lo que era capaz de hacer, incluso cuando yo no podía. Gracias por cuidarme y por siempre estar ahí. Este trabajo va dedicado a ti y a todo el cariño que me diste para que pudiese llegar aquí.

Muchas gracias.

TABLE OF CONTENTS

1. INTRODUCTION AND OBJECTIVES.....	1
1.1. INTRODUCTION	1
1.2. OBJECTIVES.....	2
1.3. WORK METHODOLOGY	2
1.4. DOCUMENT STRUCTURE.....	3
2. RELATED WORK.....	4
2.1. LANGUAGE MODELS FOR GENERATION, CONTEXTUALIZATION AND REASONING	4
2.1.1. Language models for advanced text generation and synthesis	4
2.1.2. Context-aware reasoning in large language models.....	6
2.2. ADVANCED PARADIGMS FOR LEVERAGING LANGUAGE MODELS.....	9
2.2.1. The Retrieval-Augmented Generation (RAG) paradigm.....	9
2.2.2. The autonomous agent paradigm powered by language models.....	12
2.3. VALIDATION AND EVALUATION APPROACHES FOR LANGUAGE-BASED SYSTEMS.....	13
3. DESIGN OF A UNIFIED FRAMEWORK FOR DATA CONTEXTUALIZATION AND PREDICTION BASED ON A BEHAVIOR ANALYSIS FRAMEWORK.....	16
3.1. PROBLEM STATEMENT	17
3.2. FRAMEWORK OVERVIEW	18
4. A RETROSPECTIVE REVIEW OF A SYSTEM FOR BEHAVIORAL ANALYSIS AND ANOMALY DETECTION.....	22
4.1. DATA PREPROCESSING AND HYPERPARAMETER EXTRACTION MODULE	23
4.1.1. Data reformatting step.....	23
4.1.2. Hyperparameter extraction step	24
4.2. BEHAVIORAL MODEL BUILDING MODULE	26
4.2.1. Places of interest extraction step	26
4.2.2. Behavioral patterns extraction step	28
4.3. ANOMALY DETECTION AND SCORING MODULE.....	29
4.3.1. Anomaly detection step.....	30
4.3.2. Anomaly labeling step	33
5. DESIGN AND IMPLEMENTATION OF A SIMULATOR MODULE FOR SYNTHETIC POLICE REPORT GENERATION	34
5.1. CRIME EXTRACTION AND GROUPING COMPONENT	35
5.1.1. Crime extraction step	36
5.1.2. Crime grouping and categorizing step	37

5.2.	TIMELINE BUILDING COMPONENT.....	39
5.2.1.	Movement timeline extraction step.....	40
5.2.2.	Crime timeline definition step.....	40
5.3.	REPORT GENERATION COMPONENT.....	42
6.	DESIGN AND IMPLEMENTATION OF A DATA CONTEXTUALIZATION AND INTERPRETATION MODULE.	47
6.1.	DATA INTEGRATION COMPONENT.....	48
6.1.1.	Data contextualization and normalization step	49
6.1.2.	Data vectorization and integration step.....	56
6.2.	RETRIEVAL-AUGMENTED GENERATION COMPONENT	58
6.2.1.	Data retrieval step	59
6.2.2.	Prompt engineering step.....	61
6.3.	REPORT GENERATION COMPONENT.....	62
6.3.1.	Report generation branch	63
6.3.2.	Chatbot interface deployment branch	64
7.	DEVELOPMENT OF A DATA GENERATION AND MOVEMENT PREDICTION MODULE	67
7.1.	A BRIEF OVERVIEW OF THE MOVEMENT SIMULATOR.....	68
7.2.	REPORT ANALYSIS COMPONENT.....	70
7.2.1.	Parameter extraction step	70
7.2.2.	Parameter structuring step.....	72
8.	RESULTS	73
8.1.	EVALUATION SCENARIO.....	73
8.2.	EVALUATION METHODOLOGY.....	75
8.2.1.	Content-based evaluation methodology	76
8.2.2.	Fact-based evaluation methodology.....	81
8.3.	EVALUATION OF RESULTS OBTAINED.....	84
8.3.1.	Content-based results	84
8.3.2.	Fact-based results.....	85
9.	CONCLUSIONS AND FUTURE WORKS	87
9.1.	CONCLUSIONS.....	87
9.2.	FUTURE WORKS.....	88
10.	BIBLIOGRAPHY	90
	APPENDIX A: ETHICAL, ECONOMIC, SOCIETAL AND ENVIRONMENTAL ASPECT	98
A.1	INTRODUCTION	98
A.2	DESCRIPTION OF RELEVANT IMPACTS RELATED TO THE PROJECT	98

A.3 ETHICAL AND LEGAL REQUIREMENTS AND IMPACTS OF THE PROPOSED FRAMEWORK.....	99
A.4 CONCLUSIONS.....	102
APPENDIX B: FINANCIAL BUDGET	103

LIST OF FIGURES

Figure 1. Proposed framework architecture	18
Figure 2. Overview of software pipeline with I/O of each implemented task	20
Figure 3. Behavioral modelling and anomaly detection system architecture	22
Figure 4. Structure of the data preprocessing and hyperparameter extraction module within the behavioral modelling and anomaly detection system	23
Figure 5. Structure of the behavioral model building module within the behavioral modelling and anomaly detection system	26
Figure 6. Structure of the anomaly detection and scoring module within the behavioral modelling and anomaly detection system	30
Figure 7. Police report simulation module architecture	34
Figure 8. Structure of the crime extraction and grouping component within the report simulation module	35
Figure 9. Structure of the timeline building component within the report simulation module	40
Figure 10. Architecture of the dynamic programming algorithm implemented in the timeline building component	42
Figure 11. Structure of the report generation component within the report simulation module	43
Figure 12. Example of generated report through the report simulation module	43
Figure 13. Prompt to generate the information in the 'Person(s) Involved' section of the synthetic report	44
Figure 14. Prompt to generate the information in the 'Witness(es)' section of the synthetic report	44
Figure 15. Prompt to generate the information in the 'Incident Details' section of the synthetic report	44
Figure 16. Prompt to generate the information regarding the injuries in the 'Damages and Injuries' section of the synthetic report	45
Figure 17. Prompt to generate the information regarding the damages in the 'Damages and Injuries' section of the synthetic report	45
Figure 18. Prompt to generate the information in the 'Actions Taken' section of the synthetic report	45
Figure 19. Data contextualization and interpretation module architecture	47
Figure 20. Structure of the data integration component within the data contextualization and interpretation module	49
Figure 21. Contextualizer AI agent node-based architecture	51
Figure 22. Instructions to indicate the AI agent how to generate the web search query	53
Figure 23. Architecture of the metasearch engine implemented in the AI agent	54
Figure 24. Instructions to indicate the AI agent how to summarize the search results	55
Figure 25. Instructions to indicate the AI agent how to reflect on the summarized results	56
Figure 26. Structure of the retrieval-augmented generation component within the data contextualization and interpretation module	59
Figure 27. Structure of the report generation component within the data contextualization and interpretation module	63
Figure 28. Structural example of final report generated	64
Figure 29. Data generation and movement prediction module architecture	67
Figure 30. Architecture of the previously developed simulator implemented in the data generation and movement prediction module	68
Figure 31. Structure of the report analysis component within the data generation and movement simulation module	70
Figure 32. Evaluation timeline with real events and investigative events	73
Figure 33. Content-based evaluation framework	76
Figure 34. Fact-based evaluation framework	81

LIST OF TABLES

Table 1. External tools used to store the results of the behavior analysis and anomaly detection system	23
Table 2. Existing tools used in the development of the crime clustering and categorization step	37
Table 3. Existing tools used in the implementation of the report generation task	46
Table 4. Existing tools used in the development of the contextualizer AI agent	52
Table 5. Existing tools used in the implementation of the data embedding and storage task	57
Table 6. Existing tools used in the implementation of the knowledge integration task	61
Table 7. Existing tools used in the implementation of the prompt engineering task	62
Table 8. Existing tools used in the implementation of the chatbot interface branch	65
Table 9. Existing tools used in the implementation of the movement simulator	69
Table 10. Existing tools used in the implementation of the movement profile parameters extraction task	71
Table 11. Mention thresholds for topic detection in each of the phases of the evaluation scenario	77
Table 12. Detail types for topic detection in each of the phases of the evaluation scenario	78
Table 13. Metrics to carry out the content-based evaluation	80
Table 14. Importance weights for topic evaluation in each of the phases of the evaluation scenario ..	80
Table 15. Metrics to carry out the fact-based evaluation	84
Table 16. Content-based report evaluation results	84
Table 17. Fact-based report evaluation results	85
Table 18. Analysis of the impacts of the developed framework in social, ethical, legal, and environments contexts	98
Table 19. Project's financial budget	103

LIST OF ABBREVIATIONS

NLP: Natural Language Processing

LLM: Large Language Model

RAG: Retrieval-Augmented Generation

AI: Artificial Intelligence

GPT: Generative Pretrained Transformer

1. INTRODUCTION AND OBJECTIVES

1.1. INTRODUCTION

An increase in the relevance of location-based services in the last decade has played a crucial role in the parallel growth in the generation and study of georeferenced data. Many fields of study that had not traditionally relied on this type of data have started to adopt it, including fields like agronomy and modern farming [1], marketing [2], tourism [3], or healthcare and epidemiology studies [4], and many of the ones who always considered this kind of data are now starting to rely on technological advances that increase the efficiency of georeferenced data exploitation, like traffic and urban planning [5], emergency evacuation studies [6], and of particular interest, the field of law enforcement and criminal justice. Out of the many interesting applications the leveraging of georeferenced data brings forward, some stand out as being of special interest in the field of police work. Starting with behavioral modeling, this application poses an interesting approach to the study of human movement patterns and conducts. These kinds of models, which summarize a series of spatial and temporal characteristics that define an individual's preferences when moving in a certain environment, provide a way for mobility data to be further understood and processed. This provides a basis for decision-making based on movement data.

Furthermore, behavioral modeling gives rise to another application of interest, which is anomaly detection on movement and behavior-based data. In this context, anomalies can be understood as patterns of behavior that deviate from the expected conducts established by the model. Anomaly detection offers interesting perspectives in the mentioned field as the identification of outliers and data instances that do not follow an established norm might indicate relevant changes in circumstances or behavior which should affect decision-making. However, all the applications steaming from the exploitation of georeferenced data often must tackle the limitation of reduced interpretability, which makes it difficult for humans to fully grasp their meaning without additional context.

Along similar lines as the rise of georeferenced data, language-based systems and models are becoming an uprising tool in the field of artificial intelligence and machine learning, as they provide the means to enhance explainability and coherence in otherwise complex and sometimes black-box models. From finance [7] to healthcare [8], the use of language-based systems allow data, as well as the insights derived from complex systems, to be better understood by humans, facilitating transparency and responsibility. Furthermore, tools like large language models (LLMs) allow to not only generate and process text-based data, but also to understand and reason about text, making processes based on the interpretation of text more efficient. These LLMs have also become a crucial factor in the development and sophistication of AI agents, which are autonomous systems which have also seen a growth in interest in recent years. Agentic AI systems allow to automatize many costly or time-consuming tasks through the deployment of autonomous agents that can self-supervise a series of given tasks. This technology combined with LLM helps to improve the contextualization, explanation and interpretation of data and models.

The intersection of both growing fields of study, georeferenced data exploitation and language-based system, presents applications of interest. Language models applied to mobility data could help us contextualize the nuances given by behavioral models in multiple scenarios, including crime prediction through spatial crime analysis [9]. LLMs could also help to explain patterns detected in mobility data, with the goal of improving future pattern prediction, and decision-making [10]. These language models also allow to better understand anomalies, and in many cases, to detect them as well [11]. Furthermore, the combination of AI agents and LLMs could allow the development of efficient systems that integrate knowledge on human mobility, which helps better process and predict data [12].

In this line of thought, the work presented in this Master's Project is aimed at designing and implementing the prototype for a framework that combines a series of data processing and language-based techniques to analyze and interpret relevant data in a criminal justice use case through the integration of a behavioral analysis and anomaly detection system for georeferenced data developed in

the Bachelor's thesis with a series of additional processing modules that integrate LLMs and a series of LLM-drive tools and architectures. The main goal of this framework is to properly analyze georeferenced data and integrate it with a series of contextual data sources to properly carry out context-aware interpretations and reach well-founded conclusions to be presented to the user in a structured, coherent, and understandable way.

1.2.OBJECTIVES

As previously mentioned, the main goal of this project is to define and implement a framework prototype aimed at leveraging some of the data sources used by police forces, especially those which are location-based, in the context of a police investigation to assist in data processing tasks. The framework is aimed at efficiently analyzing the available data to draw useful conclusions and identify possible correlations between them, and presenting the findings in an understandable and readable way, to facilitate human decision-making. To carry this out, a series of sub objectives are defined to be tackled as follows:

- **O1. Improvement of an existing behavioral modelling and anomaly detection system** developed under the same line of work with the goal of ensuring a smoother interoperability with additional processing modules in the larger, overarching final developed framework. The improvements must ensure that other modules can easily access conclusions reached by the system, that said conclusions are readable and interpretable by the modules, and that the system can take as input the output of other modules as well, being able to tackle multiple data formats.
- **O2. Design and implementation of a data contextualization architecture** which enhances context-dependent and preliminary data that lack useful external information needed for better human interpretation and understanding of its significance in the context that surrounds it. The architecture must be able to take multiple data sources as input (conclusions reached by system improved in O1, like location-based data, behavioral model, and anomalies) and contextualize each accordingly with relevant external information to the use case.
- **O3. Design and implementation of a data interpretation architecture** which can present the results reached by all the systems in an understandable, structured, and transparent way that allow the human user to make informed decision-making through the use of language models and language-based techniques.
- **O4. Development of a tool for police report simulation** to generate synthetic, realistic, and structured police reports to properly validate the final framework considering all the real available data sources the police forces have. The tool should be able to generate reports that could be used for the contextualization tasks presented in O2 and the interpretation tasks presented in O3.
- **O5. Design of a movement prediction module** that combines a movement data simulator developed under the same line of work with additional report analysis techniques to define a movement profile based on previous conclusions reached on a suspect. The module must be able to extract all the information needed by the movement simulator from reports gathered from a certain suspect making use of language-based techniques.
- **O6. Design of novel techniques for language model validation** for the correct evaluation of the results obtained by architectures implemented in the previous objectives. It must combine known validation techniques with additional mechanisms and rules to ensure that the information given to the final user is truthful and coherent.
- **O7. Definition of a data analysis and interpretation framework** through the integration of a movement data analysis system and additional data contextualization, interpretation, and prediction modules.

1.3.WORK METHODOLOGY

This project has been developed following a somewhat linear lifecycle, with phases of approximately one month of duration. Each of these phases focused on the design and development of a different component of the final project, with the results of one of the phases often being necessary for the

development of the work contained in the next phase. Therefore, the overall development of this work has been sequential in nature, except for phases 4, 9 and 10 which were developed iteratively, with periodic validations to ensure the correct integration of the final prototype. The phases of development of this work include the following: 1) Review of the state of the art, 2) Improvement of behavior modeling and anomaly detection system, 3) Design and development of a report simulation module, 4) Design of evaluation methodology, 5) Development of contextualizer agents, 6) Design and development of the data contextualization and interpretation module, 7) Design and development of the data prediction module, 8) Integration of the complete prototype, 9) Testing and evaluation, and 10) Drafting of document.

All the code included in this project has been developed using Python as programming language. Additionally, some already trained and developed models were used, including mainly open-source large language models including Qwen2.5-14B, Mistral 7B, and Llama3.2. Furthermore, some existing frameworks were used for the development of agents and chatbots, including LangChain and LangGraph. Some external engines were deployed to carry out tasks, including an OSRM (Open-Source Routing Machine) engine needed for the behavior modeling and anomaly detection system, as well as the movement simulator, and a metasearch SearXNG engine needed for the AI agents developed for the data contextualization and interpretation module. All the code and the engines have been designed, developed, and deployed accordingly as weaknesses, errors and limitations arose in each of the phases.

All the modules and systems developed under the scope of this project have been thoroughly documented both on the code and on additional reports for future possible use by other developers. Moreover, all the architecture layouts designed, processes followed, and decisions made along the development of this work have also been documented. Finally, this document has been developed containing all the relevant and most important ideas, processes and conclusions reached in the lifecycle of this project.

1.4.DOCUMENT STRUCTURE

This project document is arranged following a section-based structure which divides the work done as follows. In Section 2, an in-depth analysis is done on the current state of the literature regarding the tools used to develop the different modules that make up the framework presented. Section 3 gives a general overview on the problem tackled in this work and what the proposed solution entails. Furthermore, Section 4 makes a retrospective review on the existing behavior modeling and anomaly detection system, giving an overview rundown of how the system works, the inputs and outputs it expects, and the slight changes made for its improvement and integration into the framework. Section 5 presents the police report simulator developed and its integration in the framework. Sections 6 and 7 present the main two modules developed to enhance the anomaly detection system and build the final framework, which are the data contextualization and interpretation module and the data prediction module respectively. Firstly, Section 6 details the technical design of the contextualization and interpretation, and its implementation with existing and newly developed tools. Secondly, Section 7 details the integration of the existing movement simulator with additional processing tools for the prediction of possible future movements based on past written reports. Finally, Section 8 introduces the developed evaluation approaches and evaluation scenario, and presents the results obtained. Section 9 goes over the conclusions reached in this work and the future work to be developed. Information regarding the impact of this project in various fields, as well as the budget needed to finance it is presented in the two final annexes included in this document.

2. RELATED WORK

2.1. LANGUAGE MODELS FOR GENERATION, CONTEXTUALIZATION AND REASONING

2.1.1. LANGUAGE MODELS FOR ADVANCED TEXT GENERATION AND SYNTHESIS

In the landscape of language model applications, text generation is established as the core task which has quickly become the backbone of most language-based solutions. This language model application is based on the idea of creating coherent sequences of text, often based on a given context or prompt, to fulfill a purpose dependent on the application of the model, which can range from content generation to conversational chatbots. In recent years, text generation has expanded across a variety of fields of study like healthcare [13] [14] or education [15] [16]. In the context of this work, text generation through the usage of language models has also become part of many security and crime analysis-oriented solutions [17] [18]. For this work, text generation is fundamental, as one of the main goals is to enhance the interpretability of a movement data analysis system by generating written explanations that are coherent and easily understandable by humans.

Due to the relevance of this application, most language models, especially the large language models (LLMs), have text generation capabilities. However, while most models can generate simple text, certain architectures are specifically aimed at efficient, coherent, and context-relevant text generation, with the most significant and wide-spread architecture being the decoder-only architecture, popularized by OpenAI's Generative Pretrained Transformer (GPT) models [19]. Since its first formal introduction in 2018, the decoder-only architecture has become the base of most language models aimed at fluent text generation due to its autoregressive processing approach, where words in a text are generated through predictions based on previous content. Although other existing approaches have also made a staple in the text generation field, with the most popular alternative being encoder-decoder architectures based on sequence-to-sequence transformations where text is corrupted and reconstructed, with known model families like T5 [20] by Google Research or BART [21] by Meta, decoder-only architectures continue to prove their leadership in text generation tasks due to their scalability, uniformity, and generality [22] [23]. Nevertheless, within the decoder-only architecture scope, many model families have emerged through the years, each with different approaches in training, architectural optimization, and efficiency.

OpenAI's GPT models, as mentioned, were the pioneers in decoder-only architecture, being the first to break through many limitations in the text generation field. Due to OpenAI resources, GPT models were the first to implement pretraining on a major scale, with posterior supervised fine-tuning, which in recent years has developed into zero-shot learning (introduced in GPT-2 [24]), in-context learning (introduced in GPT-3 [25]), and reinforcement learning from human feedback (introduced in GPT-4 [26]). These progressive advancements have yielded current GPT models that are capable of efficient and scalable coherent text generation, with widespread implementations like ChatGPT. The reputation of these models has also led to their implementation and study in many areas of investigation. Authors like Meduri et al. [13] or Haupt and Marks [27] aim their works at the analysis of the possible implementation of GPT models in the field of healthcare. Meduri et al. [13] argues that GPT models can enhance healthcare intelligence as an additional tool for the medical professionals, for improvement in decision-making through the integration of various data sources for better understanding of the evolving healthcare field, the improvement of diagnosis accuracy, and the generation of tailored treatment plans for patients. Haupt and Marks [27] explore the value of GPT models in healthcare from the both the health professional and the patient's point of view as a possible tool for medical assistance in tasks like medical report generation and preliminary symptoms diagnosis. In the field of education, many authors have also studied the implication of the use of GPT models, with many evaluating the possible benefits, with works like [15] proposing applications like educational content generation to improve the understanding of the topics, personalized guidance to help with studying approaches, and generation of test simulations for practice study and critical thinking enhancements. GPT models have also started to have a reduced but noticeable growth in

implementation is fields of security and crime investigation, with works evaluating the performance of these models in often manual police tasks, ranging from the generation of police reports and investigation hypothesis [17] to the evaluation of crimes [28], both implementations of interest, especially in the context of this work.

In a similar line of development, with the idea of large-scale preprocessing and commercial use, Google DeepMind's Gemini model family has become relevant in the text generation space, even though it was born as a multimodal foundation model. Since the first introduction of a Gemini model in 2023 [29] as a direct successor of other Google DeepMind models, like LaMDA [30] and PaLM [31], this model has stand out in the field due to its multimodal nature and its large context window, which make it suitable for implementations that require extensive and complex context. The mentioned multimodal nature Gemini for text generation is often seen reflected in the diversity of the fields of study which implement this model. In the work developed by Kadiyala et al. [32], the Gemini models are used in the hydraulic energy industry to generate precise and relevant responses regarding the analysis and evaluation of hydrologic data, including data regarding floods, water level monitoring, water contamination managing and agricultural water unloading. The text generation capabilities of the Gemini models are used in this case to generate detailed and contextualized descriptions of the data. Other fields, like agriculture and farming, also leverage Gemini models, in applications like the one presented by Samuel et al. [33], where the language models are used as a helping tool for crop management, pest control, agricultural practices, and soil analysis and optimization. The text generation capability of the Gemini models in this case is used to respond questions by farmers, generate explications with techniques and recommendations, and help in decision-making tasks. Aside from the multimodal nature and the large context window, models under the Gemini family introduce some architectural optimizations, like multi-query attention and mixture-of-experts approaches, which also stem into developments towards a more efficient and competitive performances. This can be reflected in certain implementations like the ones seen for GPT models, where even with a 5-year gap between the development of both model families, the performance of the Gemini models rivals those of GPT models. An example of this can be seen in the work developed by Saab et al. [34], where a medicine-oriented Gemini model is developed with the goal of generating long-form medical texts, evaluated for tasks like after-visit clinical summaries, doctor letter generation, and medical text simplification. Another example, in the field of education, is the work presented by Imran and Almusharraf [16], Gemini is analyzed as a tool with the capabilities of having high and useful functionalities in education through the generation of lesson plans, quizzes and summaries.

Nevertheless, the high proficiency of models from families like GPT or Gemini in the text generation task often leads to slight limitations, mainly around hallucinations, when the model generates incorrect information. Furthermore, their complex structure and capabilities have resulted in the models being closed-source, which is a high disadvantage in investigation fields, as the restrictions limit the development of capabilities and new applications of these models. Therefore, model families like the ones mentioned until now are necessary to be studied to better understand language models that stem from them, as they have established the architectural base and foundations for text generation, but they are not often implemented in research-oriented studies, which often make use of open-source language models for text generation.

Open-source models often deal with performance limitations, as training and development resources are equally limited, but the availability of these models has garnered them attention, especially from researchers. This attention has attributed to the recognition of some open-source model families, from which one of the most widespread ones is Meta AI's LLaMa model family. LLaMa models, even in their first introduction [35], surpassed the performance of complex, bigger, and closed-source models like GPT-3, quickly becoming a free of use alternative with an equal high-performance. With time, LLaMa models have improved into allowing more context (introduced in LLaMa 3 [36]), adding Mixture-of-models designs (introduced in LLaMa 4 [37]), and even the usage of Interleaved Rotary Position Embeddings (iRoPE) to better tackle huge amounts of contextual input. Thus, currently, the LLaMa model family is one of the main open-source alternatives due to its capabilities for easy and efficient deployment, and the competitive capabilities it presents. Therefore, the text generation

capabilities of these models have applications in fields ranging from cybersecurity and software engineering [38] to e-commerce [39] to even the legal domain [40]. Many authors focus their implementation of LLaMa models to the field of healthcare, as with the model families seen before. Wang et al. [41] presents a work in the healthcare field which makes use of LLaMa models for text generation. The main goal of this text generation implementation is to generate quick and useful responses to medical questions, but the text generation capability is further used in the development of this work for the generation of training data for validation, showcasing its usefulness in various steps of the development cycle. Nevertheless, although this model family is useful in a big variety of implementations, a case study of interest in the development of this work is the use of LLaMa models in the context of security and crime investigations. Works like the one presented by Xing and Chen [18] aim the text generation capabilities at the analysis of police reports, the extraction of entities of interest from the data, and the generation of associated explanations. Others, like the one developed by Michelet and Breitingner [42], aim text generation at more typical tasks, like police report generations, by giving the model prompts associated with each of the sections included in the report. Both implementations are of use in law enforcement tasks regarding the automatization of manual tasks and the enhancement of important data pieces for better decision-making.

Another model family which is open-source and of interest in the task of text generation is the Mistral family. Mistral AI language models have not yet completely closed the gap with the leading models in the community, but they are starting to slowly gain popularity and traction due to their text generation capabilities, easy implementations, and good results, starting to rival models in the Llama family and even the GPT family. With the first introduction of this model occurring in 2023 [43], the Mistral models have since then evolved into more complex structures, making use of approaches like the mixture-of-experts technique for higher performance, following in the steps of previously developed and more sophisticated models like Gemini. This has enabled Mistral models to progressively generate texts of better quality quickly and with computational efficiency. Therefore, many applications that require the quick generation of good quality texts often implement Mistral models. In the field of healthcare, for example, many implementations are more focused on generating quick and efficient notes rather than lengthy reports, like in many of the other implementations seen for previous models. For example, the work presented by Jung et al. [14] implements a Mistral model to generate medical discharge notes for patients with the goal of generating responses more efficiently and ensuring a continuity in the attention and care to the patients. Other quick implementations of interest of these kinds of models are presented by Ikonić Nešić et al. [44] under the investigative work in the field of literature and text analysis, where text generation is used to generate small classification labels associated to each of the segments of text extracted. However, the quickness of the text generation does not signify lower quality texts, as the Mistral models are optimized to generate complete and context-rich texts, useful in many applications, with the most interesting ones in our case being, once again, applications on security and crime analysis.

In the development of this work, text generation is mainly aimed at the task of generation reports, either for police report simulator, for data contextualization, or for data interpretation. Thus, given the capabilities of decoder-only architectures for this task, the goal is to implement open-source decoder-only models to generate text of different sizes using prompts and instructions, with the main considered models being from the LLaMa and Mistral families.

2.1.2. CONTEXT-AWARE REASONING IN LARGE LANGUAGE MODELS

With the growth in the complexity of language-based problems, there has been a parallel emergence of new and more complex capabilities that these models can perform, with one of the main ones being reasoning on text. Reasoning on text is based on the idea of connecting ideas, extracting relationships and correlation, and applying prior knowledge to get to a reasonable conclusion regarding the data. While most of the well-known language model applications are based on simply repeating known content, reasoning on text expects the model to analyze in depth the data and draw conclusions. This idea is carried out with the help of two additional capabilities of most language models. Firstly, the capability of the model to contextualize data, meaning the capacity to understand what the text is saying using context clues from the given prompt or conversations with the user. Secondly, the

capacity of language models of generating ‘thought sequences,’ which mimic real reasoning sequences, with the goal of better processing and understanding the data.

Reasoning on text, while not the initial goal behind the development of language model, has with time become as much of a relevant capability of LLMs as text generation, in many cases being implemented more often than the latter. Many applications based on language models nowadays are aimed at interpreting and contextualizing data, rather than generating it. Data interpretation through reasoning on text can be found in applications from different fields of study, ranging from traffic and urban planning [45] [46] to education [47] [48]. Even in the context of security and crime analysis of this work, most of the implementations fall under the reasoning on text label [49]. Since another main goal of this work is to properly interpret the information extracted from a previously developed behavior analysis and anomaly detection system and integrate it with other data sources to contextualize it, the task of reasoning on text is especially relevant for this work.

Since reasoning on text stems directly from text generation, as a continuation in the increasing complexity of language models, the landscape of language models oriented towards reasoning on text is quite like the one explored in the previous section. Thus, although reasoning-oriented models have more example of encoder-decoder architectures, with model families like Flan-T5 [50], like with text generation, the leader architecture in state-of-the-art reasoning models is the decoder-only architecture. Since the overwhelming architecture used is the same across models, the reasoning on text landscape is often analyzed through the perspective of the reasoning techniques used by the models. These reasoning techniques, also known as prompting techniques, are relevant in the landscape because they help understand the significance of the decoder-only architecture in reasoning tasks, since this architecture integrates with the techniques in a smoother manner than the encoder-decoder alternatives, and because they give an insight to how reasoning is implemented in certain models.

Few-shot learning, also known as in-context learning, was one of the first prompting approaches in the paradigm of reasoning on text, with its first introduction in 2020 [25]. The main idea behind few-shot learning was for language models to be able to learn a task and apply logic to solve it through prompts that contained enough information for no fine-tuning to be required. This information includes a description of the task to be carried out, and a few examples of what should be created, usually used as context. In the case of reasoning on text with few-shot learning, the prompt contains the necessary instructions and context for the language model to dynamically adapt to carry out the goal only with the information given.

The first model family to properly implement the few-shot learning approach was OpenAI’s GPT models, specifically models from GPT-3 onwards. The implementation of reasoning through few-shot learning in GPT models pioneered, as it was the first big language model to implement a task like reasoning without requiring additional fine-tuning. The implementation of this technique in these types of models can take three different approaches which include a zero-shot approach (instructions are given but examples not), a one-shot approach (instructions and one example are given in the prompt), and a few-shot approach (instructions and a few examples are given) but in all three cases, if examples are given, they are always input-output examples. Nevertheless, out of all the approaches, few-shot learning is often the one used in reasoning applications with these types of models. Works like the one presented by Huang and Wang [51] evaluate few-shot prompting in GPT-style models for logical reasoning tasks, like puzzle solving through textual reasoning, concluding that the performance for these types of tasks is good, but noticeably improves with fine-tuning. Other works, like the one developed by de Padua, Qureshi, and Karakaplan [52], implement few-shot prompting-based reasoning on a completely different task, focusing on the reasoning of questions to respond with context-aware answers, rather than on the task of simple textual reasoning. However, zero-shot learning, although less used, is still present in GPT reasoning applications, and for certain tasks it outperforms few-shot approaches, like the one presented by Bommarito et al. [53], where zero-shot is implemented without fine tuning and the model is still able to reason over complex topics like finances, ethics, and accounting.

Meta AI's LLaMa family has also become one of the main ones to implement this prompting approach for reasoning on text, often combining it with other techniques like the one we will see later. The reasoning approach was first introduced with the LLaMa 2 model, putting, once again, open-source models on the same performance level as closed-source models like the GPT ones previously mentioned. Nevertheless, LLaMa follows a similar structure of implementation, considering both the zero-shot and few-shot approaches, but it introduced an improvement regarding the implementation by GPT models, as it allowed for more context to be considered, often obtaining better results in reasoning task due to the larger size of context data. However, even with a similar implementation of the prompting technique, the landscape of applications differs greatly with that of GPT models, as zero-shot prompting is more often seen in the literature regarding LLaMa models for reasoning on text than few-shot prompting. For example, authors like Gruver et al. [54] bring forward interesting applications of zero-shot prompting with LLaMa models, like time series prediction, treating the chain of numeric values as a string of text, and the task of predicting the next number as a task of next word prediction through word reasoning. [55] follows a similar approach, but with next location prediction, based on the same principle of not considering examples (zero-shot) and translating the next location prediction task into a text reasoning and prediction task.

Finally, in the landscape of few-shot prompting approach, a new model family has started to arise in recent years, which is Alibaba's Qwen model family [56]. This language model family, although fairly new, is quickly becoming one of the most important ones for the task of reasoning on text due to its strong performances and its reasoning-oriented architecture, which make this model highly compatible with multiple prompting approaches, as we will see later. This model follows a similar idea as LLaMa, taking advantage of the paradigm already defined by GPT to implement few-shot learning, in this case also considering zero-shot, one-shot and few-shot approaches. Due to the model's recency, the applications of this model are few and far between, but they start to showcase useful and promising results. More specifically, they show high performances in both few-shot and zero-shot prompting approaches. However, works like the one presented by Mon, Fenollosa, and Lecumberri [57] carry out a performance comparison in the task of hate speech detection through text reasoning where they identify that the stronger approach is the few-shot prompting. This outperformance is further validated by the results of other works in the field, like for example the work by Marchitan, Creanga, and Dinu [58], where reasoning on text with few-shot prompting for emotion detection on text equals that of older and more robust models.

Nevertheless, although the few-shot prompting approach is easy to implement and yields good results, the limitations in complexity and robustness created the necessity of a more sophisticated prompting technique for the reasoning on text task. Out of all the emerging techniques, the one that has established itself as the most practical and widely used is the Chain-of-Thought (CoT) prompting approach [59]. The main idea behind this approach is for the language models to compute intermediate reasoning steps before reaching the conclusion (in a multi-step) fashion, rather than just responding with a given prompt with instructions as before. This approach often increments the accuracy of few-shot prompting, as it computes a multi-step reasoning, encouraging the model to learn the relationships and correlations within the text more in depth. One of the main reasons behind the effectiveness of this approach was its easy integration both with ideas presented in the few-shot prompting technique, with the definition of zero-shot CoT [60] which thinks step by step without considering any examples, and with the use of robust decoding tricks, with the definition of CoT with self-consistency [61] which implements several CoT samples and votes for the final answer among them.

Most of the language models, including the ones introduced for the few-shot approach, have the capability of implementing both approaches, often combining them for the reasoning on text task. Nonetheless, the implementation of these techniques, even though similar, is not the same between few-shot and CoT prompting. Starting with OpenAI's GPT models, CoT prompting was first introduced with the GPT-3 model, but became more popular with GPT-4 [26]. The approach is similar to the few-shot prompting, with zero-shot CoT and few-shot CoT being the main implementations for this prompting, each thinking step by step, but considering non or a few full step-by-step rationale examples. In this case, the number of implementations based on few-shot and zero-shot vary. On one

hand, some proposed works in this field of study, like the work developed by Guan et al. [62], implements few-shot learning for the analysis and reasoning over clinical texts. On the other hand, other examples aimed at basic text reasoning to solve questions proposed by the text (e.g. riddles), like the one presented by Zhang, Ahmed, and Martin [63], implement zero-shot CoT with the additional fine tuning to obtain better results. However, due to the complexity of these models, new CoT approaches are also developed, like the ones included in the work by Lu et al. [64], where two CoT methods are introduced for the task of predicting and interpreting outcomes from chemistry experiments. The changes done to the original CoT structure are aimed at both breaking down the chain into reasoning segments, to evaluate the performance of this approach, and to build the chain into a more monolithic, unified sequence, to evaluate the opposite implementation.

Meta AI's LLaMa models also have a chain-of-thought prompting approach, but this one is not as widely used. However, Alibaba's Qwen model family introduces interesting capabilities from its Qwen 2 model [65] forward, as these models can implement strong reasoning, with high-performance results. These capabilities are obtained through the combined use of zero-shot and few-shot CoT for simple tasks, and a combination of multiple samples and self-consistency for more complex and tougher tasks. Regarding the use of zero-shot CoT prompting, examples of implementations of interest can be seen in works like the one presented by Luo et al. [66], where a solution is implemented for a Qwen language model to make autonomous driving possible through chain reasoning considering abstract ideas like perception, knowledge, and decision-making. Thus, zero-shot CoT reasoning is implemented as a way of responding to certain driving situations and environments to mimic human conduction patterns. Approaches implement self-consistency, as mentioned, tend to be a bit more complex, like the solving of complex mathematical problems as introduced by Tahmid and Sarker [67], where self-consistency is implemented with other approaches (like tool-integrated reasoning) for the model to be able to self-verify the results generated and improve the accuracy before returning the final answer.

In the context of this project, text reasoning is necessary for data contextualization and integration, since multiple data sources will be available but with no established connections between them. Like with text generation approaches, the goal is to implement a language model aimed at reasoning on text which is open-source and can take context to respond to queries regarding data. Models like the ones introduced in the LLaMa and Qwen families are of interest for the implementation, due to their easy implementation with high performance rates in text reasoning.

2.2. ADVANCED PARADIGMS FOR LEVERAGING LANGUAGE MODELS

2.2.1. THE RETRIEVAL-AUGMENTED GENERATION (RAG) PARADIGM

In recent years, due to the increase in the use of pretrained large language models due to the required resources to train an LLM with task-specific data, a new challenge has arisen, as pretrained LLMs are limited by the data they were trained with. Although most training paradigms make use of huge amounts of multifaceted data for the models to be as adaptable as possible, when implemented to very specific tasks that require knowledge regarding the task-specific data, language models struggle to fully adapt them, often resulting in hallucinations where the model invents false data to make up for the unknown information. To tackle this issue of non-adaptive LLMs without requiring a pretraining step to take place, seen in proposals like Google's REALM [68] which are effective but costly, Retrieval-Augmented Generation (RAG) architectures were created.

Retrieval-Augmented Generation (RAG) architectures [69] use retrieval models to allow the language models to use additional references when carrying on certain tasks. This means that, instead of answering prompts considering only the knowledge obtained during training, language models can consider task-relevant information given through augmented prompting that combines the initial query with the additional data retrieved by the model. Since task-specific information is considered, language models become more specialized and controlled, answering queries with precise, updated, and factual information. Therefore, the base RAG concept was built considering advancements in various fields, including the studies on semantic search and data retrieval, with important previous

contributions like Sentence-BERT [70], which allowed for the computation of dense sentence semantic embeddings for similarity computations, and Dense Passage Retrieval (DPR) [71], which first introduced the relevance of using dense beddings instead of keywords for retrieval tasks, and studies on text generation and reasoning on text, with the introduction of language models with context-based input like the previously mentioned GPT and LLaMa families.

Naïve RAG is the label given by the community to the popularized and ‘standard’ RAG architecture, which has become the most used and widespread RAG approach, implemented nowadays in most of the retrieval-based language systems, with implementations ranging from healthcare [72] to tourism [73]. Naïve RAG architectures are based on a fixed structure idea, in which the process followed is optimized and always the same, with the use of pretrained elements, avoiding some of the limitations present in earlier renditions of the architecture, where both the retrieval model and the language model implemented are previously slightly pretrained for their respective tasks in the RAG workflow. This fixed architecture is the one usually associated with RAG systems due to its importance in the field, which is the one defined with an initial query that prompts a retrieval model to extract relevant pieces of data from an available vector database which are then given to an integration and augmentation component to define a new prompt, including the initial instructions and the retrieved context, given to a final pretrained language models. Naïve RAG architectures have two main implementation approaches, which are non-fine-tuned implementations, known as zero-shot RAG, and fine-tuned implementation.

Zero-shot RAG is widely more used than the alternative, as it implements the idea of using the models in the architecture as they come. The simple implementation of this approach combined with the high-performance results has resulted in many works preferring this alternative over more computationally costly ones. This is reflected in the wide variety of zero-shot RAG implementations across multiple study fields. In the field of healthcare, many of the RAG implementations are based on chatbots, where the answer to questions is given considering specific and updated medical documentation. Authors like Kemboi [74] present this idea in their work, where they develop a chatbot for patient use that has direct access to the needed medical documentation for all the questions to be answered accordingly. Other works, like the one presented by Nayinzira and Adda [72], follows the same chatbot idea but aimed at conversation rather than Q&A, as the work develops a chatbot aimed at helping people struggling with mental health by using the RAG architecture to carefully analyze and interpret the sentiments plastered on the messages. Some chatbots are also developed with the medical professional in mind, like the one presented by Raja and Yuvaraajan [75], where natural language processing techniques are combined with a RAG architecture with the goal of providing doctors and medical investigations with relevant and contextualized information regarding infectious diseases.

However, in other fields, RAG architectures are implemented for goals that often exceed chatbots. For example, under the works aimed at the field of transportation and urban planning, authors like Yang et al. [76] use the RAG architecture to integrate simulations with real data to generate an intelligent transport system that can simulate and analyze the transportation system in cities in real time, while others, like Banerjee, Satish, and Wörndl [73], use these architectures to enhance tourism recommender systems, considering the user preferences, the city popularity and layout, the demand of popular tourist attractions at different points in the year, and the environmental impact. In fields like education, both the chatbot and processing applications of RAG architectures are considered, with works aimed at generating chatbots to help high-school student by giving study orientation based on verified external information [77], and other works aimed at the search and discovery of useful academic resources based on required academic concepts [78]. In the field of security and crime investigation, which is of special interest in the context of this work, naïve RAG architectures are also starting to arise as a useful tool to be used by law enforcement for investigatory tasks. Authors like Fang, Meng, and Macdonald [79] combine RAG architectures with knowledge graphs to construct autoregressive reasoning chains, used to identify and integrate gathered evidence to answer complex questions regarding a crime investigation.

Fine-tuned RAG implementations, although far less frequent in the literature, are still relevant to mention. These RAG implementation, while still based on the core naïve RAG architecture, are widely categorized under a different label, known as Advanced RAG, which encompasses fine-tuned

RAG implementations as well as additional implementations that differ from the naïve architecture to obtain better results. The main goal of advanced RAG, in the case of fine-tuning solutions, is to optimize the components of the naïve architecture to obtain a higher performance. This implementation, due to its complexity in some cases, is not as often seen in the literature, but it still represents a significant landmark in the Retrieval-Augmented based systems, as many of the architectures developed under this scope have presented more adaptative and ‘intelligent’ solutions. Depending on the application, the part of the architecture that is fine-tuned differs, but certain fine-tuning approaches dominate the literature more than others.

Most approaches in advanced RAG are aimed at fine tuning the generator model, which is the language model that carries out most of the logic behind the operations. The LLM fine tuning has the goal of improving the decision-making and response generation capabilities of the architecture, by adapting the system to specific tasks. Works like [80] are aimed at innovating the naïve RAG architecture by using large-scale LLMs fine-tuned with the LoRA and QLoRA methodologies with vector databases, with a human-on-the-loop feedback approach to ensure the correct adaptation of the LLM to the desires of the user. Other works, like the one presented by Chen et al. [81], focus instead on small language models rather than large-scale ones, for goals like tackling the hallucination problem in LLMs by training the models to express uncertainty rather than making up false information, with the goal of improving the precision of the RAG architecture. Many works also focus on fine-tuning the LLMs to effectively carry out certain tasks in the RAG architecture workflow, like the one presented by Zhang et al. [82], which enhances the model’s ability to reason on and use the retrieved documents by not only fine-tuning the model to better carry out domain-specific tasks, but also directly cite and showcase the detected relevant segments from the given documents. Other language model fine-tuning approaches present more complex ideas, like the one presented by Sung, Lee, and Tsai [83], where self-fine-tuning is carried out on a large language model with domain-specific question-answer pairs, for instruction generation task carried out by the RAG architecture to improve, obtaining more precise and domain-specific answers.

Aside from the generator model, the retriever model is also often fine-tuned to obtain better results, in many cases being fine-tuned alongside the generator model. In this case however, the tuning is aimed at avoiding bottleneck scenarios due to the inability of the retriever to find the needed documents, and instead tuning the retriever model to be able to better understand the intentions of the domain-specific task and the content of the available documents. Works like [84] are aimed at fine-tuning a small embedding model to be able to better extract the context information, so the RAG architecture can obtain more precise results without requiring the fine-tuning of the generator model. This is done through the input of positive-negative example pairs given to the retriever. Other solutions, as previously mentioned, fine tune the retriever while also improving the generator model. An example of this approach is developed by Zhang et al. [85], where the retrieval model is first improved through contrastive learning using negative sampling strategies like homogeneous in-batch negative sampling (HIS) and auxiliary hard negative sampling (AHNS), and then the generator model is also tuned through retrieval-augmented fine-tuning, where the model is trained specifically to answer questions given a context input.

Nevertheless, although advanced RAG applications are strongly starting to emerge, becoming more noticeable in the literature in recent years, advanced RAG still deals with some resource limitations, as fine-tuning in many cases is computationally costly. In the context of this work, the goal is to implement a RAG architecture to consider domain-specific data that language models are often not trained with (e.g. data accessible to law enforcement) in the analysis process. Due to the widespread tools for naïve RAG implementation, and its developed and highly documented structure, implementations based on this approach are of special interest. However, many of the more recent retriever models usually explored by advanced RAG application are also considered, as some architectures under the advanced RAG label make up for the lack of fine tuning by using complex and highly trained models.

2.2.2. THE AUTONOMOUS AGENT PARADIGM POWERED BY LANGUAGE MODELS

In the general landscape of machine learning, the continuous emergence of new technologies and applications has led to the development of intelligent systems that can compute complex tasks by themselves by mimicking human reasoning. One of such systems, which in recent years has garnered attention due to their complex nature but useful implementations, are AI agents. AI agents [86], which were first formally implemented in the robot Shakey [87], are autonomous systems which carry out objectives by interacting with its environment, reasoning over the task, making decisions, and acting accordingly. They do so by implementing perception-reasoning-action loops, in which given an initial input, the system reasons over how to carry out the needed task and implements an action, receiving an output that is observed and reasoned upon to repeat the loop until the output is the response desired. Language models are a key component in these systems, as they essentially work as the ‘brain’, carrying out the reasoning, planning, and decision-making tasks for the system to properly execute the needed actions to obtain the final goal.

Looking at AI agents strictly from a language model point of view, these systems allow LLMs to become proactive, rather than reactive like they usually are, since they are aimed at automating complex tasks, interact with external sources, and solve complex tasks that LLMs by themselves struggle with by combining the language model capabilities with a set of additional tools in a fully integrated and working system. However, depending on what the task to compute is, the agentic systems tend to be defined slightly differently, with different ways of interacting with the external environment, different decision-making tactics, and different levels of ‘intelligence’. In the literature, more specifically in the work presented by Russell and Norvig [88], five main types of agents are defined based on these characteristics, but with time, some have become more relevant than others in the literature.

Learning agents are the most relevant agents nowadays, as they have some of the most complex and advanced architectures, carrying out tasks and learning from experience using memory and feedback loops to continuously improve in performance. Most of the complex models defined under the machine learning and deep learning scopes can be identified under this label, as often agents which start under a different label are developed and improved towards becoming a learning agent, which learns instead of sticking to an established set of rules. Therefore, learning agents have become one of the most developed and important agent architectures in the literature, with different goals, fields of study, and learning paradigms. However, within the learning agents’ landscape, some architectural applications stand out over others, with the most widespread learning paradigm being based on Reinforcement Learning (RL), with some works, like the one presented by Andre and Russell [89], being specifically aimed at developing a framework for reinforcement learning-based learning agent design and generation. Therefore, the application of this paradigm has become the main one in the literature. This is exemplified by works like [90], where agents that learn and adapt by themselves are implemented using random neural networks with a reinforcement approach with the goal of developing a simulator. The agents improve their performance as the simulation goes through adaptive learning made possible with the neural networks. Other works, like the one presented by Balaji, German, and Srinivasan [91], also follow the reinforcement learning trend in the learning agents’ landscape, but in this case through the development of a multi-agent reinforcement learning system where a series of traffic light agents work together to adjust their light based on their environment and the state of their neighboring agents. In the same area of application (urban planning and traffic control), and through a more complex implementation of the RL paradigm, are works like [92], which implement deep reinforcement learning structures for the agent’s learning, considering both reward signals and action policies in the learning process. Nevertheless, some works with renowned results do not implement the RL paradigm for learning agents. An example in one of the mentioned previous applications with RL, simulation, is the work developed by Park et al. [93], where generative learning agents are presented as tools for human simulation in open virtual environments, by using episodic memory that stores all the events occurring with relevance and temporal decay values associated, for the GPT brain to be able to generate useful simulations considering temporal

coherence and realistic social interactions without the need of the reward signals used in RL approaches. Thus, due to the widespread study of these types of agents, while some paradigms have become more popular, the landscape of implementations keeps on growing, introducing new and different approaches to the task of self-learning.

However, even though learning agents are the most widely used due to their adaptability and self-learning capabilities, other types of agents have also demonstrated interesting approaches with similar high-performance rates, many being used often as a component integrated into a larger application, rather than stand-alone system. The most relevant type of agent which does not fall under the self-learning capabilities of learning agents but has gained recognition in many implementations is goal-based agents [94]. Goal-based agents are agents with a defined goal which generate dynamic plans to obtain them. They are slightly simpler than learning agents, as they do not learn based on their mistakes, but they do repeat processes and recalculate plans dynamically until the goal is achieved. Thus, goal-based agents still present a simpler but equally as interesting approach, with many applications of interest across multiple fields. Goal-based agents can be seen in implementations like predictions, with example works like [95], where the trajectories of autonomous vehicles are predicted through the consideration of the local goals and behaviors of the vehicle extracted from historic. The work uses two agents, each defined with different objectives, to carry out the steps needed to fulfill the final prediction. They can also be seen in different implementations, like behavior simulation in the case of [96], where realistic agents with associated personalities are developed by defining the personalities as sets of personalized goals and preferences. Each agent makes the necessary decisions to align with said sets, showcasing a ‘personality’ in doing so. Goal-based agents are also present in works which see the goals in these agents’ paradigms from a different perspective, like the one presented by Tung, Zelek, and Clausi [97], where the agents are seen as intelligent surveillance systems, and the goals are used to determine and detect unusual trajectory-based behaviors based on how coherent and aligned they are with the established goals. Thus, goal-based agents are very adaptable systems, which can be implemented for a multitude of different objectives. In the specific context in which this work is developed, although most implementations are aimed at modelling and predicting crime or criminal behaviors, the easily adaptable nature of these systems make them interesting for crime data extraction and crime contextualization tasks.

Although more agent types exist, like utility-based agents, simple reflex agents, and model-based agents, they are less popular and often aimed at tasks usually found in problems based on uncertain environments and conflicting objectives. Therefore, the implementation of agents in this work is mainly focused around the ideas of learning agents implemented through the structure of goal-based agents, as these agents are, as seen, often implemented for tasks that align better with the ones developed in this work.

2.3. VALIDATION AND EVALUATION APPROACHES FOR LANGUAGE-BASED SYSTEMS

One of the main challenges of language-based systems, including the mentioned large language models stand-alone implementations, the RAG architectures, and the AI agents, is the validation of the performance. Since language-based systems are often aimed at mimicking human reasoning and mannerism (when writing, thinking, etc.), the most standardized evaluation approach for this systems is often human evaluation, as more often than not, validating the ‘humanness’ of a result requires either a methodology that also perfectly mimics humans, or a set of parameters and characteristics that help identify human patterns on synthetic data, which is as hard of a task due to the volatile and heterogenous nature of human conscience. However, human evaluation, although mostly trustworthy in its validation results, is a limited resource, as humans in the loop often come with inefficiency, especially in complex tasks like manual validation.

Therefore, many of the studies surrounding language-based systems in the literature are not as focused on the systems themselves, but rather how those systems can be validated. For the most popular systems, including the ones tackled in this work, there are existing validation solutions that, although not formally standardized, do have slight recognition in the mainstream. Nevertheless, most validation

approaches are still in research stages, often working as experimental prototypes rather than established methodologies. However, even with the lack of standardization, the existing popular validation approaches are interesting as a base system for further ideas and developments in the field, as they establish non-official guidelines of what should be considered when developing validation frameworks. These existing approaches, additionally, vary slightly depending on the system being validated, showcasing the differences in characteristics and objectives among these systems.

In the case of stand-alone large language model implementations, validation is often focused on the precision, coherence, generalization, and alignment with human reasoning of the responses given to the prompts by the model. Thus, the existing validation approaches are often focused on one or more of these characteristics, and due to the difference in nature of each of them, the approaches also vary greatly between them. One of the main approaches for LLM validation is the evaluation of the performance through a series of benchmarks. The most widely known validation frameworks are under this category, including frameworks like GLUE [98] and SuperGLUE [99], which both aim at measuring the overall general understanding of the language through several natural language processing tasks by using an aggregated average score, MMLU (Massive Multitask Language Understanding) [100], which aims at evaluating general academic and professional knowledge through 57 defined tasks by using an accuracy score, HELM (Holistic Evaluation of Language Models) [101], which is aimed at evaluating LLMs on complex metrics like precision, calibration, robustness, bias, fairness and toxicity, and BIG-bench (Beyond the Imitation Game-benchmark) [102], which is aimed at evaluating the limiting capacities of LLMs through a series of extremely difficult tasks ranging from complex logic reasoning to joke understanding.

Aside from the more complex and structured frameworks, some general validation metrics, which are often used in the development of novel validation frameworks, are sometimes also used. This usually includes similarity metrics either with reference answers, like in the metrics BLEU [103], ROUGE [104] and METEOR [105], or with embeddings, like the metric BERTScore [106]. However, as mentioned, many works in the literature are aimed at developing new validation frameworks, either using these known metrics or by designing and developing new approaches. Works like the one developed by Reddy [107] present new validation frameworks that do not use any of the known approaches, but rather validate the output of LLMs using new implementations, in this case validating accuracy, coherence and factuality through the integrated implementation of fact-checking automated systems (cross-reference approach), semantic coherence texts, and source-based authentication methods. The VALID framework, introduced by Estevez et al. [108] also aims at validating the responses given by LLMs through novel approaches, which include the use of variable-level performance metrics (benchmarking against human abstraction approach), automated verification checks for inconsistencies in the data, and replication and benchmark-based analysis. These approaches are used to extract numeric metrics (accuracy, consistency, completeness, etc.) aimed at validating the performance of LLMs in the task of accurate and reliable data extraction. Other validation frameworks even consider possible additional limitations to validate, like the ReLM framework [109], which validates behaviors in LLMs like memorization (memorized training data), gender bias, toxicity, and language understanding through standardized regular queries formulation.

For the simulation of more complex LLM-based systems, like RAG architectures, the approaches are also complex, as more than one model must be validated to understand the result of the overall architecture, including the retrieval model and the generation model. Complete and popular validation tools have been developed for the simplification of RAG validation, like the RAGAS (RAG Assessment) tool [110], which automate the calculation of a series of usually implemented metrics in RAG validation. However, these tools introduce an idea widely seen in the literature as one of the main validation approaches, which is the ‘LLM as judges’ approach, in which LLMs are validated and used to validate. These approaches yield good and effective methodologies, but can also become complicated if the LLMs are not properly integrated into a framework in which the limitations of the validator model do not affect the validation process. Nevertheless, aside from ‘LLM as judges’ solutions, the RAG validation landscape also has a series of somewhat known metrics to validate both the retriever and the generator. For the retriever, usual metrics include Precision@K [111], to validate the proportion of relevant documents within the K documents retrieved, Recall@K [111], to validate

the proportion of actual relevant documents that are retrieved, and MRR (Mean Reciprocal Rank) [112], to validate the efficiency of quick and correct retrieval through the evaluation of the ranked position of the first retrieved relevant document. For the generator, the approaches used tend to be like the ones used for stand-alone LLM implementations, usually through the mentioned relevant metrics.

Some works base their developed frameworks on the base ideas introduced by the mentioned metrics and validation frameworks. An example of this is the AutoNuggetizer framework presented by Pradeep et al. [113], which is directly based on the evaluation methodology of ‘nuggets’ (atomic facts) first introduced in the Text REtrieval Conference (TREC) in 2003 [114]. This evaluation methodology is based on the evaluation of specific facts included within the generated responses using LLM-based approaches for nugget generation and assignation. However, as with the validation of LLMs, many works in the RAG validation literature are aimed at developing new validation frameworks and metrics. Works like [115] present novel frameworks for validation, which combine a series of validation strategies, like the proper selection of metrics and baselines, the qualitative analysis of architectural errors, and the application of systematic refinements to improve the architecture. This is done with the goal of validating the RAG architecture’s configuration dependencies and limitations to improve the precision through the mentioned adjustments to the system. Other works, like the one presented by Friel, Belyi, and Sanyal [116], introduce both a validation framework as well as validation benchmarks. The framework, TRACe, implements explainable metrics aimed at validating the utilization of relevant documents by the language model, the relevance of the documents retrieved, the alignment of the LLM response to the context data given, and the completeness of the response regarding the query given. This is carried out with a generated benchmark database, RAGBench, which stems over various domains for the proper validation of the architecture.

Finally, out of all the validation approaches, the ones regarding agentic systems are the ones that showcase the most challenges due to the novel nature of the agentic AI field of study, which lacks as many known validation approaches as other fields, and due to the complex nature of the AI agents themselves, which are constructed on sequential decision-making-based systems. Thus, the validation of these systems is based on the evaluation of complex environments which are highly dependent on the type of agentic system validated and its implementation. Rather than complete validation frameworks, agentic AI validation is often based on benchmarks and abstract metrics, some of which are slightly known, but none that have established a standardized methodology of evaluation. Some of the more known benchmarks include AgentBench [117], aimed at validating the agent’s ability to interact with different environments, WebArena [118], to simulate realistic web environments that require complex tasks, and game-based benchmarks, like MineRL [119] or StarCraft II Learning Environment [120]. However, as for the metrics, no defined metrics are usually seen in the literature, but rather, generic rates and trail verifications to somewhat validate the system. This includes rates like task success rates (based on the percentage of correct task completions), and verifications like average steps of computation, average cost of tasks, and robustness of the system against changes in the instructions and environment. Thus, since base frameworks and metrics are not fully defined for this validation approach, the works within this landscape are often more aimed at the evaluation of validation limitations and challenges, like [121] and [122], rather than in new approaches of validation of these systems.

In this work, all the language-based systems implemented will have to be validated considering the domain-specific requirements from each one of them. To do so, as many of the works in the literature, a novel validation framework approach will be developed, slightly based in some of the ideas defined by known metrics and frameworks, as well as other novel approaches analyzed.

3. DESIGN OF A UNIFIED FRAMEWORK FOR DATA CONTEXTUALIZATION AND PREDICTION BASED ON A BEHAVIOR ANALYSIS FRAMEWORK

As briefly mentioned in the introduction, the intersection between georeferenced data exploitation systems and language-based systems is of great interest since both are areas of study which are currently living a significant growth. In the specific field of law enforcement and criminal justice, these types of systems could help tackle many of the existing limitations in the processes of analyzing and interpreting data gathered. The aim of this Master's Project is to design and develop a framework which combines a system based on analyzing movement data through the extraction of behavioral patterns and the detection of anomalies with additional processing modules which enhance the conclusions gathered by properly contextualizing and interpreting them, for both decision-making and future movement prediction in the context of a police investigation.

More specifically, the proposed framework is aimed at enhancing an existing system developed under the same line of investigation for the Bachelor's thesis. This system is aimed at capturing relevant characteristics from georeferenced data. These characteristics include behavioral patterns, which give an insight on how the individual moves and what his/her conduct is while doing so, useful in prediction and crime analysis tasks, and anomalies in conduct, which help detect deviations in behavior that might indicate possible unusual situations or circumstances to consider when making decisions. This system, although useful as a standalone tool, has a main limitation related to the interpretability and transparency of the results. While the behavioral model and anomalies detected are easily understood by processing modules within the system, they are not as readable to a human user. Furthermore, the system only considers the input georeferenced data, without contemplating outside context, not only related to the investigation itself, but also to the locations visited by the individual and the significance of them.

The framework designed in this work envelopes this system, firstly improving some of the existing processes within it, and then creating a complex workflow through the organized integration of a series of processing modules that carry out the necessary task for the outcome given to the user to be useful, complete, context-aware, and understandable. Two main sub objectives are considered to obtain this goal. Firstly, given that this framework is developed in the context of its use in law enforcement, the desire is for the final structure to consider all the possible data that would be considered by the police in their current data interpretation processes. This means that the main georeferenced data considered should be correlated and contextualized with both existing police reports which might make a reference to related crimes as well as external known context which might make certain behaviors, anomalies, or locations relevant in an investigation (e.g. behaviors associated with certain crimes, locations known to be crime hotspots, etc.). Secondly, raw location data like coordinates is often non-interpretable by humans, as it includes many data instances with no explicit rationale regarding what the data means in a broader context and why it might be relevant. Thus, behavioral models built based on this data and anomalies detected considering this data also might not be directly interpretable, since they also lack an explicit explanation and justification on their relevance. The second desire is for this framework to properly interpret all the results given and convert them into an easily understandable form for humans, for the user to be able to draw his/her own conclusions.

In this section, we will go in detail over the problem statement, including a slight mention of the problem to tackle with the existing system, as well as the newly developed modules, and we will give an overview of the framework architecture proposed to cover the previously mentioned subobjectives.

3.1.PROBLEM STATEMENT

Location has always been a relevant factor considered in investigations and police work. With the recent technological advances made in the field, georeferenced data has become easier to leverage as more systems arise which efficiently tackle the tasks of properly analyzing this kind of data. Movement data gathered through GPS-enabled tracking devices or mobile phone logs include useful insights that could help determine the modus operandi of a suspect, the relationships among suspects, and even confidently predict what a suspect or group of suspects might do next, allowing law enforcement to not only better understand how suspects behave, but also to avoid future crimes. However, as previously mentioned, this kind of data, although useful, carries a big limitation, which is the lack of meaning and interpretability. For location and movement data to be fully exploited, this obstacle must be addressed. Two main problems contribute to the existing limitations surrounding the efficient exploitation of movement and location-based data.

The first of the two problems is the lack of value and meaning in raw location data. Georeferenced data, especially in the case of police work, carries meaning when it is given to it by considering the context surrounding it. This context includes two main things:

- The underlying patterns that model the individual's behavior and allow to detect deviations from it. Location data is useful if these underlying patterns can be properly defined and used for analysis and prediction.
- The investigation which prompted the extraction of the data in the first place. The data by itself only showcases the movement of an individual, without considering the context surrounding the individual and the places he/she visits. Knowing the significance of the behaviors, anomalies and locations visited by the individual in the investigation and in a criminal analysis context is what gives them meaning.

However, the extraction of both the underlying patterns, as well as the integration between multiple data sources for contextualization purposes is not an easy task, especially through manual analysis, which is often infeasible. Therefore, these limitations create the need for analysis tools that can give meaning to the location data through pattern extraction and data contextualization. More specifically, the framework proposed must be able to firstly extract the needed patterns from the georeferenced data, to give it meaning by itself, and then integrate, connect, and contextualize it with diverse sources, to give it meaning in the context of the investigation. To do so, first a series of behavioral patterns of interest and anomalies (deviations from behavior) must be defined for the problem to be bounded to the useful patterns. Then, data should be properly analyzed for the detection of these patterns and the future detection of anomalous conducts on new data. Furthermore, the framework should consider all the data sources which would be available to the police force and are of interest to integrate with the georeferenced data and to contextualize it. These sources should be used to give meaning to the locations included in the data, the behavioral patterns extracted and the anomalies detected. Finally, the contextual significance given to the data should be enough for the police force to make both informed decision-making as well as predictions of possible future movements and behaviors.

The second of the two mentioned problems is the lack of interpretability of location data. Even with attached context, georeferenced data is often non easily understandable by humans, in some cases even non-readable. Although data like coordinates or locations of interest are crucial in a police investigation, they are not easily interpretable without an attached rationale and explanation which enhances that interpretability. This makes transparency and reliability more difficult, which diminishes the usefulness of the system and decreases the likelihood of it being used. Thus, this limitation creates a need for the data to be presented in a coherent way that is easily understood by humans and allows the user to make his/her own interpretations regarding the data.

The proposed framework must also tackle this second mentioned necessity by implementing tools that allow for comprehension and reasoning over text and data, to fully understand what the context surrounding the data is saying about it and properly extract its significance. Moreover, the framework must make use of tools which are able to adapt the information given and generate new passages of

text which are coherent, relevant, and connect the information to derive useful conclusions. This allows for the conclusions to be given to the user in an easily understandable format (like text).

The framework proposed in this work is aimed at tackling all the explained necessities through the integration of processing modules that work interconnectedly towards the main goal of, given a series of data sources in which the georeferenced data is the main point of interest, being able to automatically extract the relevant information from the data, properly contextualize and integrate it with the known data, and present it to the user in a coherent way for he/she to be able to draw conclusions and take the necessary actions.

3.2.FRAMEWORK OVERVIEW

To properly consider and tackle all the mentioned necessities, a complex framework is designed and implemented. This framework combines an existing system with a series of additional modules which allow for the input data to be transformed into the structured, context-aware, and coherent output. Four main components make up this framework, as can be seen in Figure 1. These components can be defined as the behavior analysis and anomaly detection system, the police report simulation module, the data contextualization and interpretation module, and the movement prediction module.

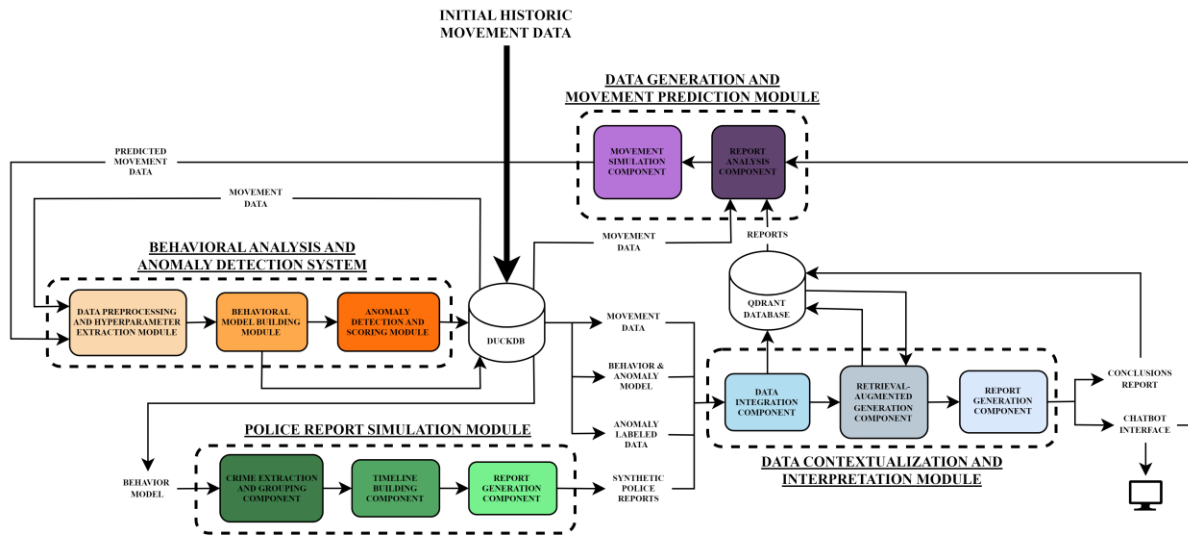


Figure 1. Proposed framework architecture

The *behavior analysis and anomaly detection system*, as mentioned, was designed and validated previously in the same line of work for the Bachelor's thesis. In its integration with this framework, the system is slightly modified to better adapt to different data formats and to be better integrated with the remaining modules in the framework. However, the main goal of this system is still the same, which is extracting relevant behavioral patterns from georeferenced data to build a behavior model and later detect anomalous behaviors in new data based on this model. Thus, this system takes as input a series of georeferenced data points (coordinates and timestamps) regarding the movements made by a certain followed individual. The system implements three main modules, which sequentially preprocess and reformat the data, and extracts the behavioral patterns, building a behavioral model and saving it into a database. Then, when new data regarding the same individual comes along, a similar process is carried out to extract the behavioral patterns that could contain anomalies of interest. Then, the original behavior model built with historic movement data is compared to the new data gathered and anomalies are detected. These anomalies are labeled on the new data with both the type of anomaly and the degree of the deviation from the established expected behavior. This labeled data is also stored in a database for future contextualization.

The *police report simulation module* is developed with the goal of generating synthetic police reports that can be used by the system to further contextualize the location data, as it is expected for these reports to be available in an actual implementation of this system by the police force. To generate these reports, the actual locations visited by the tracked individual are considered, for the reports to be both relevant to the actual data and useful in the contextualization, and for them to be as realistic as

possible, considering real crimes that might be mentioned in other sources used. Thus, this module is made up of three main components. The first crime extraction and grouping component takes as input the behavioral model built around the individual and is aimed at extracting the relevant locations and timelines and contextualizing them with real crimes that have occurred there through open database searches. The crimes are then clustered and categorized into different types. The second timeline building component takes these clusters as input and, based on the crime category desired to be considered for the reports, generates a realistic timeline of events that could have occurred in the locations and times given by a single individual (a realistic crime timeline). To do so, both the actual locations visited by the individual, as well as the real patterns associated with certain crimes, are considered. Finally, the report simulation component takes this timeline and generates a series of structured and realistic reports. These reports are then stored in a database by future use.

The *data contextualization and interpretation module* is aimed at properly integrating all the data sources available to contextualize and interpret the location data, and to then present the conclusions reached to the user in a coherent and understandable way. To do so, a three-component structure is designed. Firstly, the data integration component takes into consideration all the sources available, which include the raw movement data, the behavioral model and the anomalies detected in the first system, the reports generated in the previous simulation component, and the sources available online, which give further general knowledge. These sources are scoured to extract the relevant information to correlate them with the location data and contextualize them with additional information. The final context with its data is then transformed into numeric representation for future retrieval. The retrieval takes place in the second retrieval-augmented generation component, which, based on a prompt given by the user (specifying which data they wish to analyze/contextualize), retrieves all the relevant stored data, processes it, and generates a new prompt for a final language model. The prompt must include all the available data, and the instructions of the desired outcome, which is a structured response that explains all the conclusions reached by combining the data. This outcome is generated in the last report generation component, which generates a complete report including the findings to date, the connections to other data, and the possible relevance to the investigation. In this last component, a chatbot interface is also made available to the user to ask direct questions regarding the data in case the report has left something out or the user has pending unanswered matters.

Finally, the *data generation and movement prediction module* is aimed at making predictions on what the individual might do next based on the current conclusions reached and summarized in the reports. The module takes as input the reports available on an individual and simulates future possible movements, for the police force to have an approximated idea of where the suspect might go next and what he/she might do. To carry this out, this module is composed of two components: the report analysis component and the movement simulation component. Firstly, the report analysis component extracts the needed information to define the parameters needed for the second component. This information is directly extracted from the previous data contextualization and interpretation module, either via the chatbot interface, from which the value of user-dependent parameters is defined, or via the generated reports, from which the value of suspect-dependent parameters is defined. Once the information is extracted, a reformatting and structuring step is carried out to ensure that the input given to the next component is structured as desired. The second movement simulation component, which is based on a simulator previously developed on the same line of work, takes the structured and reformatted parameters as input and generates movement data based on them. This generated data will follow the expected behavioral patterns to create an approximation (a prediction) to what the suspect might do in the future. The generated data is directly given back to the framework, starting with the behavior analysis and anomaly detection system, to be processed, so law enforcement can properly interpret it.

Due to the nature of these processing modules, all of which follow LLM-based approaches but target different objectives, a careful evaluation of the existing tools and models regarding language models and language-model-based paradigms has been carried out to determine the proper resources to carry out the needed tasks by each of the modules. This evaluation was performed based on the requirements by each of the modules, mainly regarding the desired outputs based on the expected inputs, and the tools needed in each case to reach one from the other. A comprehensive and modular

software pipeline exposing the I/O of each implemented task defined for this evaluation can be seen in Figure 2.

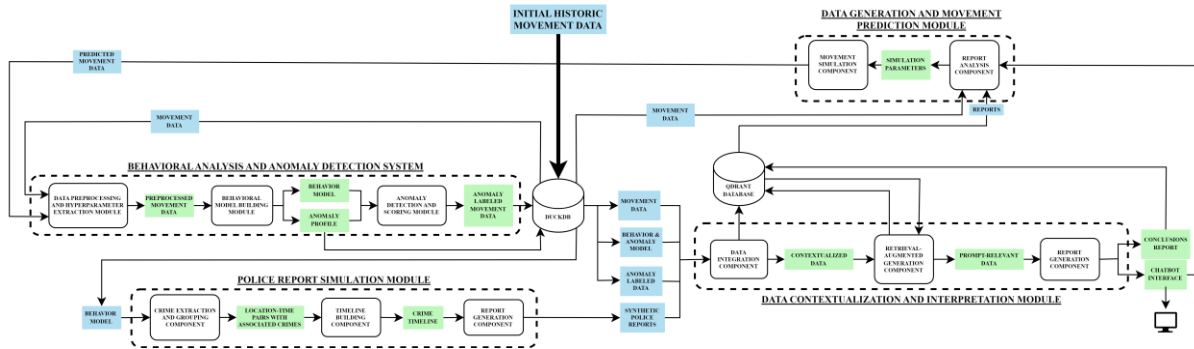


Figure 2. Overview of software pipeline with I/O of each implemented task

Based on these I/O requirements, as mentioned, a series of tools and models developed in the LLM landscape were assessed and compared to determine the most suitable ones for this project's goals. The general selection guidelines were based on three criteria. The first one was the model or tool's self-hosting capabilities, with only open-source and locally deployable solutions considered due to the nature of the project and the framework future implementation goals (fully self-deployable in law enforcement systems). The second one was the model or tool's performance in specific needed tasks. The framework incorporates similar tools with shared overall objectives (e.g. different LLMs) due to the task-specific nature of many of the tools and models, where even with the same base, some are optimized for specific tasks to achieve better performance. The third and final one was the model or tool's ability to easily integrate into a broader system, due to the complex architecture of each of the modules and the framework that they constitute. An overview of all the selected tools and models, and their implementation in the framework can be summarized as follows:

- **Large Language Models (LLMs)**
 - *Mistral 7B model* for text generation tasks in the police report simulation module.
 - *Qwen2.5-14B model* as the language model in the AI agentic system for the data contextualization tasks in the data contextualization and interpretation module.
 - *Llama3.2 model* as the language model in RAG architecture for report generation tasks in the data contextualization and interpretation module.
 - *Mistral 7B Instruct model* for report processing tasks in the data generation and movement prediction module.
- **Text embedding models**
 - *all-MiniLM-L6-v2 model* for text comparison tasks in the police report generation module, as embedding model in que RAG architecture for the data integration task in the data contextualization and interpretation module, and for topic detection task in the content-based evaluation framework.
- **Databases**
 - *DuckDB database* to store the data inputted and outputted into the behavior analysis and anomaly detection system.
 - *QDrant vector database* to store the integrated data to act as the knowledge base for the RAG architecture for the data contextualization and interpretation module.
 - *ChromaDB database* to store the conversation historic for the chatbot interface task in the data contextualization and interpretation module.
- **Engines and processing motors**
 - *ChatOLLama engine* as chat engine in AI agentic system for the data contextualization task in the data contextualization and interpretation module.
 - *SearXNG metasearch engine* for the source scouring task in the AI agentic system for the data contextualization task in the data contextualization and interpretation module.
 - *OLLama engine* for knowledge integration in the RAG architecture in the data contextualization and interpretation module.

- *ChatPromptTemplate engine* for prompt engineering in the RAG architecture in the data contextualization and interpretation module.
- *Open-Source Routing Machine (OSRM) engine* for the data generation task in the simulator implemented in the data generation and movement prediction module.
- **Frameworks**
 - *LangGraph* for the definition of the AI agentic system for the data contextualization task in the data contextualization and interpretation module.
- **Data processing algorithms**
 - *HDBSCAN algorithm* for crime grouping task in the police report simulation module.
- **Natural Processing Inference (NLI) models**
 - *cross-encoder/nli-deberta-v3-base NLI model* for the statement comparison task in the fact-based evaluation framework.

4. A RETROSPECTIVE REVIEW OF A SYSTEM FOR BEHAVIORAL ANALYSIS AND ANOMALY DETECTION

The framework developed in this project is built over a previously defined and implemented system as part of the work presented for the Bachelor's thesis. This system is aimed at processing movement data defined through location tracks (coordinates and timestamps) to build a comprehensive behavioral model, to explain the patterns of movement included in the data, and to detect anomalies on the data, to give insight to relevant deviations from the detected behavior. Thus, this system encompassed a series of technical processes that implement statistical and probabilistic approaches to carry out the complex tasks, considering solely the data inputted to the system. It was built around three main modules, as seen in Figure 3, which include a data preprocessing and hyperparameter extraction module, a behavioral model building module, and an anomaly detection and scoring module.

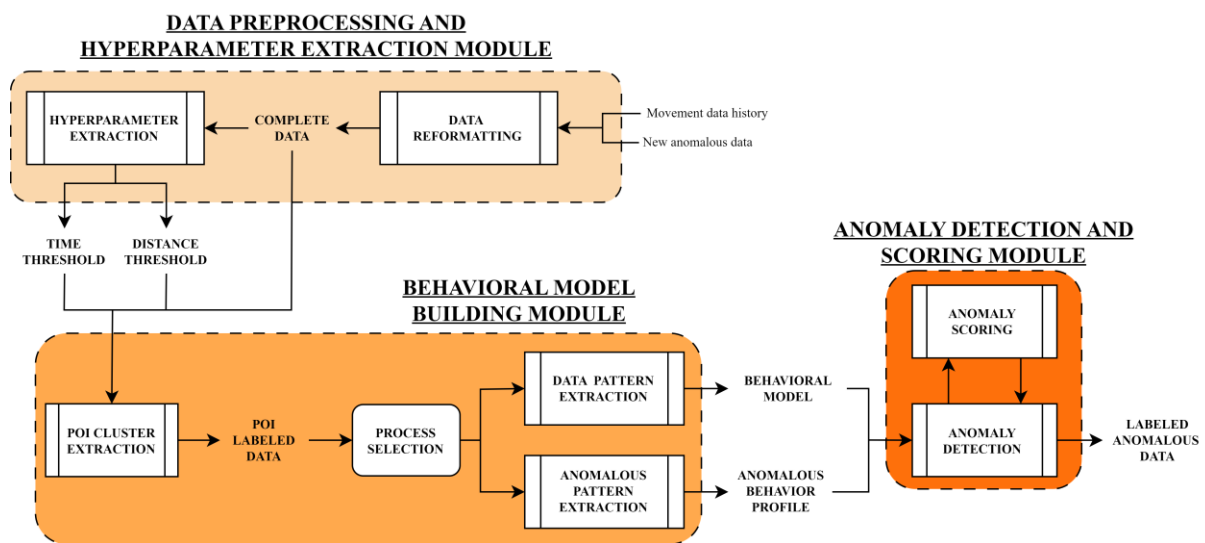


Figure 3. Behavioral modelling and anomaly detection system architecture

The modules in the system work sequentially, obtaining new results based on the conclusions reached from previous processes. The system starts by preprocessing the raw movement data which, as previously mentioned, includes information regarding the coordinates and timestamps followed by a certain individual. These movement tracks are pre-processed to both format them correctly for future processes, as well as to extract the necessary hyperparameters for the algorithms implemented in the next behavior model building module. This second module takes the formatted data and the hyperparameters and creates a statistical model around all the behaviors defined as relevant for the use case. The behaviors are identified on top of a movement model built based on the places of interest (POI) identified for the individual. Once this model is defined, it is possible to detect anomalies on new data instances based on what has been defined as expected. The last anomaly detection and scoring module takes the model built and new data instances and detects anomalies on the new data, associating a certain degree of deviation to each one of them. Thus, at the end of the system's workflow, given a series of raw location tracks, a behavioral model is built, which showcases the individual's usual conduct and preferences when moving, and a series of anomalous instances are detected on new data, which showcase instances where the individual deviates from usual conduct with a certain degree of deviation.

All the modules in this system have an output connected to a database, for future processing modules in the general architecture to correctly access the data. This database thus stores information regarding

the raw movement data, the behavioral models built, the anomalous behavioral profiles defined, and the anomalies detected. The specifications regarding the database can be seen in Table 1.

Table 1. External tools used to store the results of the behavior analysis and anomaly detection system

TOOLS	DESCRIPTION	RELEVANCE	USE
DuckDB	Column-based database optimized for quick retrieval tasks in data analysis processes.	- Free of use, can be deployed easily locally, and its implementation does not require complex infrastructures - Allows for a quick access to the data due to in-memory storage	To store raw movement data and data outputs from behavior analysis and anomaly detection system.

This section will include a revision of the functionality of this system and each of the modules that compose it, including needed inputs and expected outputs. Furthermore, the limitations on the existing system will be identified and refined for its correct integration with the newly developed modules to define the final framework proposed in this work.

4.1. DATA PREPROCESSING AND HYPERPARAMETER EXTRACTION MODULE

The data given to this system is expected to be raw movement tracks gathered from tracking devices, often only including information regarding the location a specific individual was at (through coordinates) and the time the individual was at that location (through timestamps). For behavioral information to be properly extracted from this data, two main tasks must first take place. Firstly, the data must be correctly cleaned and formatted, to ensure that all the needed information is included to define movement and trajectories from the data, and to ensure that the information has the format expected by the algorithms implemented to build the behavior model. Secondly, the necessary hyperparameters used in the algorithms implemented to build the behavior model must be extracted. To carry out both tasks, this first data preprocessing and hyperparameter extraction module is defined.

This module is made up of two steps: the data reformatting step and the hyperparameters extraction step. Although one step does not directly depend on the other, to avoid possible errors due to incomplete data instances, the steps are computed sequentially, first cleaning and correctly formatting the data, and then computing the necessary hyperparameters using the newly cleaned data. The structure of this first module, containing both steps, can be seen in more detail in Figure 4.

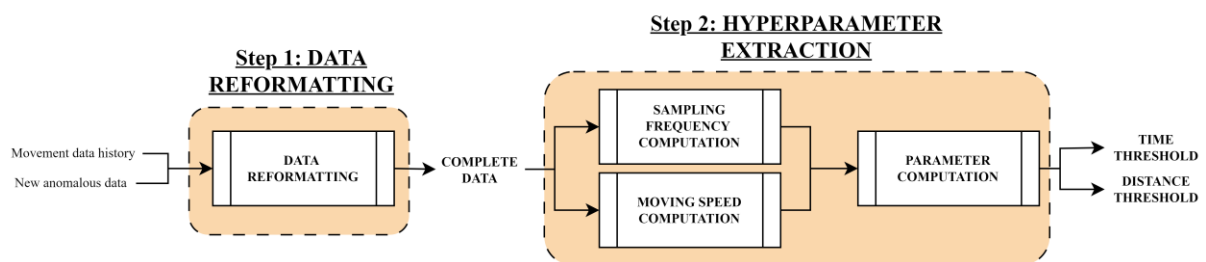


Figure 4. Structure of the data preprocessing and hyperparameter extraction module within the behavioral modelling and anomaly detection system

4.1.1. DATA REFORMATTING STEP

The behavioral model building algorithms implemented require each of the data instances to contain a defined set of information to properly extract meaningful behavioral patterns and build the model. Therefore, before that process is carried out, we must ensure that the data given to the model building

module is complete and contains the information required. This first reformatting step is aimed at preprocessing the raw data and structuring it in the format expected by upcoming processes.

The raw data instances are expected to be composed of a coordinate defined with latitude and longitude, to express location, and a timestamp containing the time and date, to express time. However, to properly implement future algorithms, the data instances must contain five fields which include a unique identifier which relates the data instance to the individual that generated it, a timestamp containing the date and time information in the format required, a numeric value indicating the latitude position, a numeric value indicating the longitude position, and a numeric index indicating the instance's position in relation to the whole set.

Starting with the unique identifier, this identifier must be equal to the one used to refer to the individual across documents and file names, to avoid possible future errors due to discrepancies in the names when extracting or storing data in the databases. The identifier does not have restrictions in content, like requiring only numeric or alphabetic characters, as long as it uniquely identifies every individual across the whole database. Thus, a track of all the defined identifiers must be kept to ensure consistency in their definition and no repetitions.

Both the timestamp and the respective latitude and longitude numeric values can be easily defined, since the base information is directly gathered from the content of the raw data. The timestamp which defines the date and time a data instance occurred has to simply be reformatted into a 'day.month.year hour.minute.seconds' format, where the date is defined considering that the day can take values from '01' to '31', the month from '01' to '12', and the year any four digit number, and the time is defined considering that the hour can take values from '00' to '24', and the minutes and seconds can take values from '00' to '59'. Considering the limitations, a datetime object is generated which correctly defines each structure. Furthermore, as for the latitude and longitude values, these are expected to be defined in the raw data through string values formed by a positive numeric value and a hemisphere indicator (N, S, E, W). To be able to properly work with the coordinate values in future algorithms, these strings are transformed into completely numeric values, using the hemisphere indicator to establish whether the number stays positive (when N in latitude and E in longitude) or becomes negative (when S in latitude and W in longitude).

Finally, a numeric index is attributed to each data instance regarding their position in the raw data file. Therefore, at the end of this step we have a new set of data, containing the same number of instances but each with a given index, well formatted time and location attributes, and an identifier equal to all the instances generated by the same individual. This new formatted data will work as input both for the next hyperparameter extraction task, as well as the next behavioral model building module.

4.1.2. HYPERPARAMETER EXTRACTION STEP

As previously mentioned, the behavioral model built in the next module is based on the extraction of points of interest used to structure movement. These points of interest, as will be further explained in the next section (Section 4.2), can be understood in the data as dense clusters of points formed by a high number of instances which are close together in space and time. The clustering algorithm used to define these clusters requires the specification of two parameters: a time threshold and a distance threshold. These parameters are later used to define the distance and time limits at which a new point is no longer considered part of a cluster and the cluster is therefore marked as completed. Thus, depending on the value of these parameters, different points of interest will be defined on the data.

To automate the process of selecting the parameters that define the best possible points of interest in the data (the closest definition to the correct values), a trial-and-error optimization approach is carried out. The main goal is to define a series of equations that make use of the information available to compute an approximation to a value for the two needed parameters. Thus, to define these equations, we first define what information should be considered when establishing what the structure of the places of interest to be detected looks like. This information includes the following:

- **Sampling frequency:** Depending on how often the points are sampled in the data, the clusters that represent the POIs on the data will be sparser or denser (points will be closer or more

separated). Furthermore, depending on the usual time distance between points, the threshold of time to consider that a cluster should be defined might differ.

- Moving speed: Depending on how fast the tracked individual moves, the clusters that represent the POIs on the data might also be sparser or denser, as a fast-moving individual, even with a high sampling speed, might generate sparser points (there is more distance travelled between the samples), and vice versa with a slow-moving individual, which might generate denser points.

Thus, the parameters must be adapted to both the sampling frequency of the data and the average moving speed of the individual which generated said data. As previously mentioned, a trial-and-error approach was implemented to define the equations which take both values as input and generate the needed distance and time thresholds.

Starting with the distance threshold τ_d , this threshold is needed to determine how far a point must be from a cluster's core to consider that it is not part of the cluster anymore. Thus, we make use of the average distance avg_d between points to determine which distances are 'normal' between points. Multiplying said average distance by a certain parameter N , we establish the distance threshold as a factor of the average distance between points, where a point being at a distance superior to the average scaled by N from the cluster's centroid is considered 'far enough' to not be part of it. This is computed using Eq. 1, where the parameter N is determined to be 4 by the trial-and-error optimization process, and the average distance is computed with both the moving speed avg_v and the sampling frequency f_s .

$$\tau_d = N * avg_d = N * (avg_v * f_s) = 4 * (avg_v * f_s) \quad (1)$$

As for the time threshold τ_t , this threshold is used to establish the minimum amount of time that must pass within a cluster (between the first and last points in the cluster) to consider it a point of interest. Thus, the average time difference between points is used, to establish, once again, what is the 'normal' difference, and what value above said difference could work as a threshold to determine that the individual has made a significant stop (has stayed at a point of interest). In this case, it has been determined through trial and error that this threshold is best computed through a linear function with a certain slope m and an intercept value b . As seen in Eq. 2, the sampling frequency f_s is used to establish the average time difference between points, and the slope m and intercept value b are determined to be 1.1 and 117 respectively through the mentioned optimization approach.

$$\tau_t = m * f_s + b = 1.1 * f_s + 117 \quad (2)$$

Given the extracted threshold functions, the hyperparameter extraction task computes the needed parameters through two simple steps. Firstly, the information required by the defined equations (sampling frequency and moving speed) are computed using the data available. For the sampling frequency f_s (as seen in Eq. 3), only the average time difference between pairs of points (avg_{t_diff}) and the total amount of points $total_p$ are needed. As for the moving speed avg_v (seen in Eq. 4), both the average time difference avg_{t_diff} as well as the average distance between pairs of points avg_{d_diff} is needed to compute the value.

$$f_s = \frac{avg_{t_diff}}{total_p} = \frac{\sum_{i=0}^{total_p} t_{p_{i+1}} - t_{p_i}}{total_p} \quad (3)$$

$$avg_v = \frac{avg_{v_diff}}{total_p} = \frac{\frac{avg_{d_diff}}{avg_{t_diff}}}{total_p} = \frac{\frac{\sum_{i=0}^{total_p} d_{p_{i+1}} - d_{p_i}}{\sum_{i=0}^{total_p} t_{p_{i+1}} - t_{p_i}}}{total_p} \quad (4)$$

Once both values are computed, they are respectively used to compute the thresholds using the equations Eq. 3 and Eq. 4 previously defined. Both thresholds, next to the restructured data extracted in the previous task, are given to the next module as input to generate the behavioral model.

4.2.BEHAVIORAL MODEL BUILDING MODULE

Behavior in the context of this project is understood as a series of observable patterns which define the way an individual moves around a space, considering attributes like preference and the environment setup. Thus, it is defined based on mobility patterns generated within a defined movement model. The goal of this second module is to take a series of georeferenced movement tracks and use them to build a behavioral model by firstly establishing the movement model within the data, and then detecting the relevant patterns within the model that signify certain mobility-based behaviors of interest. Once these patterns are detected, they can be analyzed to define the underlying model that generates them, which enables for the better understanding on the decisions and preferences an individual follows when moving, as well as for the prediction of future possible movements. In the main use case of this work, oriented at the generation of a framework for suspect investigation, the behavioral model allows to better analyze the motives between the suspect's movements, the possible connection between a certain suspect and others, and the prediction of possible future crimes.

To carry out the task of building the behavioral model based on the movement tracks, the module is divided into two main steps. Firstly, the underlying movement model within the data is extracted. This model defines movement as transitions between several points of interest the individual usually visits. Thus, the first step in this module consists of defining the points of interest within the data by implementing an adapted clustering algorithm. Once the POIs are defined, a second step is carried out, which consists of the extraction of the relevant behavioral patterns within the movement model defined. Although the first module is implemented equally for both the historic data as well as the new anomalous data, in this module, depending on whether we are analyzing historic data (which will define the expected behavioral patterns) or new anomalous data (which might have deviations from the established expected behavior), the second step will be carried out slightly differently. At the end of this task, the module returns either the behavioral model, or the anomalies profile, depending on the data being analyzed. The complete architecture can be seen in Figure 5.

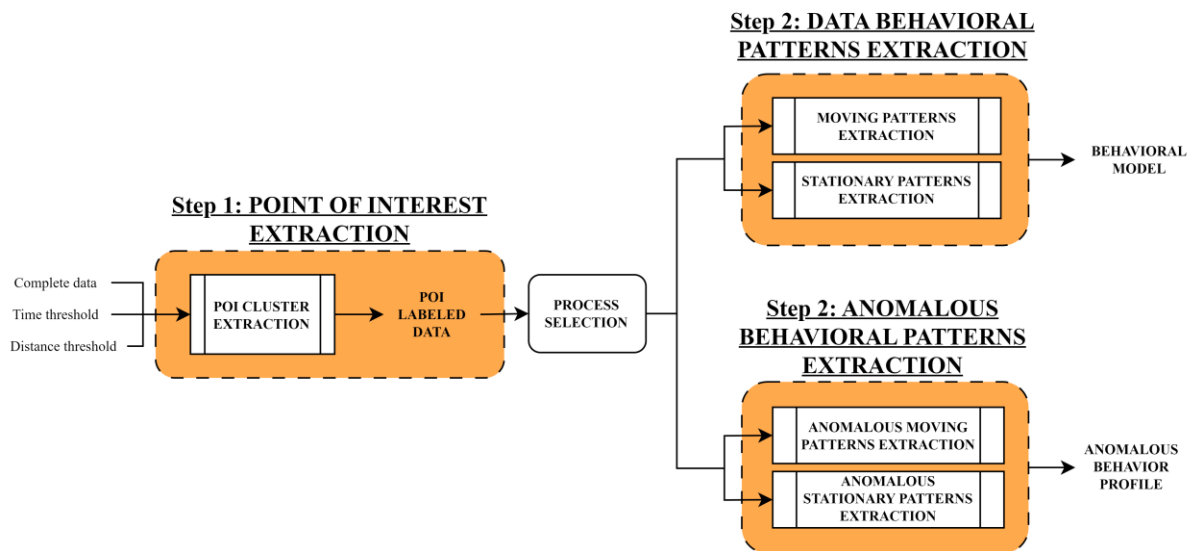


Figure 5. Structure of the behavioral model building module within the behavioral modelling and anomaly detection system

4.2.1. PLACES OF INTEREST EXTRACTION STEP

As previously mentioned, in the context of this work, we model behavior based on movement. Movement in this case is modelled as a sequence of transitions done between places throughout the day by an individual. Thus, the movement model can be structured like a Markov chain, where the places are the states and the individual makes transitions between them. However, in the movement model we additionally consider the movement traces generated both during the transitions (the routes taken to move between states), and the traces generated in the places when reached (the movement made in the states), where the individual is expected to stay in for a certain period before making

another transition. The behavioral model is based on this movement idea. Therefore, the first step in this behavioral model building module is to define the key elements needed to build the movement model, which are the places of interest that make up the states in said model.

Since movement in our case is understood as an individual transitioning between places in which he/she stays for a certain period, and seeing these movement traces like pairs of coordinates with a timestamp associated as previously explained, these states in the data can be defined by considering all the movement traces that were generated while the individual visited these certain places. Since it is expected for the individual to stay in the place, the points generated within the place will be close in space (as the individual does not move much) during a prolonged period. This is translated into the data as dense clouds of points with many data instances close together, as opposed to being sequentially positioned as expected when the individual moves between places. Therefore, the task of defining these places of interest becomes a task of identifying these dense clouds of points in the data.

A density-based clustering approach is used to detect these points of interest in the data. However, due to the nature of the data, where a highly transitioned route between places might also end up with a high point density over time, a time and distance-based algorithm is implemented, to consider density both in distance and time. The implemented algorithm is a slight adaptation of the existing Time and Distance Based Clustering (TDBC) algorithm [123]. The changes made to this algorithm were aimed at adapting it to our specific use case, by removing certain scenarios not relevant in this case, or by adding a more complex structure around the algorithm for the processing of multiple trajectories. Nevertheless, the final algorithm goes as follows. The algorithm considers two different ways in which these places of interest can be reflected. The first way is the one defined until now, in which a place of interest is defined when the individual has stayed within a certain distance threshold (has not moved greatly) for an enough amount of time (considering a time threshold). The second way places of interest are defined considers the limitation of real tracking devices, which often have slight data losses due to the environment circumstances (e.g. individual enters a subway station). In this case, a place of interest is considered if two consecutive points are close to one another spatially (within a distance threshold) but not in time, meaning that an enough time has passed between them to consider that some data has been loss and the individual has stayed at that location for a relevant enough amount of time. Therefore, considering both types of definitions for places of interest, the adapted TDBC algorithm is implemented.

This algorithm takes as input the pre-processed movement data obtained from the previous module, as well as the distance and time thresholds to consider. These thresholds will determine how close the points must be and how much time must have passed inside a cluster for them to be considered places of interest. Given these inputs, the algorithm processes each of the trajectories, and the spatiotemporal data instances that make them up one by one. Points are grouped together dynamically starting by the first one. Each time a new point is added, if the point is close enough to the group's centroid (within the distance threshold of the centroid), then the point is added. If a point is added, then the centroid is computed again and the next point is analyzed. If a point is not added, then the group is considered 'closed' (no more points are added) and checked to see if it could be considered a place of interest. A group of points is considered a place of interest if enough time has passed between the first and the last added points, meaning that the time difference is greater than the established time threshold. If the group is considered a place of interest, it is saved and a new group is started with the next point in the trajectory. If the group is not considered a place of interest, it is also saved. This is done because once the whole trajectory is processed, all the groups that were not labelled as places of interest are compared with each other and with the defined places of interest. If the centroid between two of these is within the distance threshold, they are merged and the time check is computed again to ensure enough time has passed for the new merged group. If it has, a new place of interest is defined. Once all the places of interest are defined within all the individual trajectories, they are compared across trajectories. Even in different trajectories, an individual is still expected to recurrently visit some of the same places of interest. The places of interest individually extracted are merged across trajectories considering distance once again to define the final total amount of unique places of interest and all the points, in all the trajectories, which are a part of them in the data.

After all these clusters (places of interest) are detected with the algorithm, they are labelled on the data. Thus, at the end of this first places of interest extraction step all the movement data given as input is now labelled with a series of cluster labels that determine which movement points are part of one of the places of interest, representing stationary behavior, and which are part of the transitions between places of interest (noise points), representing moving behavior.

4.2.2. BEHAVIORAL PATTERNS EXTRACTION STEP

Based on the movement model established in the previous step through the definition of places of interest, the behavioral model is built by detecting a series of behavioral patterns on the data. However, depending on whether we are working with the historic data or new data in which anomalies should be detected, the final model built will pursue a different goal, being in the first case to represent the usual and expected behaviors, and in the second to represent the new detected behaviors which might deviate from the previously established ‘normal’ behaviors. Nevertheless, in both cases, the behavioral patterns that are searched for in the data are the same, for them to be compared later.

The behavioral patterns can be divided into two different types, which are stationary behavioral patterns and moving behavioral patterns. Starting with the stationary behavioral patterns, these are all those behaviors showcased when the individual is ‘stationary’ in one of the places of interest. To limit the scope of possible behaviors that could be considered, and taking into account the police investigation use case considered for the development of the final framework, only one stationary behavior of interest is modelled, which is the duration of an individual’s stay in a place of interest. This stationary behavior is modelled and detected as follows:

- Duration of stay:
 - Modelled using a Gaussian distribution defined by a mean stay time μ_t and a standard deviation from said mean σ_t associated to each of the different places of interest. So, the duration of stay at a specific place of interest POI_i is modelled as $T_i \sim N(\mu_{t_i}, \sigma_{t_i})$.
 - Extracted from the data by computing the time difference between the first and last points of each of the data clusters that define a place of interest. The time differences are then used to compute the mean and standard deviation.

As for the moving behavioral patterns, most of the patterns included in the final model are of this nature. Once again, the scope of possible behavioral patterns to consider is limited to the ones detected as relevant in a police investigation or suspect tracking scenario, which in the case of moving behavior include the following behaviors: the usual speed an individual takes when transitioning between places of interest, the individual’s preference on the routes to take to move between places of interest, and the individual’s preference on where to go (where to transition) based on where he/she currently is. These behaviors are modelled and detected as follows:

- Speed:
 - Modelled using a Gaussian distribution defined by a mean speed μ_v and a standard deviation from said mean σ_v associated to each of the possible transitions between places of interest. So, the speed taken by the individual to move between two places of interest POI_i and POI_j is $V_{i,j} \sim N(\mu_{v_{i,j}}, \sigma_{v_{i,j}})$.
 - Extracted from the data by computing the spatiotemporal difference between moving points between POIs. The difference is computed in time and space to compute the velocity in which the individual is moving. The combination of all these computations are used to then define the final average speed of movement and the standard deviation needed to model speed.
- Route preference:
 - Modelled using empirical probabilities defined based on the observed usage frequency of each of the routes. From all the k routes that the individual takes between places of interest POI_i and POI_j , each defined as $R_{i,j}^{(k)}$, the preference $P_{R_{i,j}^{(k)}}$ (which can be understood as a probability of taking the route) the individual has for

each of these routes is computed as the amount of times that route is taken in the data divided by the amount of times the transition is made.

- Preferences are extracted from the data by first defining all the routes taken between each of the possible POI pairs, and then combining them into final routes by comparing how close the coordinates are. From the final routes, the preferences are determined as previously explained.
- Transition preference:
 - Modelled using empirical probabilities defined based on the observed frequency of transitions between places of interest. From all the possible places an individual can go to from a specific POI_i (which are all the remaining POIs), the preference on where to go to $P_{i \rightarrow j}$ (which again can be understood as a probability of making a certain transition) is computed as the amount of times a certain transition has been made in the data divided by the amount of times the individual moved from POI_i .
 - Preferences are extracted from the data by first defining all the transitions made from each of the POIs to other POIs and then computing the probabilities previously defined.

Considering these behaviors and the places of interest extracted in the previous step, the behavioral model is built, by analyzing each of the data points and interchangeably computing the stopping behavior and the moving behavior tasks, which extract from the data the given behaviors as explained previously. All the gathered patterns are then stored in a structured object for future use.

As mentioned previously, this process will have a slightly different output depending on whether the input is the historic data or the new anomalous data. Nevertheless, even for the definition of the anomalous behavior profile, the tasks implemented are like the ones previously explained, where each of the behaviors of interest are extracted and stored in a structured object for future comparison with the behavior model. Thus, in the first run of this behavior model building model the output of this second component is a behavioral model built with historic data, while for the remaining runs, anomalous behavior profiles are built to compare to the model and determine deviations from the established expected behavior.

4.3. ANOMALY DETECTION AND SCORING MODULE

Anomalies in the context of this work are understood as deviations, with a certain associated degree, from behaviors established as 'normal' or 'expected' from a certain individual. The detection of these deviations allows for the police force to identify unusual behavioral patterns that might indicate a change in environment or conduct, which could point to criminal or dangerous activity. The efficient detection of these deviations allows for better decision-making, as well as early interventions and more in-depth investigations, helping law enforcement in the tasks of avoiding and detecting crime. The goal of this third anomaly detection and scoring component is to detect and label relevant anomalies on new movement data based on a specified behavioral model. The detected anomalies are limited to the ones defined as highly intertwined with the behaviors considered for our use case. Furthermore, the anomalies on the data are labeled with a certain degree of deviation, which helps understand how anomalous (how much it deviates) a certain detected behavior is.

To carry this out, this final component carries out two main steps which can be defined as a first anomaly detection step, and a second anomaly labeling step, as can be seen in Figure 6. First, both the behavioral model and the anomalous behavior profile are given as input to the first step. This first step carries out a comparison between the patterns in both inputs using statistical methods. Depending on how similar the modelled patterns are between the behavioral model and anomalous profile, a deviation (or anomaly) degree is associated to each of them. Once these anomalies are detected, the points defining each of the behaviors in the data are labeled with the associated anomalies and their respective deviation degrees. At the end of this process, we have the original new anomalous data file with the anomalies labeled on each data point.

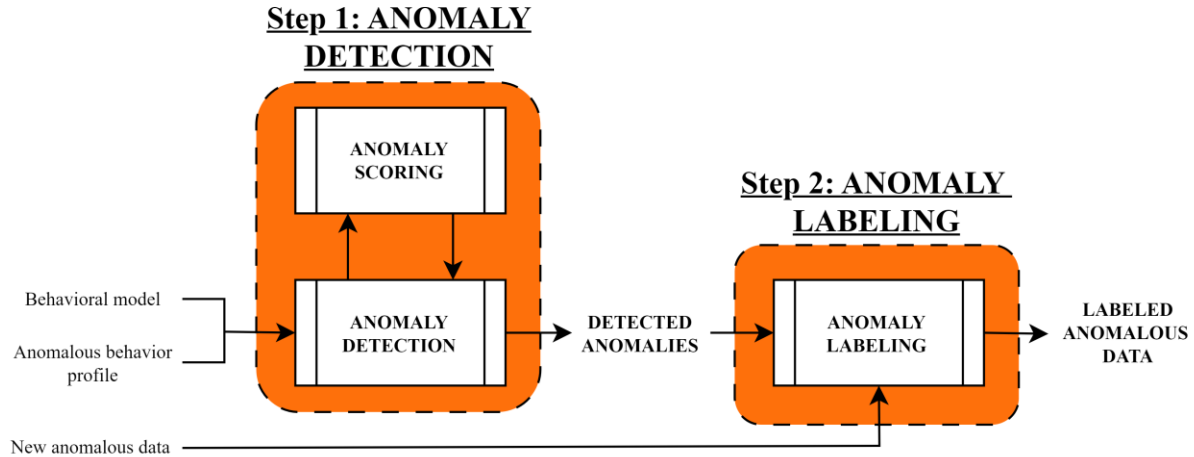


Figure 6. Structure of the anomaly detection and scoring module within the behavioral modelling and anomaly detection system

4.3.1. ANOMALY DETECTION STEP

Multiple anomalies can be considered in movement data, especially if behavioral patterns are also taken into account. Depending on the specific context of the anomaly detection process, the anomalies that could be considered as relevant might differ greatly. Therefore, to limit the scope of possible anomalies to be detected by the system, we define a pool of relevant anomalies which are highly connected to the behavioral patterns considered as relevant in the context of this work. These are the only anomalies considered by this first anomaly detection step.

Depending on how the associated behavioral pattern is modeled, the anomaly will be detected using different approaches, but always based on statistical inference ideas. The main idea behind this anomaly detection process is that if a certain new pattern does not follow the same model as the established expected patterns, then said pattern is considered anomalous. To carry out this process of comparing new patterns with the expected model, a hypothesis test is formulated for each type of anomaly, in which the null hypothesis (the statement to accept or reject) is always that the new observed pattern is not anomalous. Thus, given the anomalous behavior profile, each of the possible patterns are compared to the models through inference techniques and a p-value is computed in each case, which will be a way to measure how far from ‘normal’ the new patterns are. This value will be the one used to associate a deviation degree to each of the anomalies later.

Thus, due to the complexity of this process, this first anomaly detection step is divided into two main tasks, which are the anomaly detection task and the anomaly labeling task. Both tasks work interchangeably, as anomaly types are detected and labeled one by one.

Starting with the anomaly detection task, as mentioned previously, depending on how certain behavioral patterns are modeled, the approach is slightly different, as certain models have an established way in which p-values can be computed, while others do not. Thus, for each of the considered anomalies, we must establish what the anomaly entails, and how it could be detected based on the associated expected behavior model. The anomalies considered, and their detection approach, can be defined as follows:

- **Duration of stay anomaly (Type I)**
 - Definition: Deviation from stationary behavior surrounding the usual duration of the stays in each of the places by the individual. The deviation occurs when the individual stays much longer or much shorter than usual at a specific place.
 - Detection: Associated to duration of stay behavioral patterns, modeled using a Gaussian distribution $T_i \sim N(\mu_{t_i}, \sigma_{t_i})$. The null and alternative hypothesis defined for the comparison between the new detected patterns and the expected duration of stay model are as follows:
 - *Null hypothesis* (H_0): “The new duration of stay is consistent with T_i .”
 - *Alternative hypothesis* (H_1): “The new duration of stay deviates from T_i .”

The hypothesis test is implemented using the parametric statistical measure known as Z-score (Z of Gauss), which defined how many standard deviations a value is away from the mean of a Gaussian distribution. Thus, the p-value is computed considering the new duration of stay value T and the Gaussian that models the expected patterns following Eq. 5.

$$p_{val_{stay}} = 2 * \left(1 - \Phi \left(\left| \frac{T - \mu_{t_i}}{\sigma_{t_i}} \right| \right) \right) = 2 * (1 - \Phi(|Z_{score}|)) \quad (5)$$

- **Speed anomaly (Type II)**

- Definition: Deviation from moving behavior surrounding the usual speed the individual takes when moving between places of interest. The deviation occurs when the individual goes much faster or much slower than usual when moving between two specific places.
- Detection: Associated to speed behavioral patterns, modeled using a Gaussian distribution $V_{i,j} \sim N(\mu_{v_{i,j}}, \sigma_{v_{i,j}})$. The null and alternative hypothesis defined for the comparison between the new detected patterns and the expected speed model are as follows:
 - *Null hypothesis (H_0)*: “The new speed is consistent with $V_{i,j}$.”
 - *Alternative hypothesis (H_1)*: “The new speed deviates from $V_{i,j}$.”

The hypothesis test is implemented using the same Z-score (Z of Gauss) measure as the previous anomaly. Thus, the p-value in this case is computed similarly, by considering the new speed V and the Gaussian that models the expected patterns following Eq. 6.

$$p_{val_{speed}} = 2 * \left(1 - \Phi \left(\left| \frac{V - \mu_{v_{i,j}}}{\sigma_{v_{i,j}}} \right| \right) \right) = 2 * (1 - \Phi(|Z_{score}|)) \quad (6)$$

- **Suboptimal route anomaly (Type III)**

- Definition: Deviation from moving behavior surrounding the route taken when moving between places of interest. The deviation occurs when the individual takes a route that is longer in distance or takes longer in time (is suboptimal) instead of taking the shortest route possible when moving between two specific places.
- Detection: Associated to route preference behavioral patterns. New route is compared to the defined optimal route, and the value of the areas generated from their intersection are extracted. The expected value of the areas of two equal routes is modeled using a Beta distribution $A_{eq} \sim Beta(\alpha, \beta)$. The null and alternative hypothesis defined for the comparison between the new computed areas from comparing the new route with the optimal and the expected area model are as follows:
 - *Null hypothesis (H_0)*: “The new area values are smaller or consistent with A_{eq} .”
 - *Alternative hypothesis (H_1)*: “The new area values deviate from A_{eq} .”

The hypothesis test is implemented using the CDF of the Beta distribution adjusted to the data, performed on the right tail, since the values will be more anomalous the furthest away from 0 they are. Thus, the p-value in this case is computed considering only the new area values A and the CDF of the Beta that models the expected patterns following Eq. 7.

$$p_{val_{suboptimal}} = 1 - F_{Beta}(A) \quad (7)$$

- **Unlikely route anomaly (Type IV)**

- Definition: Deviation from moving behavior surrounding the route taken when moving between places of interest. The deviation occurs when the individual takes a route not usually taken by him/her when moving between two specific places, instead of taking his/her usual route.
- Detection: Associated to route preference behavioral patterns. New route is compared with all the points that are part of the closest non-anomalous route between two places to compare the local density between them using the Local Outlier Function (LOF) algorithm. LOF outlier values are associated with each of the points in the new route, where points with a similar local density as its neighbors are less anomalous than points with less local density than its neighbors. If the new route is likely (many movement points have passed through it), the distribution of these outlier points will be like the distribution of the points in the behavioral model. The expected value of the outlier of the likely routes is modeled using a non-parametric approach, since the LOF values do not follow an established known distribution. The null and alternative hypothesis defined for the comparison between the new computed LOF outlier values from comparing the new route with the closest non-anomalous route and the expected LOF value model are as follows:
 - *Null hypothesis (H_0)*: “The distribution of the LOF values of the new route is smaller or consistent to the distribution of the LOF values of the expected routes.”
 - *Alternative hypothesis (H_1)*: “The distribution of the LOF values of the new route deviates from the distribution of the LOF values of the expected routes.”

The hypothesis test is implemented using the Mann–Whitney U statistic, used to compare non-normal distributions. This statistic returns the needed p-value.

- **Unlikely transition anomaly (Type V)**

- Definition: Deviation from moving behavior surrounding the transitions between places of interest. The deviation occurs when the individual goes to a place, he/she does not usually go to from the current place the individual is at (makes an unlikely transition).
- Detection: Associated to transition preference behavioral patterns. New transition (with whatever route) is compared with all the points that are part of the transition between places of interest, again to compare the local density between them using the Local Outlier Function (LOF) algorithm. If the new transition is likely (many movement points between two places), the distribution of these outlier points will be like the distribution of the points in the behavioral model. The expected value of the outlier of the likely transitions is also modeled using a non-parametric approach. The null and alternative hypothesis defined for the comparison between the new computed LOF outlier values from comparing the new transition with the existing transition points and the expected LOF value model are as follows:
 - *Null hypothesis (H_0)*: “The distribution of the LOF values of the new transition is smaller or consistent to the distribution of the LOF values of the expected transitions.”
 - *Alternative hypothesis (H_1)*: “The distribution of the LOF values of the new transition deviates from the distribution of the LOF values of the expected transitions.”

The hypothesis test is implemented using the Mann–Whitney U statistic, which returns the p-value.

The parallel anomaly scoring task takes the p-values extracted from the anomaly detection process and computes an associated anomaly degree to each of the detected anomalies. The p-value extracted indicates the chance of obtaining a value as or more extreme than the one used, assuming that the null

hypothesis is true. Translated to our use case, the p-value indicates the chance of having a pattern as extreme or more extreme than the one detected if the pattern detected is not anomalous. It can be understood as a way to measure how ‘weird’ the new pattern is under the null hypothesis. Thus, if the p-value is close to 1, then that indicates that the new pattern is close to ‘normal’, while if it is close to 0, it indicates that the new pattern deviates from the ‘normal’, meaning that it is anomalous. In our case, the goal is to associate each anomaly detected with a deviation degree going from 0 to 1 (that can be translated to a deviation percentage by multiplying it by 100). When the degree is close to 1, then the anomaly detected deviates greatly from the established normal and vice versa.

Therefore, to obtain the anomaly degree associated to each of the p-values extracted ($AD_{p_{val}}$), a direct linear transformation (Eq. 8) is implemented, that computes the anomaly degree as the direct opposite of the p-value. Each of these new anomaly degrees are associated to each of the anomalies accordingly for the next labeling step.

$$AD_{p_{val}} = 1 - p_{val} \quad (8)$$

4.3.2. ANOMALY LABELING STEP

The goal of this final anomaly detection component is to have the anomalies labeled directly on the data for future processes to be able to associate locations and timestamps to the detected deviations for the analysis of how deviations occur and change in space and time. However, the inputted data often includes many movement points to be labeled, and with many being part of various anomalous events rather than only one. Therefore, the detections carried out in the previous step must be formatted in a way that can be used to label the data.

This final anomaly labeling step is aimed at properly labeling each of the points with the needed data to associate the anomalies and give enough information for them to be interpretable by later modules. To do so, for each of the points contained in the new anomalous movement data file given, a dictionary is defined. The dictionary has as keys the labels given to each of the anomalies, these labels being ‘1’, ‘2’, ‘3’, ‘4’. and ‘5’, which refer, respectively, to the anomalies of type I, II, III, IV, and V. If a specific point has a certain anomaly, the new key will be inputted to its dictionary, with the value being the anomaly degree associated with the anomalous events. If a point is not part of an anomalous event of a certain type, then the key is not included in the dictionary. Thus, at the end we have a series of georeferenced chronological points, each with an anomaly label associated, which includes all the anomaly type incidents the point is a part of, and how much that incident deviates from the expected behavioral patterns of the individual who generated the data.

5. DESIGN AND IMPLEMENTATION OF A SIMULATOR MODULE FOR SYNTHETIC POLICE REPORT GENERATION

In the context of police work and investigations, in which this project is developed, a crucial source of information are reports developed by the police forces themselves. The system of police reporting allows the police to keep track of all the knowledge gathered regarding a specific suspect or crime both from external sources and internal investigations. This system is a key information registration and tracking process with a base methodology shared across police forces all over the world. Since the goal behind this project is to build a framework that could help the police force in their investigation tasks, it is essential to properly consider the information gathering systems already in place in the force, as well as all the sources that would be available for them.

Police reports are considered in the development of this framework as one of the main sources of information available, since they carry information and internal insights on the status of investigations, often only available to the police. However, due to the sensitive nature of these reports and the data they contain, they are often not available to the public. Thus, to properly tackle the lack of availability of real examples of this data source, a police report simulator is developed. This simulator is aimed at generating realistic and structured police reports that mimic the ones developed by the real forces for the proper and complete development of the framework proposed, which is desired to be able to account for these reports and properly integrate the data in them.

Although the generated reports do not require to follow the crime lifecycle of any real criminals, they are expected to follow a realistic sequence of events that could be seen in a real investigation for them to be useful for the evaluation of the final framework. Therefore, a complex architecture is built to define this simulator with the goal of not only generating structured reports, but for the reports to include coherent and realistic information that can be properly used later to contextualize data and to reach new conclusions. This architecture is made up of three main components, which work sequentially to gather useful information and to build context-rich and structured reports. These components, which can be seen in Figure 7, are defined as the crime extraction and grouping component, the timeline building component, and the report simulation component.

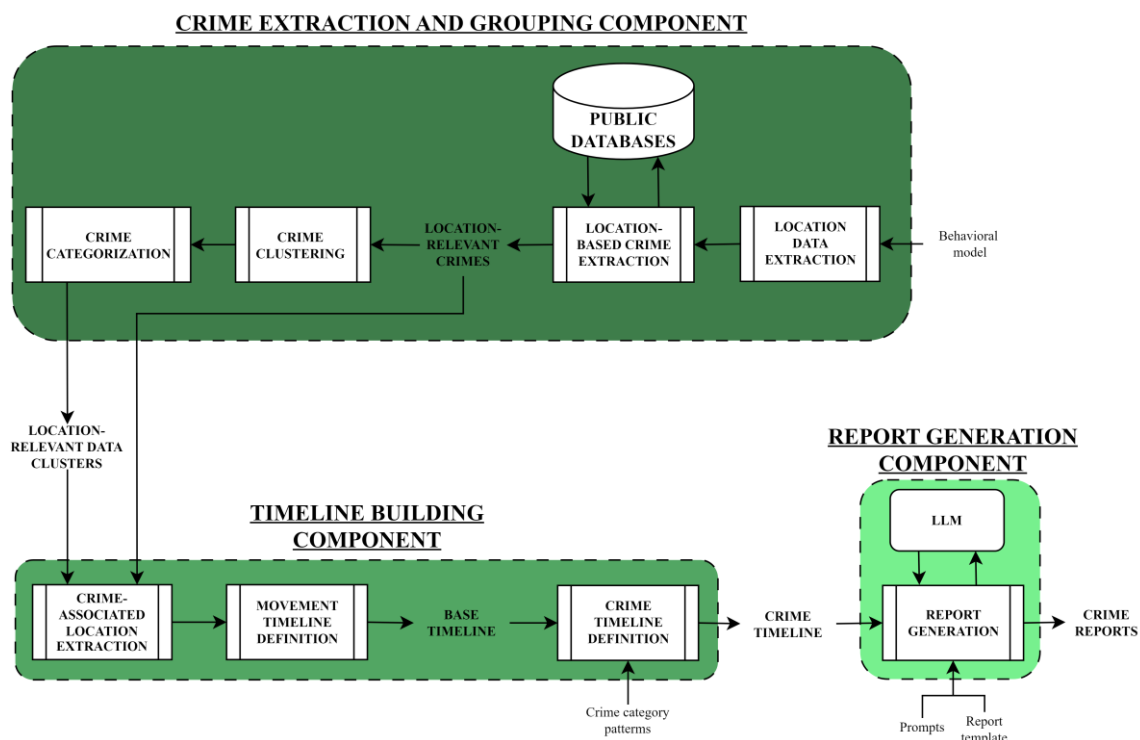


Figure 7. Police report simulation module architecture

To determine if the reports are being properly used by the data contextualization and interpretation module, they must be related to the suspect profiles used for the validation process. The three mentioned components that make up this structure make it possible for the generated reports to consider the frequently visited places by the suspect and the usual crimes that occur in said places for the reports to be aligned and coherent with the actual locations studied further along in the data. To do so, firstly the crime extraction and grouping component takes the significant locations extracted from a suspect's movement tracks and contextualizes them with the reported crimes that have occurred in that area through open-source crime databases. Once each of the locations have a series of crimes associated, these are grouped together based on how similar they are to reduce all the crimes to a series of possible categories. Based on the crime category the user wishes the reports to be based on, the timeline building component takes as input all the crimes in a specific category and generates a realistic timeline of similar crimes occurring on a coherent sequence of dates in the locations given. Based on this timeline, the report simulation component generates a series of reports. Depending on what the user specifies, the generated reports can be tied to the specific suspect (polices knows who committed the crime) or can be simply reporting crimes with an unknown perpetrator (police must find out who committed the crime).

Given this general architecture, this section will focus on defining each of the components, their functionalities, and their implementation into the module.

5.1.CRIME EXTRACTION AND GROUPING COMPONENT

The main goal of this module is to generate realistic reports which integrate well with the general framework and all the processes that compose it. Since in other modules of the framework's work cycle real crime data is used to contextualize the available information, for the whole process of data analysis to be coherent, the reports should somewhat refer to actual crimes occurring in certain places. If the reports are completely made up without considering the actual places the suspect has visited and the reported crimes occurring in those places, the reports will give different, and even contradicting, information to the one obtained by future modules which use actual data gathered from external sources. This could make it hard to correctly simulate how a real implementation of the system would look like, where the reports contain information on actual crimes investigated by the police, and to evaluate the usefulness of the information given by the reports, as the fake reports could make it harder to conclude things about the investigation rather than enhancing the available information. Therefore, the first component of this simulation module is aimed at tackling this issue.

The crime extraction and grouping component is aimed at properly analyzing all the significant locations visited by a suspect and extracting all the known and relevant crimes occurring in said locations for them to be included in the reports. However, for the reports to show a realistic sequence of events, with known criminal patterns, the crimes detected must be analyzed and selected accordingly. To carry this out, a component with two main steps is developed, as can be seen in Figure 8.

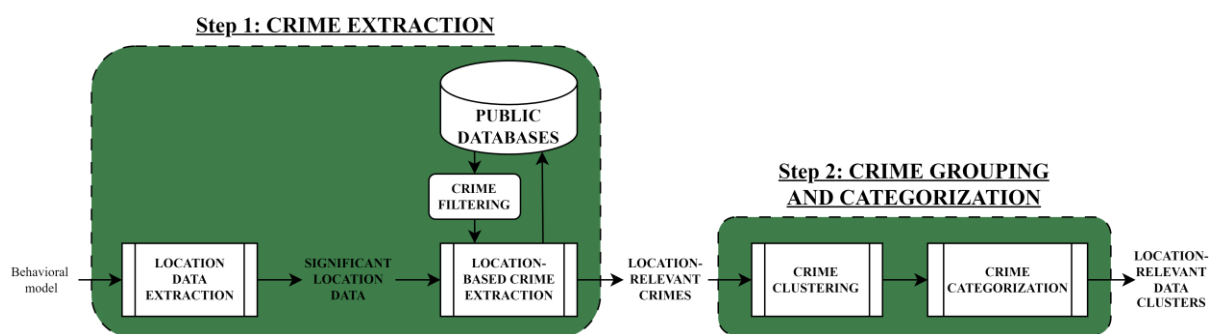


Figure 8. Structure of the crime extraction and grouping component within the report simulation module

Firstly, a crime extraction step is carried out aimed at extracting the data on the significant locations and all the relevant crimes occurring in said locations. Then, a crime grouping and categorization step is carried out, to properly categorize all the detected crimes into a series of labels. To maintain a

coherent and realistic sequence of reports, the simulated suspect will only commit crimes of a certain category. Thus, this first component will define all the possible crimes to be considered given the movement patterns of a suspect, and will properly categorize those crimes for the simulated reports to follow a realistic *modus operandi*.

5.1.1.1. CRIME EXTRACTION STEP

The crime extraction step is composed of two tasks which work sequentially to properly determine all the relevant crimes which a certain suspect could have committed given the location points and the movement patterns extracted. These two tasks can be defined as a first location information extraction task, and a second location-based crime extraction task.

The first location information extraction task is aimed at properly defining all the location the suspect usually visits or stays at, and all the relevant information surrounding them which will be needed for future steps in the simulation workflow. This information includes the specific coordinates of the location, the street and city associated with said coordinates, and all the times the suspect was at the location. To efficiently extract this information, this first task makes use of the behavioral model defined around the suspect based on the movement tracks gathered. As explained in the previous section (Section 4), with the behavioral model we have defined certain information regarding each location labelled as significant (each place of interest) in the suspect's movement patterns. In this specific case, the most relevant information includes the coordinates defining each of the locations (which will be the defined centroids of the POIs extracted in section 4.2) and the timestamps from all the points making up the locations (all the points in the clusters defined in section 4.2), which will help us determine all the unique dates in which the suspect was in that location.

As for the information regarding the city and street name associated with certain coordinates, a geocoding service based on the OpenStreetMap is implemented. This geocoder allows us to translate numeric geographic coordinates into an address through inverse geocoding. In the case of this first task, only the information regarding the specific city the coordinates are in, as well as the street, are extracted. With the city and street information, an object is built for each of the significant locations defined in the behavioral model, containing, as mentioned, the coordinates that define it, all the unique dates in which it was visited by the suspect, the city it is in, and the name of the street it is located at. This information regarding each of the significant locations is given to the next task.

The location-based crime extraction task is aimed at extracting all the relevant crimes that have occurred in a specific area. To carry this out, open crime databases are used. Unlike information found on the internet, which might refer to crimes not yet fully reported by the police force, public crime database APIs have access to data contained in government and police available databases. This information is more reliable and realistic, as it contains crimes that have been fully investigated and reported by the police. Depending on the area and country, the available sources and the APIs that have access to them can vary greatly, as well as the information retrieved by said API. Therefore, this second location-based crime extraction task starts by implementing a data filtering process. This process is aimed at accessing the available APIs and filtering all the irrelevant data. In our case, the relevant data will include documents reporting crimes occurring in the same city in which the suspect moves and that specify the type of crime committed and the specific location of the crime (either through coordinates or through address). Given all the documents, a new process is carried out to extract the relevant crimes.

Although all the documents regarding the crimes occurring in a specific city have been extracted, only those containing crime occurring in the areas and locations visited by the suspect are significant when generating the reports. Thus, an additional location-based filter is implemented on the extracted documents from the API, in which only the crimes committed within an established distance of the significant locations associated with the suspect are considered. This is necessary because all the processes in the framework are heavily location-based. Not considering location when defining the final crimes to consider for the report simulation might render the generated reports useless in future modules. Therefore, for each of the extracted crime data instances, the location information is compared with the significant places' location data defined in the previous task. Depending on

whether the crime data instance has location defined through coordinates or through an address, the location will be compared with the significant places directly by computing the distance between coordinates, or will first be transformed with the previously explained geocoder into coordinates and then compared through distance. In both cases, the shortest distance between the crime's location and the suspect's significant place will be computed with the Haversine formula (Eq. 9), which, considering the Earth as a sphere (approximated with a radius of $R = 6.371$ km), allows us to get an approximation to the shortest distance between two points on the Earth's surface. Defining the two points as $coord_1 = [lat_1, lon_1]$ and $coord_2 = [lat_2, lon_2]$, the shortest distance d can be computed from the central angle a between points obtained from the Haversine formula (Eq. 10) as follows:

$$a = \sin^2 \frac{lat_2 - lat_1}{2} + \cos(lat_1) * \cos(lat_2) * \sin^2 \frac{lon_2 - lon_1}{2} \quad (9)$$

$$d = 2 * R * \arcsin(\sqrt{a}) \quad (10)$$

Thus, a radius defined in kilometers is established as the maximum distance that a crime can have occurred from one of the suspect's significant places for the crime to be considered relevant (consider that it occurred in the significant location) and used for the report generation. By implementing this filter, at the end of this first crime extraction step, we have all the crimes we could consider for the report simulation, each with one of the suspect's significant locations associated to them.

5.1.2. CRIME GROUPING AND CATEGORIZING STEP

Criminals often have defining and widely studied criminal patterns, even with petty and non-premeditated crimes. Most criminals stick to a specific modus operandi (or MO for short), meaning that they tend to commit crimes that are similar in nature and are carried out in similar fashion. However, this is highly dependent on the type of crime, as certain crimes have been proven to have stronger specialization patterns, meaning that criminals committing those crimes have a higher chance of sticking to that crime instead of changing to other crimes, while other crimes are more often committed by versatile criminals who carry out various crimes, often combining them (e.g. drug/alcohol abuse and traffic offense). This is essential to be considered when simulating the reports, as we need the reports to follow a real criminal pattern for the future modules to properly attribute certain reports to suspect and to consider these patterns when drawing conclusions.

To consider these patterns in the report simulation, instead of simulating the report of all the extracted crimes, only crimes of a specific nature or which follow specific incident patterns (like often occurring together) will be used for the reports associated to a specific individual or area. This second crime grouping and categorizing step is aimed at dividing all the extracted crimes into groups which include crimes often committed jointly or that are similar in nature, and then associating a certain crime category to each of these groups, to encompass the overall nature of the crimes included for the user of the simulation to be able to establish the type of crimes he/she wishes to simulate in the reports later on.

A study has been carried out to determine the type of crimes often included in public databases and the labels the police put them under. Based on the crimes that are under the previously given definition (with high specialization likelihood or often combined with other crimes), the following categories have been defined for the categorization part of this second step: theft/burglary, fraud, vandalism, assault/violence, sexual offenses, threats, drugs/narcotics, firearms, domestic violence, traffic/DUI, public order, administrative/legal, and reports/incidents. Based on this category, a clustering and categorization process is carried out in this second step. To carry out all the needed processes, a series of external models and algorithms have been implemented, all included in Table 2.

Table 2. Existing tools used in the development of the crime clustering and categorization step

TOOLS	DESCRIPTION	RELEVANCE	USE
HDBSCAN algorithm	Density-based clustering algorithm based on the existing DBSCAN	Generates clusters based on hierarchy and density, so no previous specification of the	For crime grouping in the crime clustering

	algorithm, which implements density and hierarchical-based clustering.	- number of clusters is required - Allows for the generation of irregular clusters, with different shapes and sizes - High-performing in the clustering of embeddings due to their high-dimensional and unevenly distributed nature	task
all-MiniLM-L6-v2 model	Text embedding model from the Sentence Transformer family. It has 22.7 million parameters, and considers sentence semantics when creating vectors.	- Fast performance with compact size, easy to run locally - High-performing for short sentences and words - Generates normalized and dense embeddings, allowing for approaches like cosine similarity to directly determine semantic closeness	For crime group categorization in the crime categorization task

The process in this second step is divided into two main tasks as briefly mentioned before: a crime clustering task and a crime categorization task. Starting with the first task, all the crimes extracted from the previous step, which have been determined to be relevant given the movement patterns of the suspect, are divided into groups. The goal of this first crime clustering task is to divide all the crimes into different groups, clustering similar crimes together so that they can also be considered together when simulating reports. To carry this out, an embedding-based clustering approach is carried out. Firstly, all the crimes are vectorized to transform them into numeric representations using the embedding model previously defined (all-MiniLM-L6-v2). Once we are working with numeric values, a clustering algorithm is implemented. Since some types of crimes are more prone to occurring than others, we expect the groups to be irregular, with some containing more vectors than others. Furthermore, although we have a set of categories defined, we do not know the exact number of groups to be created, as certain crime categories might not be included in the crimes extracted from the previous step. Finally, our main goal is to group together crimes that are similar in nature. Given that we are working with short crime descriptors, often reduced to a single word, we are looking to group together the descriptors which are semantically similar, meaning that their vectors are close in the vectorized space. Therefore, a density-based approach is regarded as the best solution to tackle all these constraints, as it groups together vectors which are close together (which generate highly dense areas) without the need of an initial input specifying the number of clusters to generate, and which allow to generate clusters with different sizes and densities.

For this clustering task, the algorithm implemented is the HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm, since it works well with high-dimensional spaces, like embedding spaces, it automatically detects noise, and it generates hierarchical density trees, which is useful in this case since it is expected for some crime descriptors to be nested inside other more general descriptors. Due to the embedding model we use (all-MiniLM-L6-v2), the generated embeddings are normalized, so the default distance used by this algorithm (Euclidean distance) can be directly used to determine the semantic closeness between vectors, as the Euclidean distance on normalized vectors is proportional to the Cosine similarity, which is the similarity metric often used for embedding comparison. Thus, at the end of this first task, crime similar in nature are grouped together.

The second crime categorization task takes as input the groups defined on the previous clustering task and associates one of the 13 categories previously defined. There is no limitation on the number of groups which can be labelled with a certain category, as groups under the same category which were not grouped together in the previous task can give a useful insight into crimes that might not be

similar in description, but are similar in nature. Thus, all the groups are compared to all the categories and are labelled with the most similar category. This comparison is carried out using a similar approach as the one used for the clustering, where the categories are embedded (using the same model) and transformed into numeric representations in a higher dimensional space which are easier to compare. The cosine similarity is computed between the centroid of each of the groups and the category embeddings. The category with the highest similarity is the one associated with each group. At the end of this step, all the crimes previously extracted as relevant are labelled under a specific category.

5.2. TIMELINE BUILDING COMPONENT

Since real crimes and locations are considered when defining the content of the reports, their content is expected to be realistic and aligned with the other data considered in the final framework. However, as previously mentioned, criminality has associated patterns, which should also be considered. Although specialization and volatility patterns have already been considered in the previous step, two additional and relevant patterns in crime must also be taken into account for the report timeline to make sense: crime frequency and location variation. Firstly, depending on the crime, the frequency in which the perpetrator commits it might vary. For example, crimes related to theft are often committed multiple times in a week, while crimes like violent attacks or murder might be committed with more time in between. Secondly, also depending on the crime, the perpetrator might be more prone to changing locations when recommitting the crime, or staying on the same area. For example, fraud-prone criminals often change locations to avoid being traced, while other criminals, like those who commit vandalism, might like certain areas more than others.

From the last component, we have a series of possible crimes labelled under a specified number of categories and with locations associated to them (which are the significant locations detected when modelling the suspect's behavior). The goal of this second timeline building component is to generate a realistic timeline of events which considers the following:

- The crimes committed by the same individual must be consistent with the type of crimes listed under one of the categories. The user must be able to specify the type of crimes to be reported in the created documents. To consider this, the groups generated in the previous step, and the categories they were labelled with, must be used.
- The crimes reported must be coherent with the frequency and location variability patterns attributed to them. If the category of crimes to be reported has a high or low frequency and location variability associated with them, the reports must also reflect this behavior.
- The crimes reported must be consistent with the movement data they are based on, meaning that they must occur in the locations visited by the suspect at times where the data showcases that the suspect was there. The locations associated with each of the crimes labelled in the previous component must be considered.

However, considering all this information is not a simple task. The normal movement timeline of when a suspect was at a certain point might not be aligned with the crime patterns, as, under the initial assumption that the suspect could have committed all the crimes in all the locations at all the times, the frequency and location variability of the crimes might not be realistic. Thus, to consider both the real movements of a suspect, as well as the crimes to be simulated in the reports, this complex timeline building component is defined. The workflow of this component is made up of two steps which work sequentially, as seen in Figure 9. They can be defined as the movement timeline extraction step and the crime timeline definition step.

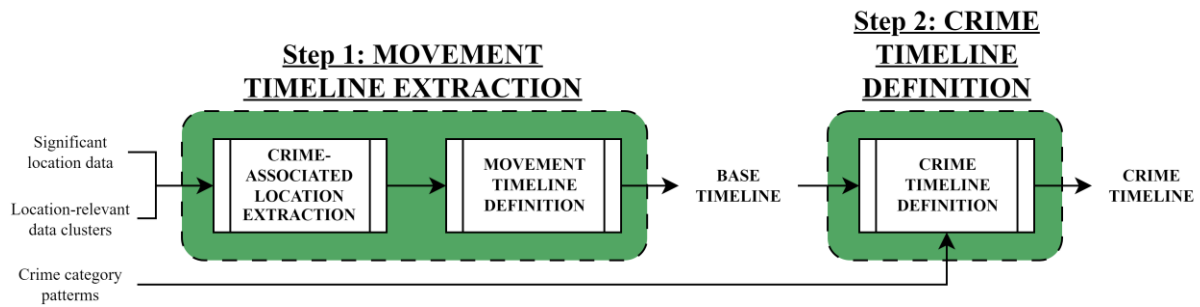


Figure 9. Structure of the timeline building component within the report simulation module

The first movement timeline extraction step is meant to extract from the data the timeline of actual movement followed by the suspect, including locations and dates, and to associate each of the locations with the crimes that were committed there which are under the category selected by the user. This timeline will work as a base to define when and where the suspect could have committed a crime and which crime it could have been. From this base timeline, the goal is to select the needed location and time pairs for the final timeline to follow the crime patterns defined for the category selected. This is carried out in the crime timeline definition step through a dynamic programming approach, where the goal is to generate the longest possible sequence of location-time pairs considering the frequency and location variability constraints.

5.2.1. MOVEMENT TIMELINE EXTRACTION STEP

As previously explained, the simulated reports must be closely connected to the other data used in the framework for their use in the workflow to be properly evaluated, as well as to simulate the information that could be gathered from real police reports. Therefore, the simulated reports must contain information that could be linked to the suspect under investigation. The locations the suspect usually visited as well as the dates in which they were visited play an important role in creating a connection between these reports and other data sources (movement tracks, behavior model, etc.). Thus, before generating the timeline of crimes the various simulated reports will follow, a base timeline is generated with all the possible combinations of time and place in which a crime could occur for it to be able to be linked back to the movements of the suspect. Essentially, we want the simulated reports to report events that could have been committed by the suspect based on where he/she was at certain dates.

To carry this out, a movement timeline extraction based on two main tasks is carried out. Firstly, given the desired category by the user, all the locations with associated crimes labelled under that category are taken. This ensures that only the locations where the actual crimes took place are considered, so that future possible contextualization tasks return data that aligns with what the reports say about the area. Secondly, for all the given locations, the dates in which the suspect visited said locations, extracted in the previous component, are defined and location-date pairs are generated. These pairs are organized in chronological order to generate a timeline of all the locations visited at the dates in which they were visited in which the suspect could possibly commit a crime. This timeline is the base of the future final crime timeline, as only in the dates and locations given can the final reported crimes occur.

5.2.2. CRIME TIMELINE DEFINITION STEP

To properly simulate actual crimes committed by a criminal, as previously mentioned, it is necessary to consider the actual spatiotemporal patterns that criminals committing certain crimes showcase. To generate the final crime timeline in this crime timeline definition step, the movement timeline generated in the previous step must be properly analyzed and transformed by eliminating the necessary location-time pairs for the remaining pairs to follow a realistic pattern of crime.

The first task to carrying this out is to properly define the patterns for each of the crime categories. Since all the crimes used are under one of the categories, the patterns established for each of the categories will be the one used for each of the crimes. Furthermore, due to the unique and volatile

nature of crime, to limit the scope of possible criminal patterns, the crime frequency and location variability are reduced to a limited number of possible values they can take, which can be defined as follows:

- Possible crime frequency values associated with a crime category:
 - *'Daily'*: The criminal commits the crime every 1 to 3 days.
 - *'Weekly'*: The criminal commits the crime every 1 to 2 weeks, or every 7 to 14 days.
 - *'Monthly'*: The criminal commits the crime every month or month and a half, or every 28 to 31 days.
 - *'Yearly'*: The criminal commits the crime once a year or a year and a half, or every 360 to 400 days,
- Possible crime location variability values associated with a crime category:
 - *'Non variable'*: The criminal does not change the location of the crime when recommitting, e.g. domestic violence.
 - *'Mildly variable'*: The criminal slightly changes the location between crimes, but often shortly returns to the previous locations, e.g. vandalism.
 - *'Highly variable'*: The criminal greatly changes the location between crimes, not returning to previous location until enough time has passed, e.g. traffic violation.

Considering these possible patterns, each of the crime categories is modelled considering a crime frequency value and a crime location variability value. The labels have been defined based on existing research on the behavior of criminals committing certain crimes [124] [125] [126], but both the possible values of the crime patterns as well as the patterns associated to each category can be changed and adapted by the user to any specific use case desired. In the case of this implementation, the given labels are as follows:

- **Theft/burglary**: A daily crime frequency and a highly variable location variability.
- **Fraud**: A weekly crime frequency and a highly variable location variability.
- **Vandalism**: A daily crime frequency and a mildly variable location variability.
- **Assault/violence**: A monthly crime frequency and a mildly variable location variability.
- **Sexual offenses**: A yearly crime frequency and a mildly variable location variability.
- **Threats**: A weekly crime frequency and a highly variable location variability.
- **Drugs/narcotics**: A daily crime frequency and a mildly variable location variability.
- **Firearms**: A monthly crime frequency and a highly variable location variability.
- **Domestic violence**: A weekly crime frequency and a non-variable location variability.
- **Traffic/DUI**: A yearly crime frequency and a highly variable location variability.
- **Public order**: A monthly crime frequency and a mildly variable location variability.
- **Administrative/legal**: A yearly crime frequency and a non-variable location variability.
- **Reports/incidents**: A monthly crime frequency and a non-variable location variability.

Based on these patterns, the final crime timeline can be sculptured from the defined movement timeline. Given a specific crime category and a timeline containing the possible locations and dates the crimes could occur at, the goal is to generate the longest possible new sequence of these location-date pairs (the longest timeline) which satisfies the crime frequency and location variability constraints defined by the crime category. This is done through a dynamic programming approach. Dynamic programming allows us to decompose the problem of generating the longest timeline sequence into smaller, and more easily solved subproblems. Specifically, our problem of generating timeline could be compared to the known Longest Increasing Subsequence (LIS) problem as our aim is to determine the longest sequence of established elements (location-date pairs) which satisfies a series of restrictions. Instead of trying all the possible combinations, which given our type of data could end up being highly costly, with the dynamic processing approach, this task can be carried out more efficiently. A visual breakdown of what the algorithm does can be seen in Figure 10.

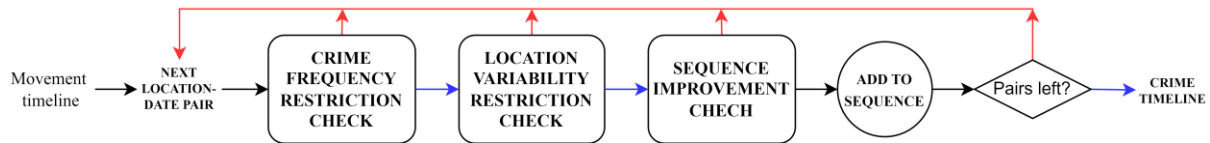


Figure 10. Architecture of the dynamic programming algorithm implemented in the timeline building component

What is done in this task is that, given all the possible pairs that could be added to the final timeline, we choose whether we add them or not based on the restrictions given. Firstly, the crime frequency restriction is checked, to see whether the expected time between crimes (previously established by the crime pattern model) has passed. If sufficient time has not passed, then the pair is discarded and the next pair is analyzed. If the sufficient time has passed, then the location variability restriction is checked, to see whether the current location varies from the previous ones as required (does not vary, slightly varies, etc.). If the variability is not the expected one, the pair is ignored and the next pair is tested. If the variability aligns with the required by the model, a final check is carried out, which is the sequence improvement check. The base idea behind the efficiency of dynamic programming is that the length of the best sequences obtained for each possible final pair are stored as the algorithm progresses, with the goal of not having to recalculate everything again. When a new pair is added to the current sequence being created, and if the pair satisfies both the frequency and location variability restrictions, the final check done by the algorithm is to ask whether adding this new pair would obtain a sequence that is better (longer) than the previous best sequence obtained, which was the one that ended in the current last pair in the sequence. This avoids adding elements that do not add anything to the timeline.

Thus, at the end of this process, we have defined timeline which includes all the dates and locations in which a crime must be simulated through a report by the police, which also follows the needed and realistic criminality patterns associated with a given crime category. This timeline is the base for the next report simulation component, which generates the reports for each of the instances in the timeline based on the date, location, and crime.

5.3.REPORT GENERATION COMPONENT

The main goal of the simulation module is to generate a series of reports that replicate real reports that the police force would have access to. Given that we have defined a timeline of realistic reports that could be filed, the last step to generating them is to properly define the structure of the report and what it should contain. Depending on the country, police report must contain certain pieces of information by law. In general, most places require the reports to have mention of the following: information regarding the incident (including date, location, type of incident and description), information regarding the people involved in the incident (victims, witnesses, etc.), descriptions regarding the possible damages/injuries or evidences left behind, and information regarding the actions taken by the police. For the framework to be easily adaptable to a real case scenario, the reports generated must have at least all these pieces of information. Thus, this last report simulation component is aimed at generating realistic reports with a set structure and a certain amount of information included considering a timeline of events given.

To carry out this task, the report generation component is mainly based on a language model. Although information regarding the date, location and type of crime is given as input from the previous components through a timeline, every other description previously mentioned which must be included in the report is not explicitly defined. To automatize the process of generating these reports, this component's architecture is based on a template-filling system based on prompts. The overall structure of the report is defined as a dynamic template, and for each of the crimes mentioned in the given timeline, the template must be filled accordingly, mimicking the real process of filling out police reports. To correctly fill out the information not explicitly given by the input, a high-performing language model is used. Prompts are given to this model which are also dynamically filled out to consider the new crime setting for each of the reports. Thus, the reports are filled out through the dynamic generation of information through prompts to a language model. The overall structure of this component can be seen in Figure 11.

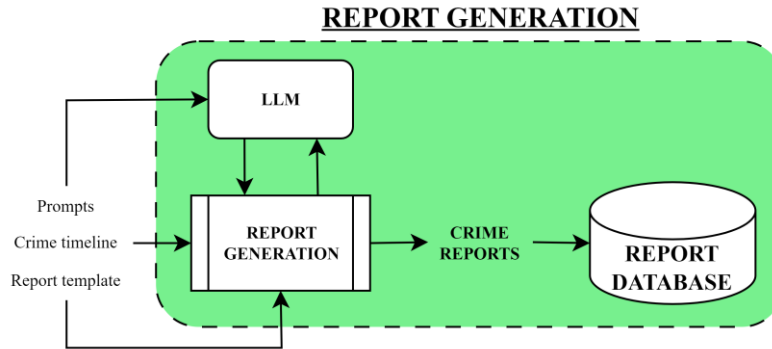


Figure 11. Structure of the report generation component within the report simulation module

Therefore, in this case, aside from the information input given from the previous component (crime timeline containing date, location, and crime type), two additional inputs must be defined, which include the report template to be filled out and the instruction prompts to the large language model to properly generate the information needed in the report.

Starting with the report template, this template is generated using a HTML approach for the final report to have a well-defined structure. As can be seen in Figure 12, the report is composed of four main sections, aside from an initial value indicating the date of the accident. The first of the sections is divided into two, 'Person(s) Involved' and 'Witness(es)', and includes information regarding the people involved, both the victim (or victims) and the possible witnesses. Furthermore, the second section ('Incident Details') is focused on the details regarding the crime, including information on the location of the incident (with street name rather than coordinates), the type of incident, and the specific details regarding the incident itself, as a small summary. The third section ('Damages and Injuries') contains information regarding the property damages and the injuries the victim might have sustained. Finally, the last section ('Actions Taken') includes a small insight on the actions taken by the police force when the incident was reported.

POLICE INCIDENT REPORT

Date of Report: 2023-06-08

PERSON(S) INVOLVED
Name: Samantha Johnson
Role: victim

WITNESS(es)
Were there any witnesses in the incident? Yes
Name: Sarah Thompson

INCIDENT DETAILS
Location: 16th Street, San Francisco
Type of incident: PETTY THEFT OF PROPERTY
Incident description:
 On June 8, 2023, at approximately 14:37 PST, a petty theft of property occurred on 16th Street in San Francisco. The victim was identified as Samantha Johnson. The item stolen was a Canon EOS Rebel T7i DSLR camera valued at \$550. The suspect is currently unknown and the investigation is ongoing.

DAMAGES AND INJURIES
Were there any injuries? Yes
Describe the injuries:
 Minor scrape on right elbow sustained while struggling with the suspect during the theft attempt
Were there any property damages? Yes
Describe the property damages:
 Canon EOS Rebel T7i DSLR camera lens scratched, tripod bent and broken

ACTIONS TAKEN
 Officers responded to the scene of the reported petty theft on 16th Street in San Francisco on June 8, 2023 at 14:37 PST. The victim was identified as Samantha Johnson. A report was filed and the incident number assigned. Evidence collection was initiated, including CCTV footage from nearby businesses. The investigation is ongoing to identify the suspect and recover the stolen Canon EOS Rebel T7i DSLR camera valued at \$550.

Figure 12. Example of generated report through the report simulation module

Out of all the sections included in the report, only three minor data points can be directly filled out by the input data, which include the date at the beginning on the report, the location and type of crime items within the 'Incident Details' section. All the remaining sections and section items must be made up by the language model. For the generated data to be precise and coherent, a different prompt is defined for each of the sections, specifying what the language model should generate. However, all the prompts must have three things in common. Firstly, they must include a minor introduction which contextualizes the setting, specifying that the generated text is aimed at filling out a section from a police report and the overall context of what the police report entails (the type of crime). Secondly, the prompt must specify what the data generated can and cannot include, mentioning also the length of the passage, as some sections are not expected to be as long as others. Finally, the prompts must specify that the result must be given in a format that can be easily inputted into the dynamic template. This format will either be a direct text, when the prompt is aimed at filling out a single item in an already filled out section (like in the case of the incident description), or a JSON object, when the prompt is aimed at filling out a whole section and each individual item within it. The prompts used for each of the report sections that could not be filled out by the input data can be seen in the following figures (Figure 13 to 18).

```
1 people_involved_prompt = "You are a professional police report generator"
2   "Write the data needed to fill out the 'Person(s) involved' section"
3   "in a police report."
4   "This section includes the following information: Name and Role."
5   "IMPORTANT:"
6   "- Output can only be EXACTLY ONE JSON with keys 'name' and 'role'"
7   "- The value of 'name' cannot be John Doe or Jane Doe,"
8   "it must be an actual made-up name."
9   "- To fill out the 'role', consider that the person involved is a"
10  "victim of the following crime: {crime}"
11  "- Do not include quotation marks or double quotes in the text"
12  "- Never output multiple options or multiple JSONs"
```

Figure 13. Prompt to generate the information in the 'Person(s) Involved' section of the synthetic report

```
1 witnesses_prompt = "You are a professional police report generator."
2   "Write the data needed to fill out the 'Witness(es)' section"
3   "in a police report."
4   "This section includes the following information: whether there were"
5   "witnesses or not (Yes or No), and if there were, their name."
6   "IMPORTANT:"
7   "- Output must be EXACTLY ONE JSON with keys 'witness' and 'name'"
8   "- The key 'witness' can only take values Yes or No"
9   "- You must randomly choose ONLY ONE value for 'witness'"
10  "- The key 'name' takes different values depending on the value"
11  "of 'witness'"
12  "- If 'witness' is Yes, then 'name' must take a realistic name"
13  "as value that cannot be John Doe or Jane Doe"
14  "- If 'witness' is No, then 'name' must take an empty string as value"
15  "- Do not include quotation marks or double quotes in the text"
16  "- Never output multiple options or multiple JSONs"
```

Figure 14. Prompt to generate the information in the 'Witness(es)' section of the synthetic report

```
1 incident_details_prompt = "You are a professional police report generator."
2   "You must write a concise, factual, realistic, and brief narrative"
3   "of something that could have happened regarding the following"
4   "crime: {crime}"
5   "IMPORTANT:"
6   "- Output can only be a JSON with key 'narrative'"
7   "- The crime occurred on the {crime_date} in {crime_location}"
8   "- The victim is called {people_involved.get('name')}"
9   "- You must mention real things, nothing to be filled out like"
10  "'Item 1', 'Location 1', etc."
11  "- Do not mention anything regarding an incident report number"
12  "- Do not include quotation marks or double quotes in the text"
13  "- The generated narrative must be the value of the key 'narrative'"
```

Figure 15. Prompt to generate the information in the 'Incident Details' section of the synthetic report

```

1 injury_prompt = "You are a professional police report generator."
2   "Write the data needed to fill out the 'Damages and injuries' section"
3   "in a police report."
4   "This section includes the following information: whether there were"
5   "injuries or not (Yes or No), and if there were, the injury description."
6   "IMPORTANT:"
7   "- Output can only be a JSON with keys 'injury' and 'description'"
8   "- The value of 'injury' can be either Yes or No"
9   "- You must randomly decide ONE value for 'injury'"
10  "- If 'injury' takes the value No, then 'description' must be"
11  "an empty string"
12  "- If 'injury' takes the value Yes, then you must invent a realistic"
13  "injury to be the value of 'description'"
14  "- If you have to invent an injury description (if 'injury' is Yes),"
15  "you must consider the following:"
16  "1. The crime committed against the victim was {crime_type}"
17  "2. The victim is called {people_involved.get('name')}}"
18  "3. The crime description includes the following:"
19  "{incident_details.get('narrative')}}"
20  "4. You must mention real things, nothing to be filled out like"
21  "'Item 1', 'Location 1', etc."
22  "- Do not include quotation marks or double quotes in the text"
23  "- Never output multiple options or multiple JSONs"

```

Figure 16. Prompt to generate the information regarding the injuries in the 'Damages and Injuries' section of the synthetic report

```

1 property_damages_prompt = "You are a professional police report generator."
2   "Write the data needed to fill out the 'Damages and injuries' section"
3   "in a police report."
4   "This section includes the following information: whether there were"
5   "property damages or not (Yes or No), and if they were the damage"
6   "description."
7   "IMPORTANT:"
8   "- Output can only be a JSON with keys 'damage' and 'description'"
9   "- The value of 'damage' can be either Yes or No"
10  "- You must randomly decide ONE value for 'damage'"
11  "- If 'damage' is No, then 'description' must be an empty string"
12  "- If 'damage' is Yes, then you must invent realistic property damage"
13  "to be the value of 'description'"
14  "- If you have to invent a property damage description (if 'damage'"
15  "is Yes), you must consider the following:"
16  "1. The crime committed against the victim was {crime}"
17  "2. The victim is called {people_involved.get('name')}}"
18  "3. The crime description includes the following:"
19  "{incident_details.get('narrative')}}"
20  "4. You must mention real things, nothing to be filled out like"
21  "'Item 1', 'Location 1', etc."
22  "- Do not include quotation marks or double quotes in the text"
23  "- Never output multiple options or multiple JSONs"

```

Figure 17. Prompt to generate the information regarding the damages in the 'Damages and Injuries' section of the synthetic report

```

1 actions_taken_prompt = "You are a professional police report generator."
2   "Write the data needed to fill out the 'Actions taken' section in"
3   "a police report."
4   "This section includes the following information: a description of"
5   "the actions taken by the police after a crime was reported."
6   "IMPORTANT:"
7   "- Output must be only a JSON with key 'actions'"
8   "- You must generate a short but realistic description of the actions"
9   "taken by the police after a crime."
10  "- For the creation of the description, you must consider the following:"
11  "1. The crime committed against the victim was {crime}"
12  "2. The victim is called {people_involved.get('name')}}"
13  "3. The crime description includes the following:"
14  "{incident_details.get('narrative')}}"
15  "4. You must mention real things, nothing to be filled out like"
16  "'Item 1', 'Location 1', etc."
17  "- The description generated must be the value to the key 'actions'"
18  "- Do not include quotation marks or double quotes in the text"
19  "- Never output multiple options or multiple JSONs"

```

Figure 18. Prompt to generate the information in the 'Actions Taken' section of the synthetic report

Based on these prompts, given a new crime in the timeline and the dynamic template, the language model receives the prompts filled out with the new data and generated the needed information. The template is filled and stored for future use. For this task to be correctly carried out, two external tools are used. Firstly, a trained language model is used. This model must be able to properly generate coherent text given a prompt. The main task of this model is not to combine a series of texts given and generate something new, as will be needed for future language models in different modules, but rather to invent short passages that align with the use case and with the rest of the report. Therefore, a Mistral language model is used, due to its trained ability at generating coherent text (as further explained in Table 3). Secondly, an established database is used to store all the generated reports for future use. As will be explained in Section 6, all the data sources will have to be stored in a vectorized way for a retrieval model to be able to better access them. Therefore, an embedding model is used to properly generate a numeric representation of the report's content that is then stored in a vector database. This embedding and storing process is part of a larger data integration process contained in the data contextualization and interpretation module, and thus will be explained in further detail on Section 6.1.2. The details regarding this database can be found in the Table 5 defined later. Thus, this final component of the module is deeply connected with other modules in the framework, as its output is a direct input for other processes.

Table 3. Existing tools used in the implementation of the report generation task

TOOLS	DESCRIPTION	RELEVANCE	USE
Mistral 7B model	Large language model (LLM) from the Mistral family. It has 7 billion parameters, optimized to require less resources and run faster than other heavier models.	• Can be run locally and free and it does not require many resources • Good at generating short texts based on a given prompt that do not require high depth or reasoning • Oriented at a single task it focuses on, like text generation	For the report generation task

6. DESIGN AND IMPLEMENTATION OF A DATA CONTEXTUALIZATION AND INTERPRETATION MODULE

The main goal of this project is to define a complete and unified system which enhances the behavioral analysis and anomaly detection system previously defined with additional modules to better contextualize and inquire in the behavior, and anomalies, displayed by a suspect in the context of a police investigation. To carry this out, a module has been designed aimed at analyzing and interpreting the conclusions reached by the defined system by leveraging a series of language models and contextualization tools.

For the comprehensive interpretation of the conclusions reached by the developed anomaly detection system, four data sources are considered. The correlation and integration of these sources allow for the results given to the user to be complete and context-aware. The data sources include both external information obtained by the police force regarding the investigation and the internal conclusions obtained from the anomaly detection system. The external information gathered by the police include the movement tracks of a certain suspect, defined by a series of spatiotemporal points (including coordinates and timestamps), and all the previous police reports regarding a certain suspect or crime that might be related to the investigation. On the other hand, the internal conclusions obtained from the anomaly detection system include the behavioral model built around a certain suspect, and the deviations (anomalies) detected from this behavior.

To properly consider, integrate, and correlate all the information sources, a Retrieval-Augmented Generation-based module has been developed, as seen in Figure 19. This module is aimed at contextualizing, saving, and integrating all the information in a unified data source which is then used to generate a complete report regarding the studied crime investigation.

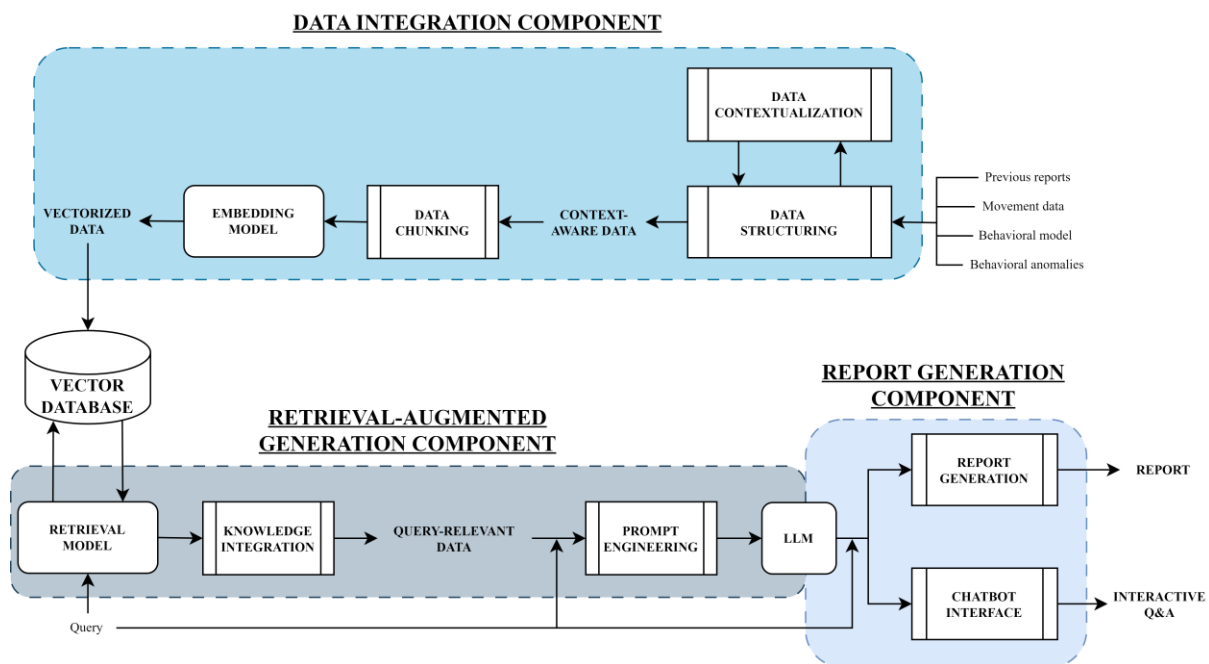


Figure 19. Data contextualization and interpretation module architecture

The architecture of this data contextualization and interpretation module is centered around a Retrieval-Augmented Generation (RAG) core which allows for the combination of retrieved external documents with language models to generate knowledge-augmented responses. This core is the main architecture that allows for the final generated report to consider the information from all the available data sources. Given a knowledge base, containing the data gathered, the RAG core retrieves the instances relevant to a specific investigation and generates a data-rich prompt. This prompt is the one

received by the language model that will generate the report given to the user. Thus, the language model uses both its pretrained knowledge, acquired from the training process, and the new retrieved knowledge to build the answer to the given prompt. Around this core, a series of additional processing components are integrated which tackle all the needed tasks for the module to be able to generate a coherent, structured, and complete report based on all the data gathered by the police and the anomaly detection system.

The first additional processing component is the data integration component. This first component is implemented as a pipeline of two main processing stages, which work sequentially to carry out the task of transforming the information gathered from all the sources into context-aware data instances, which are then normalized and saved into a single, unified database which will work as the knowledge base for the RAG core of the module. To do so, the pipeline first implements a data contextualization and normalization step, which enhances and adds support to the data gathered from all the mentioned sources. This is done through the implementation of a goal-based AI agent, which carries out relevant web searches to obtain further information and context regarding the existing data. The agent is also orchestrated with template and text processing tools to better combine the data with the newly obtained knowledge in a structured and formatted way. Once the data are enriched, the pipeline moves onto the second and final data vectorization and integration step. This last step carries out an embedding process, which vectorizes the enriched data before storing it into the mentioned unified database, which works as the knowledge base for the RAG-based core. Therefore, this first data integration component is active every time new data is gathered from any of the sources, even if the RAG core, or the second processing component are not triggered by the user, since its main goal is to properly store the context-rich data in the database for its future use by the other processes contained in the module.

The second processing component is the report generation component. This second component allows for the police force to have access to the updated status of the investigation and to identify all the advancements of interest showcased on the data. This is done through two channels. Firstly, the report generation component creates a complete and structured document containing all the relevant information gathered from the retrieval model implemented in the RAG core. Secondly, the report generation component creates a chatbot interface for the user to access to ask specific questions regarding the data available. Therefore, for both channels, this component works closely with the RAG core, as it directly uses the language model built into the core to carry out the needed tasks to present the final findings in a structured, coherent, and useful way. Thus, this second component, like the RAG core, is active every time the user requires information on a case and sends out a prompt to the system.

Based on this initial overview, we will revise the complete design and technical implementation of all the mentioned components and their integration into their respective processes to define the complete data contextualization and interpretation module.

6.1.DATA INTEGRATION COMPONENT

The first processing component from the data contextualization and interpretation module takes as input all the data given by the available sources and returns as output a contextualized and vectorized version of said data, which is saved in an established vector database for future use. The main goal is to properly preprocess and structure the data to make it useful and easily retrievable by future components of the module. To carry this out, the data undergoes a series of processes including the contextualization of the raw data instances, the structuring of the information, and the semantically-rich vectorization of the transformed instances. All these processes are built in like steps on a pipeline defined by two main steps: the data contextualization and normalization step, and the data vectorization and integration step. An overview of the pipeline's structure can be seen in Figure 20.

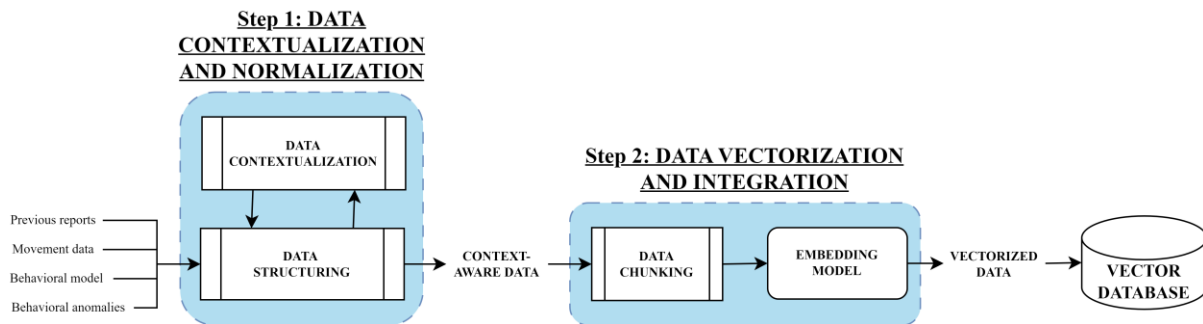


Figure 20. Structure of the data integration component within the data contextualization and interpretation module

6.1.1.1. DATA CONTEXTUALIZATION AND NORMALIZATION STEP

Although all the available data sources are useful and give different perspectives and contexts to the suspect or crime under investigation, when analyzed separately, they are limited in the amount of information they provide. Since the goal of this process is to provide the system with a fully contextualized and complete database that enhances all the possible sources, one of the first limitations to be tackled is the context-sensitive nature of the data. Furthermore, due to the heterogenous nature of the data sources, each one of them must be tackled according to their format and how the information is provided.

This first step is aimed at properly contextualizing all the available information by considering the crime investigation setting, and structuring the data, by emphasizing the instances of interest. To do so, two interrelated tasks are carried out alternatively with the goal of transforming the data, with different formats, into a series of documents which explain in detail the conclusions reached by each of the sources, their relationship with the other sources, and their relevance outside of the information available to the police.

The data structuring task is the one that takes as input all the data from the established sources. As previously mentioned, this data includes the existing reports available to the police forces, the movement tracks of suspects followed by the police, the behavioral models built for each suspect, and the deviations detected on the behavior of the suspects. Aside from the reports, the remaining information is usually given in a raw and non-intuitive way, making it non-understandable for humans. The main goal of the data structuring task is to restructure the information in a way that is easily comprehensible by the user, through the generation of written documents containing detailed explanations of what the raw data is conveying as well as its relevance in the broader context of the police investigation. Since the anomaly detection system was developed under the same line of work, the data instances extracted from it regarding the behavioral model and the anomalies detected have a known and well-defined structure. Furthermore, the movement traces gathered by police forces that this system takes as input also have a defined, required structure. Therefore, for all the data that contains non-intuitive information, an automatic systematic extraction of the information can be done through template and text processing tools.

For each of the data sources, a dynamic template is defined with the template engine developed by Jinja2. These templates contain dynamic parameters, which are filled out by injecting the contents of the data source directly onto the template. This allows to generate text-based documents with the information gathered from the data sources. Thus, a specific layout and structure of the information can be easily defined, including explanations on how each data instance should be interpreted based on the knowledge behind the structure of the data sources. For each of the sources, the structuring task can be defined by its original content and structure and the expected output generated:

- Movement tracks data of a specific suspect (gathered by the police): the original source only includes a list of different coordinates (latitude and longitude) and an associated timestamp. The new structured document is expected to include information on the names of the streets visited by the suspect, as well as the relevance of those streets in the crime landscape of the city they belong to.

- Behavioral model of a specific suspect (built by the anomaly detection system): the original source includes a wide range of information regarding how a suspect behaves throughout the day, but the most interesting insights include the coordinates of each of the places flagged as significant for the suspect, the model of length of stay at each place (including the average time and average deviation), the model of speed of transition between places (including the average speed and average deviation), the probabilistic model of transitions between places, and the coordinates of the paths taken to move between places (including the chances associated and the actual optimal path between places). The new structured document is expected to include a detailed explanation of what each of the numeric data instances refer to (stay time, speed, chances, etc.), an association of each of the places and paths to real buildings (residences, hospitals, workplaces, etc.) and real streets, and a contextualization of the given locations in their relevance to the crime landscape of the city they belong to.
- Behavioral deviations/anomalies of a specific suspect (detected by the anomaly detection system): the original source includes the same information as the movement tracks data (coordinates and timestamp) with additional labels with specify the anomalies detected in each of the coordinates and the anomaly degree associated with them. The new structured document is expected to contain information on what the anomalies are (what each label represents), where they were found (by associating the coordinates to the places in the behavioral model), and their significance level based on their anomaly degree.
- Existing police reports (developed by the police): the original source includes previous reports fully written by the police, either on a specific suspect, crime, or investigation. The reports are expected to be fully context-aware, structured, and complete. Nevertheless, an initial analysis is carried out to ensure that no numeric/non-textual data is included in the document without an associated coherent explication of how the data should be interpreted.

Nevertheless, although the knowledge behind the structure of the data gives an important insight to what the data is conveying in the context of the investigation, as previously mentioned, the data sources are limited to the information gathered by the police and the conclusions reached by the anomaly detection system. Therefore, nuances regarding why the data is relevant in the broader context of the criminal landscape of a city, considering knowledge not included in the data sources, are not accounted for. This mainly includes the background and historical context of places visited by a suspect, which might give an insight on their relevance from the perspective of crime analysis (places like crime hotspots, high-risk targets, etc.). Considering these nuances, the data can be analyzed taking into account the social and environmental circumstances behind the data (gathered from external contextualization), as well as its connection to the suspect and the investigation (gathered from police forces) to give a full picture.

To tackle the task of contextualizing the data instances, the data contextualization task is developed. As previously mentioned, the main goal of this task is to relate the data instances to outside knowledge not included within the original data, which includes context relating it to actual places, events or circumstances surrounding it, or background and prior history related to it. Therefore, to properly contextualize this information, external sources are used. Due to the wide-ranging and diverse external sources available, an AI agent architecture has been designed to tackle this task, reflected on Figure 21.

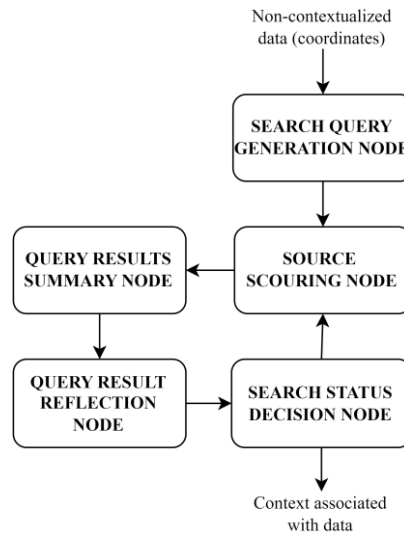


Figure 21. Contextualizer AI agent node-based architecture

The agent architecture is aimed at self-reflecting on the sources found and the information gathered from them to automate the process of gathering insight from external sources. To do so, the architecture is divided into three main processes which include the process of scouring the available sources and extracting a series of results, the process of summarizing those results into a single, non-repetitive and coherent result, and the process of reflecting on the obtained results to determine whether more information is needed. These processes, with a series of less complex steps surrounding them, are built into the non-linear execution flow seen in the above figure. Thus, the mentioned processes are defined as nodes in a graph, each with their own expected inputs and outputs.

Moreover, for the agent to properly work with the data and results it obtains over time, states are defined within the flow. These states are a way to keep track of the lifetime of the agent carrying out a specific task, including the values it takes and the outputs it generates. It does so by using attributes (variables) that persist between nodes in the graph. For our specific use case, three states are considered, which can be defined as follows:

- *Input State*: This state keeps track of the topics the user wants context on. It is the only state that the user directly changes, and the first one to trigger the agent to start carrying out its task. It only has one attribute, which is the topic given by the user. This topic includes certain data instances (coordinates) and instructions on what information we want based on them, which in our case include the historic criminal landscape surrounding the given place.
- *Output State*: This state keeps track of the results (context) obtained by the agent on the topic given by the user. It is the final state, and thus also has only one attribute, which is the final context built through the combination of various sources. To better track the data sources, and later be able to validate their veracity, this state returns both the information gathered, as well as the original sources it was extracted from.
- *Agent States*: This state keeps track of all the outputs obtained in the lifetime of the agent, meaning in all the nodes used while carrying out the contextualizing task. Thus, this state includes a total of six attributes, which save the status of each of the processes carried out. This includes attributes relating to the following: the current topic being searched, the current search query generated, all the results obtained from the searches based on the current query, the sources associated to each of the results saved, the number of times a source scouring process has taken place (considered in the search status decision node), and the current final summary of all the results and sources to be given to the user.

Considering the states of the graph, the nodes can be easily defined, as most of them take all the agent states as input to carry out their task based on the status of the overall process, and update those states as output.

To define these nodes, a series of external tools and frameworks have been used. Due to the changing nature of the large language model scene in the field of Artificial Intelligence, many different tools can be used for similar tasks. Therefore, a deep-dive was carried out (refer to Section 2 for more detail) to select the most suitable tools for our goal. In the case of the development of this AI agent architecture, the used tools are defined in Table 4. The specific use of each of them is later specified when the nodes are defined, as most of the tools serve a specific purpose in the agent's execution pipeline.

Table 4. Existing tools used in the development of the contextualizer AI agent

TOOLS	DESCRIPTION	RELEVANCE	USE
LangGraph	Framework aimed at building graph-based workflows. It is often implemented for the development of agent-based and LLM-based applications.	- Allows for graph-based flow (easy control of agent logic and conditions) - Facilitates multi-agent definition and collaboration - Easily integrated with open-source tools like LangChain and certain LLMs	To define the AI agent architecture
Qwen2.5-14B model	Large language model (LLM) from the Qwen family. It has 14 billion parameters (high reasoning and language comprehension), and is compatible with tools like LangChain and LangGraph.	- Can be run locally and free without requiring excessive resources, unlike other option which are either not free of use (GPT-4, Claude 3) or take up too many resources (Yi-34B) - Good reasoning and chatbot abilities, with capabilities as good as alternatives which take up more RAM (Mixtral-8x7B, Deepseek-Coder-33B, etc.) - Suitable for all the tasks that require a language model in our agent, unlike simpler alternatives which often only cover few tasks efficiently (Mistral-7B-Instruct, Gemma-7B / 2B, etc.)	For the tasks in the search query generation node, the query result summary node, and the query result reflection node.
ChatOllama	LangChain LLM wrapper class which allows to run local, open-source and free language models. It allows to access the models and chat with them without external data being sent to any other servers.	- Allows to develop local and private agent-based systems - Works easily with different language models - Allows full control over prompt engineering, useful for implementations with LangGraph	For the tasks in the search query generation node, the query result summary node, and the query result reflection node.
SearXNG	Private and self-hosted metasearch engine designed to tackle the limitations of the	Returns complete and neutral responses through the aggregation of the results of multiple search engines	For the tasks in the source scouring node.

	original SearX engine, making it faster and better maintained.	• Open-source and can be self hosted • Has privacy settings, as it does not share any information regarding the user, avoiding IP blocking	
--	--	---	--

Considering these tools, the nodes are defined as follows. Starting with the first node, this is the search query generation node. This node is aimed at building a complete query for the following source scouring. The query is built based on a series of defined instructions. These instructions mainly include the main goal of the node (to build a complete and coherent search query), the topic on interest the query should be built around (defined by the user), and the structure in which the query should be return, which in our case is a JSON structure containing information on the actual query, the relevance of the query to the search, and the topic being explored. All the mentioned are included in the output to better trace back the thought process of the agent, which, given these instructions, generates the expected output. This is done through the previously explained ChatOllama LLM wrapper. These wrappers allow to better interact with LLMs, especially allowing for the usage of a prompt system instead of raw API use. Both the defined instructions (which will act as a system prompt) as well as a trigger for the model (which will act as a human prompt, simply stating a query should be generated) are sent to the previously specified LLM (Qwen2.5-14B) through this wrapper. The response from this LLM is the query used for the following nodes. Therefore, this first node takes as input the topic given by the user, and returns a complete search query for the next node. It also updates the search query attribute from the agent state. The mentioned instructions used for this first node can be seen in Figure 22.

```

1 web_query_prompt = "Your goal is to generate a targeted web search query."
2 "The query will gather information related to this specific topic:"
3 "{topic_to_search}"
4 "Return your query as a JSON object with the key 'query'"

```

Figure 22. Instructions to indicate the AI agent how to generate the web search query

The second node in the agent's graph-based architecture is the source scouring node. This is the most complex out of all of them, as its goal is to find all the information based on the given query from the available data sources. One of the most used data sources for agent-based architectures are web-based sources, as they often have up-to-date information and allow for a wider search of information than limited data bases. For this agent, we will follow the approach of developing a web-based search, as it also tackles the limitations present in crime-based databases, which are often not available or do not contain movement information, which is necessary for our contextualization task. Therefore, the agent is developed to make use of public and free web APIs which allow for searched through prompts. However, the nature of our task brings front two additional limitations, as we expect to automatically contextualize a multitude of data instances (coordinates) with crime data. This causes two problems. First, free-use APIs often block IP addresses that send out too many automated queries. Although establishing time between queries is an option, it not always works efficiently in avoiding being flagged, and often makes the whole process time-costly and less efficient. Second, free-use APIs often control and censor results, especially when associated with sensitive content. Given the goal of contextualizing with criminal records, the task is often not possible due to the nature of the search, as many APIs do not return information.

Thus, to tackle all the presented issues, as well as to develop a more diverse and efficient searching engine, we make use of the open-source metasearch engine SearXNG previously discussed on Table 3. By deploying a self-hosting engine with free-use web-based APIs, the agent can carry out web searches without external data collection and IP blocking, as the deployed engine allows for the implementation of request distribution and the aggregation of multiple engines for more complete, diverse, and neutral results. The architecture of the deployed engine can be seen in Figure 23.

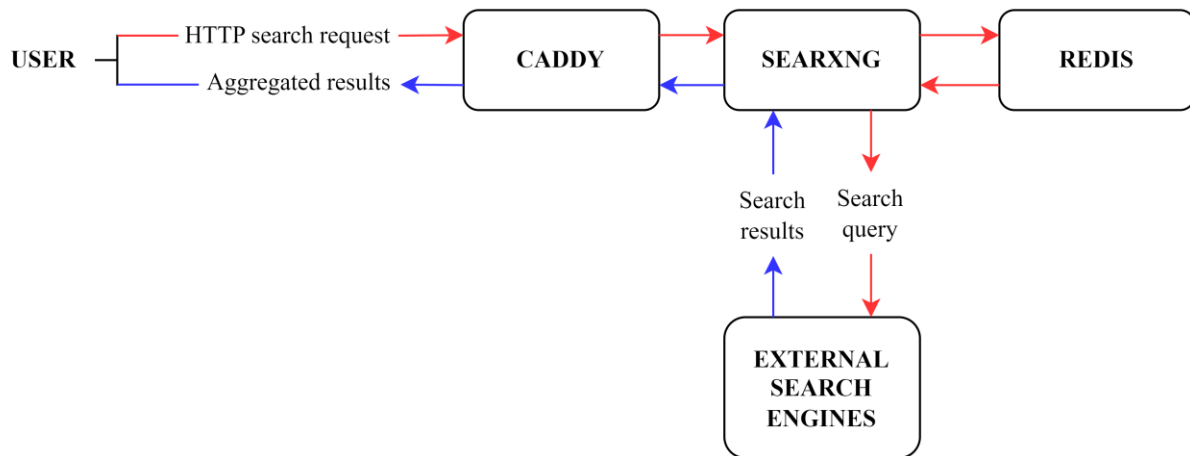


Figure 23. Architecture of the metasearch engine implemented in the AI agent

The workflow of our search engine goes as follows. The search query defined in the search query generation node is first sent to a HTTPS proxy. In the SearXNG implementation, the proxy used is Caddy, which is easy to use due to its simple and automatic configuration, which handles certificates in HTTPS communications automatically. This Caddy proxy takes the search query sent to the architecture and forwards the message to the SearXNG engine as a HTTPS request, acting as a reverse proxy. Once the request enters the engine, the first task the engine carries out is checking whether there is an existing query results queried in the architectures database Redis (Remote Dictionary Server), which is a key-value database that saves the query and the respective response. This first step is carried out to reduce processing time in case a same query is searched multiple times to optimize the process. If the query response is cached, the saved response is directly sent back to the user through the Caddy proxy. If the query response is not cached, SearXNG sends the query to a series of predefined search engines. In our case, as previously mentioned, those search engines will be free-use APIs with no search limit, which in this implementation includes the DuckDuckGo API (direct web search), the Mojeek API (Google alternative, for more diverse responses), and the Wikipedia API (for encyclopaedic-based responses). Although in most cases all the APIs return similar responses, three are used to ensure neutrality and diversity in the results. Thus, each of the mentioned engines will return a response based on the query sent. All the responses are aggregated and ranked before being sent back to the user through the Caddy proxy.

Based on this SearXNG architecture, the source scouring node takes as input the current states of the agent as well as the previously generated search query. It runs the search engine with the given query, and processes the results returned as to extract the sources from the actual content, since the search results are returned in JSON format. The content and its respective sources are saved onto the respective agent state attributes before the agent moves on to the next node.

The third node in the graph's workflow is the query results summary node. Since the agent is based on a non-linear pipeline, which repeats certain processes based on the results obtained to reach the best possible result, the current results obtained by the search might be the first ones, or they might be an improvement of the past ones. In the latter case, to not lose the information gathered previously, instead of dismissing previous results and only considering the most recent response, a summary is made of all the responses obtained, both previously and currently. This helps the agent carry out the task of filling out knowledge gaps efficiently, as, instead of generating brand new results with new gaps, the ones detected previously are filled out by joining both outputs into one complete summary. This summary is generated using the same LLM previously defined (Qwen2.5-14B) and using a similar approach as the one used in the search query generation node. A set of instructions is developed, which indicate to the model how to build a summary. Thus, the instructions include guidelines on what the goal is (to generate a coherent and complete summary), on what to include in the summary (information not yet present, emphasis on important points, maintain consistency, etc.), and on what not to include in the summary (repetition on concepts, certain phrases, etc.). Instructions are defined for both when new results are combined with old ones, as well as when the node is dealing with the first results obtained from the first agent's iteration. Based on these instructions, as well as

the results obtained from the search engine, like with the first node, two prompts are sent to the LLM. These include a system message containing the instructions, as well as a trigger message (human message) containing the trigger phrase to start the process of generating the summary. Like with the instructions, the human message will have two different versions, depending on whether the summary generated is a combination of various results, or just a summary of a unique source. Moreover, the attributes of the agent states, like in the nodes before, are updated accordingly (mainly the attribute storing the current summary). The mentioned instructions on how the summary should be carried out can be seen in Figure 24.

```

1 result_summary_prompt="Your goal is to generate a high-quality summary"
2 "of the web search results."
3 "If a summary already exists, you must extend the results."
4 "To EXTEND the results, you must:"
5     "1. Integrate new information without repeating what's already"
6     "covered"
7     "2. Maintain consistency in style and depth"
8     "3. Only add new, non-redundant information"
9     "4. Do not specify what content is new and old, carry out a smooth"
10    "transition"
11 "If a summary does not exist, you must generate it."
12 "To CREATE the summary, you must:"
13     "1. Highlight the most relevant information from each source"
14     "2. Provide a concise overview of the key points related to the"
15     "report topic"
16     "3. Emphasize the significant findings"
17     "4. Ensure a coherent flow of information"
18 "In BOTH cases:"
19 "- Focus on factual, objective information"
20 "- Avoid redundancy and repetition"
21 "- DO NOT use phrases like 'based on the new results' or 'according to'"
22     "additional sources'"
23 "- DO NOT add a title or a preamble like 'Here is an extended summary',"
24     "just directly output the summary"
25 "- DO NOT explain the tools used to extract the results, only the"
26     "conclusions"
27 "- Do not use XML tags in the output."

```

Figure 24. Instructions to indicate the AI agent how to summarize the search results

The fourth node is the query results reflection node. This node plays an important part in the agent's only loop, as its main goal is to determine if there are any knowledge gaps in the current final summary. A knowledge gap for the agent is understood as an unexplained instance in the final summary which might be important for the understanding of the summary, or to deliver a better and more contextualized picture of the topic. It stands as a way of determining, based on the original query and the current summarized result, if there is an additional question, stemming from the original one, that has been left unanswered by the result. Thus, to detect these gaps, an instruction-based approach is implemented again. A set of instructions are defined which include guidelines on what the LLM should do. This basically includes identifying knowledge gaps in the current results, and generating a follow-up question to expand the understanding on the topic and to cover ideas not fully covered in the original results. These instructions, with a defined trigger message, are given to the LLM, which generates a follow-up query. The instructions given to the agent to carry out this reflection can be seen in Figure 25.

```

1 result_reflection_prompt = "You are an expert research assistant"
2 "analyzing a summary about {topic_to_search}."
3 "Your tasks are the following:"
4     "1. Identify knowledge gaps or areas that need deeper exploration"
5     "2. Generate a follow-up question that would help expand the"
6     "understanding"
7     "3. Focus on technical details, or emerging trends that weren't"
8     "fully covered"
9 "Ensure the follow-up question is self-contained and includes necessary"
10 "context for web search."
11 "Return your analysis as a JSON object with two keys: 'knowledge_gap'"
12 "and 'follow_up_query'"

```

Figure 25. Instructions to indicate the AI agent how to reflect on the summarized results

The fifth and final node is the search status decision node. The main goal of this final node is to decide whether the search should continue to fill out the detected knowledge gaps in the previous node, or the current summary should be delivered to the user. This node works under one limitation, which is the existence of a maximum number of loops specified by the user. This number is set to avoid infinite loops, where the previous node finds small gaps infinitely as gaps might be created back and forth depending on what the responses of the search are (filling out one gap might create another). Thus, if the maximum number of loops has not been reached, then the node goes back to the source scouring node with the new updated query defined by the query results reflection node and carries out the loop again. On the other hand, if the maximum number of loops has been reached, then the current summary is returned as the final summary.

This agent architecture is continuously called by the data structuring task, as it allows to fill out the missing information when defining the context-aware, complete textual descriptions of what the data entails. Therefore, both tasks work alternately to extract the significant data instances from the sources available, to properly structure them, and to add all the needed additional contextual information for the final document associated with the data to be coherent and human-understandable, containing relevant and useful information.

6.1.2. DATA VECTORIZATION AND INTEGRATION STEP

Due to the nature of RAG architectures, although the textual data it handles can come in multiple different formats (documents, JSON, CSV, etc.), the way that data is stored plays an important role in the process of effectively retrieving and making use of said data to answer a specific query. RAG architectures usually work with vector-based databases, as the similarities between data instances can be easily computed based on their vectorial representation and the distances between said vectors. Since the core component of the data contextualization and interpretation module is based on a RAG architecture, the data extracted from the first step of this data integration component must be properly stored for future use.

This second step in the pipeline is aimed at storing the defined and context-aware data instances in an understandable way for future components. To carry this out, the second step is composed of two tasks which work sequentially to transform the data into vectorized representations, and to store it into an accessible and easy integrable database. These tasks are the data chunking task, and the data embedding task.

The first of the two tasks, the data chunking task, has the goal of reducing the large textual data instances into smaller segments that can be easily embedded. Chunking allows for the vectors to be more significant regarding their content, as embeddings of overly large texts tend to be generic, making them harder to retrieve in future processes. Thus, chunking is implemented to divide the text into smaller but semantically coherent fragments for more accurate retrieval, and for the implemented models to work with prompts with sizes that stay within most token limits. To do so, in this first task, the documents are divided into chunks of an established size (defined by the user) with a certain amount of overlap, as to not lose semantic context in the chunks. So, several textual-based data

instances are given to this first task as input, and the output is a series of chunks of the input, with semantic coherence but smaller in size.

The second task in this step is the data embedding task, which takes as input the chunks generated in the previous data chunking task. The goal of this task, as mentioned before, is to transform the given chunks into a numeric representation that can be used to determine the similarity between texts. This will be used to properly retrieve a series of chunks based on their similarity (relevance) to a given topic. To carry out this task, a series of external tools are used, which can be defined as follows in Table 5.

Table 5. Existing tools used in the implementation of the data embedding and storage task

TOOLS	DESCRIPTION	RELEVANCE	USE
Qdrant database	Vector database (“Quantum Data Retrieval” database) that stores high dimensional embeddings and supports multiple similarity search approaches and metadata filtering.	- Is easily deployed on a local computer, self-hostable and scalable (easier to deploy and scale than other existing alternatives) - Allows for fast vector search and real-time retrieval tasks - Easily integrable with tools like LangChain and RAG architectures	To store data embeddings (RAG core knowledge base).
all-MiniLM-L6-v2 model	Text embedding model from the Sentence Transformer family. It has 22.7 million parameters, and considers sentence semantics when creating vectors.	- High and fast performance with compact size, easy to run locally - Effective for similarity retrieval-based models, since it considers sentence structure and is trained based on sentence similarity and information retrieval - Integrated into and supported by many existing tools like LangChain and LangGraph	To embed the data after text chunking task.

Given the chunks generated in the task before, the data embedding task takes these chunks and runs the embedding model previously explained (all-MiniLM-L6-v2). For each of the chunks, a numeric representation is generated with this model. Many existing models are limited by their training, as they consider individual words rather than phrases, creating generic vectors which often reduce the accuracy of retrieval models which compare multiple vectors to determine similarity. Furthermore, many models have a hard time integrating into larger systems (like RAG architectures) due to their slow, and often complex, nature of processing text into embeddings. Therefore, a lighter, faster, and high-performing model has been selected for this task. The model tackles both the problem of transforming the data into vectors by considering the whole chunks when computing the numeric representation, allowing for the final vector to consider all the content and context contained in the text chunk. The model also tackles the future issue (which will appear in the next RAG core component of the module) of being able to efficiently compare and retrieve vectors with similar semantics, as the model allows for multiple similarity measures and efficient comparison. Thus, the embedding model takes as input the chunks defined in the previous task and generates as many vectors as chunks given, which are then stored in a vector-based database that can be easily accessed by future components.

The vector database in this architecture is a Qdrant database. Due to the nature of that data, which contains a high number of instances which increases over time as more information is gathered regarding a crime or suspect, the vector database was selected based on deployment and scaling capabilities, and well as its integration with future components of the module. The database deployed in the local server for this task contains as many collections as suspects stored, with the goal of keeping the database organized and allowing the RAG component to better retrieve relevant documents. These collections work as vector-optimized tables, which contain both the vectors generated, as well as the metadata associated with them. When the database is deployed and each of the collections is created, specifications of what the metadata should contain, as well as the vector sizes, needs to be specified. In our case, the metadata mainly refers to the suspect ID and the type of information contained in the vector (movement track information, behavior information, etc.). Thus, each time new vectors are created, the database is scoured to check whether information on a certain suspect is already present. If information is not present, a new collection is created and the vectors are stored. If information is present, the respective collection is expanded with the new vectors created.

All in all, the final output of this step, as well as the data integration component, is a vector-based database containing all the available information to date on a specific suspect which considers both the data gathered from internal sources (police investigation and data processing systems) as well as external sources (historic context and knowledge from web sources). As previously mentioned, this whole component, although directly connected to the next two, works independently, as it can process information and carry out its respective tasks every time new data is obtained, even if the user does not trigger the generation of a report or the chatbot interface.

6.2.RETRIEVAL-AUGMENTED GENERATION COMPONENT

The second processing component from the data contextualization and interpretation module works as the core architecture that holds together all the components in the module. The main goal of the module is to generate reports (and a chatbot interface) based on a current police investigation considering all the information gathered and available. The main limitation to tackle to carry this out, as previous explained, is the lack of information pretrained language models have on new and specific data. LLM carry out tasks only considering the knowledge they have from the data they have been trained with, which usually encompasses the structuring of sentences and texts, and general knowledge on wide-spread and known topics. Thus, while a LLM might be able to answer questions regarding usual behavior by criminals in a city, or where certain crime spots might be found, it does not have information on specific suspects followed by the police or newly committed crimes. For the LLM to be able to respond considering said information, two possible alternatives can be carried out. On one hand, the model can be trained again, which is not an efficient solution, as it requires a lot of resources and a constant pretraining to keep the model up to date. On the other hand, a Retrieval-Augmented Generation architecture can be built around the LLM.

A Retrieval-Augmented Generation (RAG) architecture enhances a language model's ability to respond to queries by providing access to additional knowledge. Based on a query given by a user, the RAG architecture extracts significant information from a defined knowledge base that is closely related to said query. Then, both the query, as well as the retrieved new information are processed to generate a new and more complete prompt (prompt engineering). Thus, the actual prompt received by the LLM is not the original query given by the user, but a new prompt containing all the necessary and updated information from the knowledge base regarding the query's topic. This allows the LLM to answer a query using both the knowledge acquired in its training process, as well as the new knowledge given through the enhanced prompt. The RAG architecture defined for this second component of the module can be seen in Figure 26.

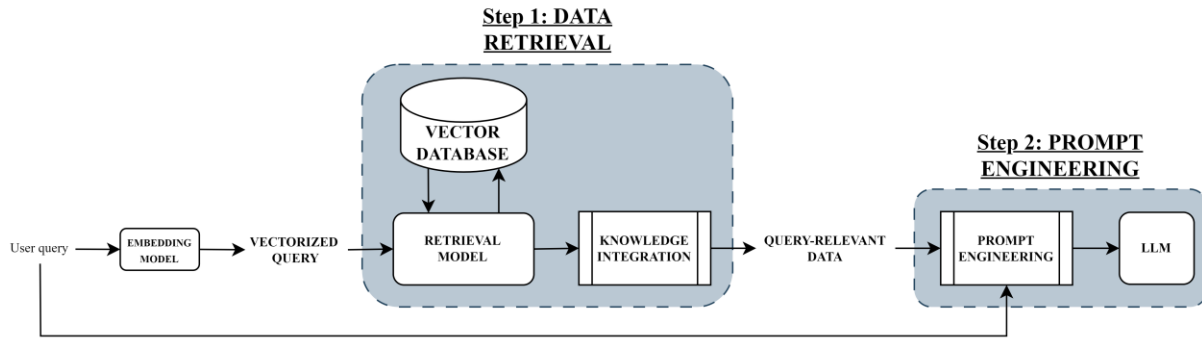


Figure 26. Structure of the retrieval-augmented generation component within the data contextualization and interpretation module

In the case of the contextualization and interpretation module, and in general in most LLM-based interpretations, continuously retraining the model to consider and include newly obtained information is not only ineffective, due to the time consumed, but also often not possible, as constantly training a large language model requires a high number of resources. Therefore, as previously mentioned, a RAG-based approach yields better results, as it allows to benefit from LLM which have been training with immense amounts of data while also allowing the model to consider up-to-date and relevant information in its responses. For an architecture like the module defined, it is necessary for the answers to consider not only general knowledge, but all the information gathered by the police through different sources to give a current and complete status of the investigation, both in the report generated, as well as in the chatbot interface. Therefore, a RAG-based solution is the core architecture of this module, which enables our model to take advantage of all our data processing tools to respond to user queries.

In this RAG component, the process carried, as seen in the above figure, is composed of two main steps, which work sequentially. These steps can be defined as the data retrieval step, and the prompt engineering step.

6.2.1. DATA RETRIEVAL STEP

As mentioned previously, the purpose of a RAG architecture is to fill out missing gaps in the LLM's knowledge to properly answer queries. These gaps are filled with the data contained in a defined database, developed for this architecture, which is expected to contain the necessary information to be considered by the LLM. This knowledge base is defined and updated through the process explained in Section 6.1, in the data integration component, in which as new information is gathered by the police force, it is immediately processes and contextualized to be storage in the database defined. Thus, this knowledge base is expected to contain a series of numerical representations (vectors) of all the data gathered by the police so far.

Based on this knowledge base, the goal of the first step of the RAG core is to properly extract all the available information that is relevant to a given prompt. To do so, two main tasks work sequentially to efficiently determine which data instances could be determined as useful or related to the prompt, and to integrate the knowledge in a coherent and concise way to be able to generate a new prompt which correctly considers all the information. These two tasks can be defined as the data retrieval task, and the knowledge integration task.

The first data retrieval task is aimed at retrieving from the knowledge base the most similar and relevant documents relating to a query. To carry out this task, some of the previously used tools for the data vectorization and integration step of the data integration component are used (Section 6.1.2.). More specifically, both the embedding model used (all-MiniLM-L6-v2) and the database configuration (Qdrant database) play a role in how information can be properly retrieved given a specific prompt. The task carries out as follows. Firstly, given a query by the user, which will be either the instruction to generate a report on a given topic or a question given to the chatbot interface, this query is properly embedded, with the goal of being able to compare it to the existing vectors in the database. Since the vectors have been created using the model explained in Table 5, the same model must be used in this case to ensure proper comparison. Secondly, the newly created vector is

compared to all the data vectors in the database. Depending on the specifications given (whether information is wanted on a single suspect or is a general query), the comparison will be carried out with data from specific collections or all the database.

A Qdrant base can be configured to be able to carry out multiple types of comparisons between vectors with different similarity measures. Given a maximum number of documents k to be retrieved (specified by the user), the database finds that number of nearest neighbors to the newly generated query vector using any of the available similarity measures. In the case of our database configuration, the cosine similarity measure is implemented. This is a standard measure which considers the angle between vectors to determine their similarity in direction which relates directly to their similarity in meaning. This is computed using the cosine formula (Eq. 11), where V_n represent the vectors being compared. The cosine is a useful metric since it yields a value of 1 when the angle between vectors is 0° (their meaning is the same), and a value of 0 when the angle between vectors is 90° (they are perpendicular, no relationship in the meaning). Furthermore, in the extreme case that the vectors have complete opposite meanings that contradict each other (angle of 180°), the cosine similarity takes a value of -1. Thus, this metric takes on a range of values between 1 and -1 which indicate the relationship in meaning between vectors, ranging from equal meaning to complete opposite meaning.

$$sim_{cos}(V_1, V_2) = \frac{V_1 \cdot V_2}{\|V_1\| \|V_2\|} \quad (11)$$

Although the tools used (like the embedding model or the vector database) are aimed at optimizing the cosine similarity measure by default, the reasoning behind choosing this metric over the alternatives like the Euclidean distance or the dot product lies on the nature of the comparison we wish to make for this data retrieval. The cosine similarity metric considers the direction of the vectors when comparing, rather than their magnitude. Although vector magnitude can be useful in many cases to determine similarity, in our case, it is expected for the query given by the user to be significantly smaller than the vectors it is being compared to. Therefore, if magnitude is considered to compare similarity, the accuracy would significantly decrease, as the magnitude would often greatly differ, returning low similarity values, even if the meaning is the same. Thus, considering this, the cosine similarity returns the best results, as it only considers the meaning of the vectors, regardless of whether the query vector is smaller than the text vector.

Given this cosine similarity metric, and given a query by the user, the query is vectorized and this metric is computed between it and all the relevant vectors from the vector database. The database returns the k documents (vectors) with the highest similarity score, which are the documents with the closest or most relevant meaning to the query. These will be the documents passed to the next knowledge integration task.

The second task in this step, the knowledge integration task is aimed at correctly combining all the information gathered from the previous retrieval task into a more concise and coherent result, which will be needed for the prompt engineering step. Although all the retrieved vectors are relevant, since we have divided the original data sources into chunks for storage (Section 6.1.2), many of them are parts of bigger documents which by themselves might lack additional context. Furthermore, many of the vectors might have overlapping information. Thus, this task is aimed at combining all the retrieved document chunks into one single result that is context-rich and coherent, in which information is not repeated. The result will need to be structured and concise enough for the prompt engineering step to be able to properly work with it, but descriptive enough for the LLM to be able to use the information given to answer queries. To carry out this task, a language model is used which takes as input all the retrieved document chunks, as well as the given query, and generates the explained combined result. This LLM model is the same model used in the next report generation component, which was specifically selected due to its volatility to cover multiple types of tasks, from text summarization to text generation. Like with previously used LLMs, a tool to properly execute the LLM model will be used to better communicate with the model selected. The external model and tools used for this task is described in Table 6.

Table 6. Existing tools used in the implementation of the knowledge integration task

TOOLS	DESCRIPTION	RELEVANCE	USE
Llama3.2 model	Family of large language models (LLM) from the Llama family developed by Meta. It has been trained to have high reasoning capabilities and allows for the use of up to 180.000 tokens of context.	• Is easily deployed on a local computer, self-hostable and open-source • It allows a high degree of personalization to specific tasks • Can efficiently process and generate both small and high amounts of textual data (due to high amount of contextual tokens considered) • Easily integrable with tools like LangChain and RAG architectures (official connectors exist)	To integrate the knowledge retrieved from the knowledge base through the retrieval model.
OLlama	Motor to download, run, and execute LLM models locally, specifically thought for the Llama family of models.	• Allows for the efficient usage of LLMs, especially for Llama models • Volatile in possible inputs and outputs, as it works with text without requiring specific structure • Easily integrated into greater systems like RAG architectures	To run and use the LLM used for knowledge integration.

Thus, this first step in the RAG component takes as input a query given by the user, and returns the same query with all the existing and available information in the locally-deployed database which might help respond to it. Both are jointly given to the next prompt engineering step.

6.2.2. PROMPT ENGINEERING STEP

Language models often take prompt instructions or questions (queries) as inputs, and carry out the necessary processes, based on the information they possess, to correctly answer them. Therefore, language models often do not directly access any databases to carry out tasks, but rather use the information they are given in the query, as well as the one learned during training, to tackle the mentioned prompts. For the language model to consider the local information extracted from all the data sources available, this information must be given to the model in the prompt. Thus, a complete prompt must be defined for the model to be aware of the local information to tackle the task of answering the given query.

The second step in the RAG component pipeline is the prompt engineering step. This step is aimed at creating a complete and data-rich prompt for the language model based on the original query given by the user and the additional information relevant to the query extracted from the vector database. To carry out this task, a template-based pipeline approach is carried out. This pipeline takes as input a defined template, which is aimed at combining the information retrieved from the knowledge base through dynamic parameters. This template then receives the information extracted from the previous step and builds a new and complete prompt. This prompt includes information regarding the additional information available, the original query given by the user, and the instructions on what the new objective is, which often indicate that the model should respond to the given query by also

considering the information specified. Thus, this second step generates a more complete prompt that is then passed to a language model to obtain the result.

To carry out this task, specific tools are used, as the dynamic templates in this case must be integrated into a more complex executing pipeline, as the LLM in this case will have to deliver the results through multiple interfaces (report, chatbot). Thus, the templates must be able to work both with basic plain text generation as well as chat-based LLM implementations. The tools used are defined in the following Table 7:

Table 7. Existing tools used in the implementation of the prompt engineering task

TOOLS	DESCRIPTION	RELEVANCE	USE
ChatPromptTemplate	Dynamic template definition engine by LangChain aimed at creating LLM-aware prompts.	- Allows the generation of prompts for both text and chat completion - Has LLM-specific optimization, that handles variable formatting, advanced logic and flexibility - Easily integrated into complex execution pipelines	To carry out prompt engineering.

The language model that takes this newly generated prompt as input is aimed at carrying out the specific task defined. Since in this implementation we want the user to have available multiple sources of information (reports and chatbot), the LLM will have to work closely with additional pipelines. Thus, depending on the pipeline executed, the prompt engineering task will generate prompts accordingly, generating more complex instructions for the report pipeline, and faster and simpler instructions for the chatbot pipeline.

6.3.REPORT GENERATION COMPONENT

The third and final processing component of the data contextualization and interpretation module enables the results to be properly presented to the final user. One of the main challenges when working with large amounts of data is the interpretability of the results by humans. In our use case, many processes have been implemented to both contextualize and explain the data available to make them readable and understandable to humans. The final component of the module is aimed at properly forwarding the context-aware and complete data to the final user.

Given the context of this developed project, which is aimed at facilitating police analysis and investigations, two main approaches have been developed to show the information to the final user. These approaches include the generation of complete police reports, which include the status of an investigation on a suspect and all the information gathered to data, and the development of a chatbot interface, with which the police force can interact with and directly as questions to. Both have been developed to tackle the mentioned limitation of human understanding of large amounts of data. On one hand, the police reports developed are aimed at showcasing the police force with the information in a format they usually work with, with the same structure as the reports they develop by hand. These reports are also expected to be directly useful for future processes, as they can also be used as input data stored in the vector database to carry out a well-defined tracking of the previous conclusions reached on a suspect by the system. On the other hand, the chatbot interface allows the police force to ask direct questions to get straightforward and quick answers. This makes the data analysis process easier and quicker when the information required is not content-rich enough for a complete report to be needed on the topic. The chatbot also works as an additional tool for remaining doubts the force might have on a suspect, crime, or investigation.

For the development of the mentioned approaches, different pipelines have been defined. The complete component architecture, including all these pipelines, can be seen in Figure 27.

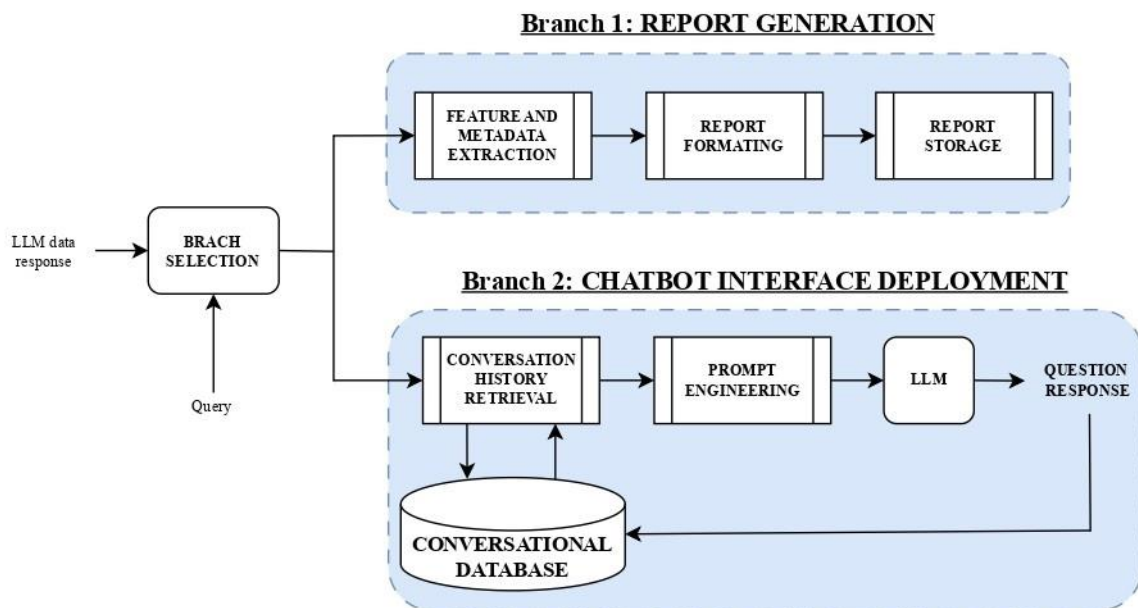


Figure 27. Structure of the report generation component within the data contextualization and interpretation module

As mentioned, in this case, two main executing branches are implemented which are independent to one another. Depending on what the user desires to carry out, the component will trigger one branch or the other. Each of them is composed of a series of steps that will carry out the necessary tasks to complete the needed processing. These branches can be defined as the report generation branch and the chatbot interface deployment branch.

6.3.1. REPORT GENERATION BRANCH

In the context of this work, police reports are important sources of information, as they give an insight to the status of the police investigation and the known information by the police. They are also the main way conclusions reached by the police are stored and shared. Therefore, most of the information gathered from any source is expected to have an associated report which mentions and explains what the source entails and its relevance to the case, working as official registers of all the processes carried out by the police. Thus, one of the main ways that the extracted information in this module is delivered to the police is through the generation of police reports, so the module can be easily integrated into the already existing police methodology.

The report generation branch is aimed at taking the conclusions obtained from the previous component, which includes a context-rich and complete response to the query given to the module by the user. This branch restructures the data given into police reports through a pipeline that implements three sequential tasks: a first feature and metadata extraction task, a second report formatting task, and a final report storage task.

The first feature and metadata extraction task is aimed at properly interpreting the data given and extracting the relevant information to build the report. The task takes as input the complete text and returns the relevant features, information, and metadata to be included in the report. These features and metadata mainly include all the information needed to position the report into the investigation timeline (suspect name, report time, crime investigated, etc.), all the necessary statistics to better understand the data and what it entails (size of data used, behavior-based probabilities, degrees of deviation from usual conduct, etc.) and all the conclusions gathered from the sources which might be helpful for the investigation (context around locations, possible motives behind the crimes, criminal patterns, etc.). Thus, a series of minor functions are developed to correctly extract all the information and define the needed parameters to later integrate into the report structure. These functions work

parallelly within the task, and combine data analysis through machine learning, and prompt engineering tasks. All the extracted information and statistics are associated to defined parameters to better integrate the data in the next step.

The second report formatting step is aimed at taking all the defined parameters and correctly structuring them into the report. Since we wish to generate structured and visually styled reports, an HTML approach will be implemented rather than the templates used before, which generated textual documents which were content-rich, but not visually as appealing. A dynamic HTML template is defined with the Jinja2 framework. This template includes the needed parameters to be filled out with the output obtained from the previous task. An example of the overall structure of the defined template, without any information, can be seen in Figure 28.

INVESTIGATION REPORT

This report includes all the data collected and conclusions reached regarding the suspect with ID "ID TO BE SPECIFIED".

PRELIMINARY REPORT INFORMATION

Suspect ID: "ID TO BE SPECIFIED"
Start Date: "START DATE TO BE SPECIFIED"
End Date: "END DATE TO BE SPECIFIED"

BEHAVIORAL PATTERNS OF INTEREST

- **PLACE:** "PLACE ID TO BE SPECIFIED"
 - **Stationary behaviors** "STATIONARY BEHAVIOR PATTERS DETECTED TO BE SPECIFIED"
 - **Moving behavior** "MOVING BEHAVIOR PATTERS DETECTED TO BE SPECIFIED"

BEHAVIORAL DEVIATIONS (ANOMALIES) OF INTEREST

DURATION OF STAY ANOMALIES

- "STAY ANOMALIES DETECTED TO BE SPECIFIED"

SPEED ANOMALIES

- "SPEED ANOMALIES DETECTED TO BE SPECIFIED"

SUBOPTIMAL PATH ANOMALIES

- "SUBOPTIMAL PATH ANOMALIES DETECTED TO BE SPECIFIED"

UNLIKELY PATH ANOMALIES

- "UNLIKELY PATH ANOMALIES DETECTED TO BE SPECIFIED"

UNLIKELY TRANSITION ANOMALIES

- "UNLIKELY TRANSITION ANOMALIES DETECTED TO BE SPECIFIED"

OBSERVATIONS OF INTEREST

"OBSERVATIONS OF INTEREST TO BE SPECIFIED"

Figure 28. Structural example of final report generated

Finally, once the report is correctly defined, the third and last report storage task is executed. This task is aimed at correctly storing the report for future use by the police forces. To facilitate the access to the gathered document by the force, two storing options are made available to the user. Firstly, the document is always stored in the previously defined Qdrant vector database (Table 5) following the same process used in the same component. This is done for this report to also be considered as an input in future processes. Secondly, the document is also stored locally in the user's computer, for the user to have direct access to it, in case a direct revision of the document is desired or the document is to be stored in any other database not defined specifically for this framework's workflow. Thus, the report is available with all the relevant conclusions reached by the system to the user directly and to the system itself for future use.

6.3.2. CHATBOT INTERFACE DEPLOYMENT BRANCH

Due to the vast amount of information analyzed in this framework, especially when it is scaled to tackle the data of multiple suspects across various investigations, the reports generated might not give a full insight on all the possible conclusions that could be reached given the data and the tools used. Many specific details regarding a suspect or crime might be left out of the report for the sake of clarity and conciseness, as the report is meant to avoid unnecessary details and go straight to the point in a structure manner. Thus, a slight limitation appears if the user of the framework cannot fully grasp all

the details of the data and is left with unanswered questions. To tackle this limitation, a chatbot interface is developed.

The chatbot interface deployment branch is aimed at developing and deploying an easily understood and useful chatbot interface for the user to access. This chatbot interface is expected to answer short questions inputted by using the data available from the framework. To do so, it mimics the RAG architecture used in the core of the data contextualization and interpretation module but in a smaller scale. It makes use of both the documents retrieved by the previous retrieval-augmented generation component, as well as previous conversations between the user and the chatbot. The latter is considered to ensure a linear conversation between the user and the chatbot, where previous conversations are remembered and can be directly referenced, and to ensure an efficient and quick response capability by the chatbot, as previously answered question do not need to be processed again. To carry this out, this second branch has a workflow made up of three main tasks: a conversation history retrieval task, a prompt engineering task, and a final response generation task.

The first conversation history retrieval is aimed at extracting past conversations between the user and the chatbot for them to be considered when answering future questions. To do so, a chatbot history database, ChromaDB, is used, as seen in Table 8. The database stores all the questions and answers in a vectorized form to be able to compute a semantic search later as to only consider the relevant previous answers to answer the current question given by the user. Furthermore, to avoid the database from overflowing or storing too much information, additionally we establish a process to limit the number of previous answers that can be stored. To do so, a FIFO (First In First Out) approach is implemented where, given an established limit of maximum chat instances that can be stored, when the maximum is reached, the oldest instance is eliminated and the new one is inputted. This approach allows firstly to avoid latency problem due to a high number of messages being stored and eliminated from the database. Secondly, this limitation with the maximum number of instances to be stored avoids future problems with the next prompt engineering step, as considering all the relevant past conversations might at some point go over the token limit of the LLM implemented further along in the workflow. Thus, in this first conversation history retrieval step, the vector database containing the past conversational instances is revised, and all the relevant ones are retrieved to be given to the next task.

Table 8. Existing tools used in the implementation of the chatbot interface branch

TOOLS	DESCRIPTION	RELEVANCE	USE
ChromaDB	Vector-based database aimed at language model-based application which is natively integrated with tools like LangChain and LangGraph.	- Free of use, can be deployed easily locally, and its implementation does not require complex infrastructures - Allows for a quick access to the data due to in-memory storage - Easily integrated to structures like RAG architectures due to its native integration with LangChain	To consider past conversations in the chatbot interface

The next task is the prompt engineering task, which combines both the data given by the RAG core retriever, as well as the past conversations extracted by the previous task and defines a new prompt. This prompt instructs the large language model to answer a given question by the user in a shorter and concise manner considering both data sources. In the final response generation task, the language

model takes this prompt and returns the answer to the user through the interface. This question-answer pair is then stored in the vectorized conversational vector database for future questions by the user. For this task, the language model used is the same as the one implemented in the previous component, the Llama3.2 model, explained in detail in Table 6.

7. DEVELOPMENT OF A DATA GENERATION AND MOVEMENT PREDICTION MODULE

In the context of a police investigation, understanding a suspect's behavior is crucial for crime analysis and decision-making. With these models, additionally, another useful task can be carried out, which is the prediction of possible future movements. The prediction of movement allows law enforcement to have an estimation of the possible events that might occur in the future. Although data prediction do not guarantee the exact movements or crimes a suspect might commit in the future, it does reduce the uncertainty around the possibilities, allowing law enforcement to make informed decisions.

Movement prediction is not a simple task, since, as mentioned previously, movement can be modelled in many ways, and the prediction itself can be oriented to many conducts that make it up. In the case of this work, the predictions made will be directly aligned with the defined model of movement and behavior. The main goal of this last data generation and movement prediction module is to generate new movement data, aligned with what the system expects and which can be directly fed to it for analysis, based on the conclusions reached from previous data analysis. Thus, in this module we are essentially predicting future probable movements and behaviors based on what a certain individual showcased previously. To do so, two main tasks need to be carried out. Firstly, from all the conclusions reached by the system regarding the behavioral tendencies and movement patterns of a certain individual, the characteristics of interest, those needed to define the whole movement and behavioral model, must be properly determined and extracted. Secondly, based on those characteristics, new data must be built following the same structure as the one initially extracted by the police (must include coordinates and timestamps), for the system to be able to properly analyze it and extract new conclusions. Therefore, this last module is made up of two main components which carry out these objectives, as can be seen in Figure 29.

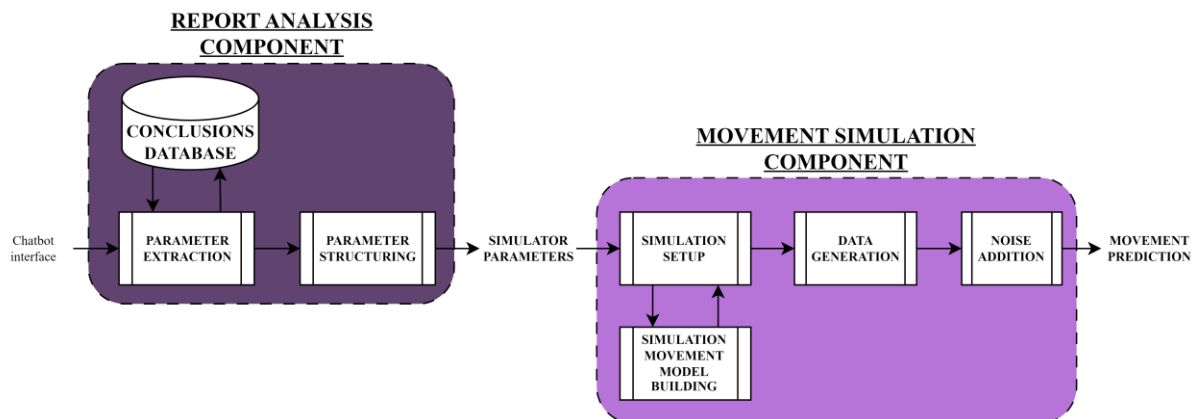


Figure 29. Data generation and movement prediction module architecture

This module is made up of a report analysis component, and a movement simulator component. The report analysis component takes all the reports generated by the system regarding a certain suspect to extract the needed characteristics mentioned. These characteristics will be the direct input of the movement simulator component. For the movement simulator, a simulator previously defined in the same line of work is used, originally developed to generate synthetic data for the validation of a georeferenced data analysis system, but repurposed in this work to be fully integrated in the proposed framework. This simulator will generate as many predicted movement trajectories as desired. The generated data will then be directly reintroduced in the rest of the system, for further conclusions to be extracted.

In this section, we will go over the components that make up this last module. Since the movement simulator was developed in previous phases of the project, only a quick overview of what the

simulator entails will be included in this work, mainly specifying what the simulator takes as input to better understand what the report analysis component must extract.

7.1.A BRIEF OVERVIEW OF THE MOVEMENT SIMULATOR

The movement simulator is aimed at generating movement data, with a series of defined behaviors, based on a movement and behavior model given as input. It essentially mimics real data gathered by standard tracking devices. It was originally developed to tackle the scarcity of available movement data which included realistic behavioral patterns and labels, as well as the even greater shortage in movement data with labelled behavioral anomalies. Therefore, the developed simulator allows for the generation of random movement data that follows a certain model with the emulation of anomalies as well. However, when integrated in this framework, the main goal will not be to generate randomly produced labelled data, but rather to follow a series of given patterns and generate multiple possible future movements with associated chances, based on how likely they are to occur based on past movements.

To carry this out, the simulator architecture is composed of a base simulation structure surrounded by additional layers to add complexity. The base simulation structure takes as input the needed parameters to carry out the simulation itself. This base relies on a movement and behavior model building process, which determines what patterns the simulated data should follow. This process takes as input the needed characteristics to define the movement and behavior models (including information on the places of interest and the behavioral patterns to consider). With this model, and the base simulator structure, the movement data can already be generated, but an additional layer of complexity is added to include anomalies on the data, which is part of the architecture of the simulator, but will not be as used in this use case. This final anomaly profile building process takes as input the needed parameters to define all the anomalous behaviors and their associated probabilities of occurrence. This last layer, although useful, will not be as used in this case, since the goal is to generate data with probable movements. This explained architecture can be seen in Figure 30.

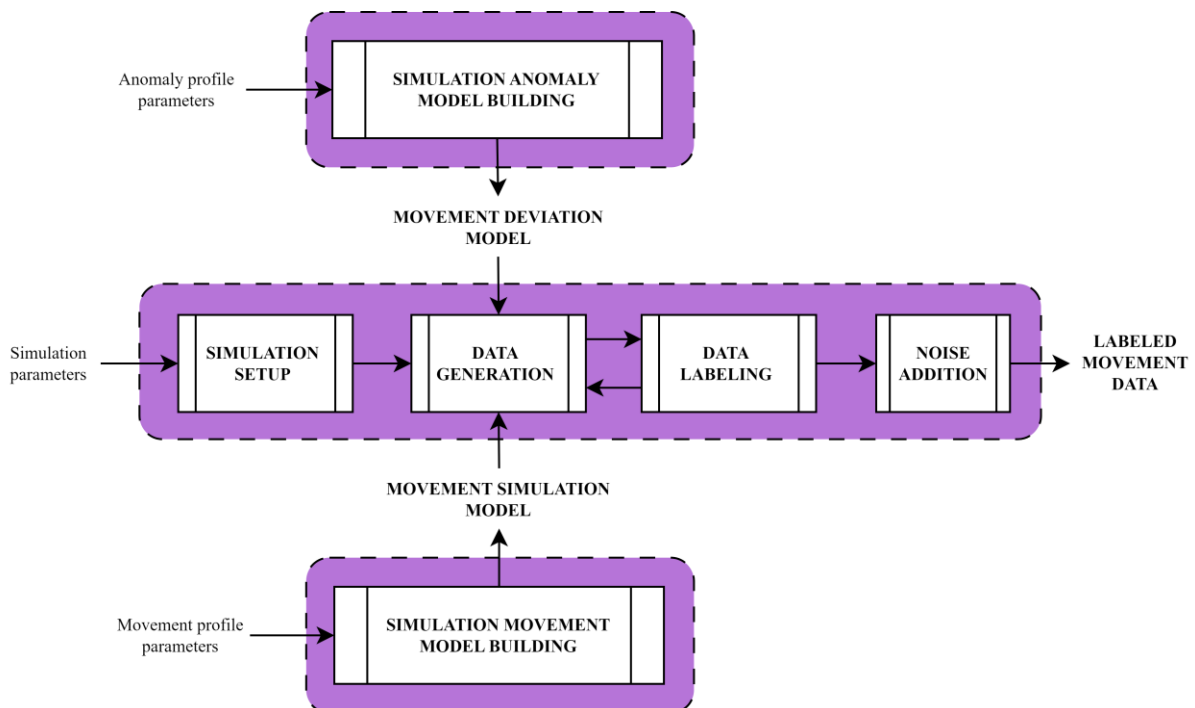


Figure 30. Architecture of the previously developed simulator implemented in the data generation and movement prediction module

Thus, the full workflow of the simulator goes as follows. Based on the parameters given by the user, the simulator first generates, for each of the established trajectories to generate, a sequence of places of interest to be visited by the simulated individual based on the given transition probabilities. Based on this sequence, the movement data is generated as a series of georeferenced points containing a coordinate and a timestamp which follow certain routes to move between the specified places and that

stay at those places once they are reached. Certain behavioral patterns defined by the built model are considered when generating these points, including how long the simulated individual will stay at each of the places, how fast the individual will go when moving between places, and which route the individual will take to move between places. These patterns are taken in to account as the movement is generated point by point with the sampling frequency specified and following the real layout of the streets. Once the sequence is completed in all the trajectories, additional noise is inputted in the data to mimic the real limitations that tracking devices have. This includes temporal noise, spatial noise, and data loss. With the added noise, the data is completed.

To properly carry out the explained workflow, firstly, the simulator will have to be given specific parameters. Given the data regarding a specific suspect, the simulator will need the following parameters to properly predict and generate possible future movements:

- **Simulation parameters**
 - *Sampling frequency* of the data points
 - *Number of trajectories* to generate
 - *Number of places* to visit in a trajectory
 - *Date and time* in which the data should start
 - *Noise parameters*, including the time noise range, the latitude noise range, the longitude noise range, the scale needed to define the average time elapsed between data loss and recovery and vice versa
 - *Suspect ID* associated to the movements
- **Movement profile parameters**
 - *Places of interest* including the label, the coordinates that define it, the mean duration of stay and the standard deviation of the duration of stay
 - *Average speed* for speed behavior
 - *Deceleration rate* and *acceleration rate* for braking and acceleration behaviors
 - *Number of different routes* to consider when moving between POIs
 - *Route probabilities* associated to each of the different routes
 - *Transition probabilities* associated to each of the different possible transitions between POIs

Furthermore, for the data to be realistic and consider the real layout of the streets, the simulation makes use of some external tools. The tools can be summarized as seen in Table 9.

Table 9. Existing tools used in the implementation of the movement simulator

TOOLS	DESCRIPTION	RELEVANCE	USE
Open-Source Routing Machine (OSRM) engine	Open-source routing engine designed to compute real distances and routes using the OpenStreetMap (OSM) data.	• Free of use, can be deployed easily locally, and its implementation does not require complex infrastructures • Allows to compute the optimal route between coordinates considering the real layout of the streets • Considers the real curvature of the Earth and allows to compute street-aware distances between coordinates	To generate movement data considering the real layout of the street in the movement simulator

In the case of the proposed framework, the goal is to properly integrate the simulator into a larger and more complex workflow, in which the parameters are extracted automatically based on previous movements analyzed and the generated data is directly inputted back into the system for analysis. Thus, in this case, the simulator, as mentioned, aims at making movement approximations, predicting what a certain suspect might do next based on previous movements and behaviors.

7.2.REPORT ANALYSIS COMPONENT

To properly integrate the movement simulator into the larger framework, an automatization process must be carried out which extracts the needed parameters for the simulator based on the previous conclusions reached by other modules. Thus, both the specific requirements of the simulator as well as the outputs of previous processes must be considered in the design of this report analysis component. In the proposed framework, many outputs refer to the possible values of the simulator parameters. However, with the purpose of optimizing the parameter extraction process, only one data source will be considered, which is the output of the data contextualization and interpretation module, which will be heavily connected to this last simulation module. Firstly, one of outputs of this module consists, as previously explained, of complete reports that include all the information and conclusions gathered on a specific suspect or crime to date. This report includes both the expected behavioral patterns extracted as well as some of the anomalies detected. Secondly, the module gives the user access to a chatbot interface, which will also be used to trigger the prediction and specify certain parameters.

Thus, this report analysis component will have to tackle the processing of two inputs from which to extract the needed information by the simulator. To carry this out, the architecture seen in Figure 31 is implemented. This architecture carries out the needed tasks through two main steps: a parameter extraction step and a parameter structuring step.

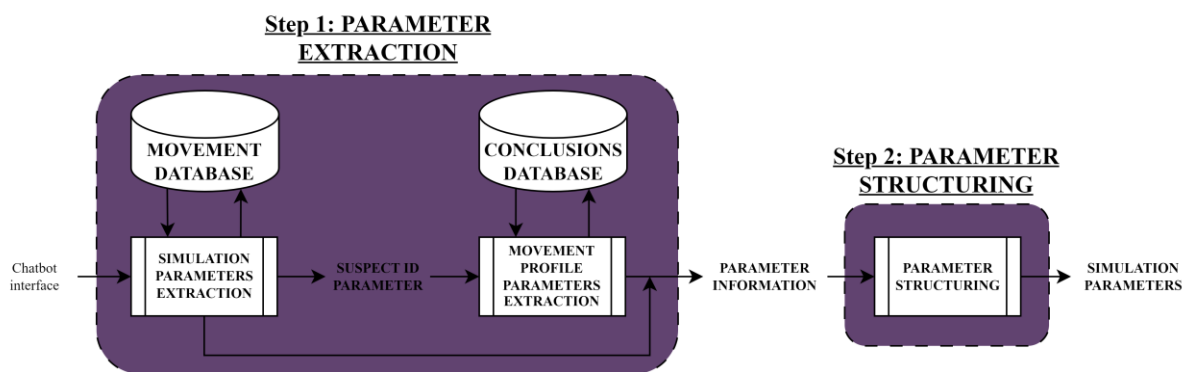


Figure 31. Structure of the report analysis component within the data generation and movement simulation module

The first parameter extraction step is aimed at processing the information given by the two inputs to extract both the simulation and the movement profile parameters for the simulator. Depending on which input is analyzed, a slightly different task will be carried out, as the information will be given in different formats. Once all the needed information is extracted, the parameter structuring steps reformats it to ensure that the input given to the simulator is in the format required. Once the input is given to the simulator, new data predicting future behaviors and movements will be defined and fed directly back to the system, in this case, the behavior analysis and anomaly detection system.

7.2.1. PARAMETER EXTRACTION STEP

As previously mentioned, this component is highly connected to the data contextualization and interpretation module. Thus, the two tasks that make up this first step are aimed at tackling the two outputs given by said module. The two outputs are considered since not all the parameters are strictly related to the suspect, but rather to what the user desires. More specifically, while the movement profile parameters can be extracted from generated reports that go into detail on the patterns of behavior detected in a suspect, the simulation parameters depend solely on what the user desires (who the suspect to consider should be, how many days should be predicted, how much noise should be included, etc.). Therefore, the first task in this step, the simulation parameters extraction task, will extract the simulation parameters from the chatbot interface, while the second task in this step, the movement profile parameters extraction task, will extract the movement parameters from previously generated reports.

Starting with the first simulation parameters extraction task, this task is triggered through the chatbot interface. If the user wishes to simulate future movements, it must do so through the chatbot, which will directly redirect the process to this first task. Through the interface, explained in detail in Section 6.3.2, the user will specify the needed simulation parameters, with the most relevant being the ‘Suspect ID’ parameter, as it will be needed for the next task. The remaining parameters, as briefly mentioned in the previous section, will include the sampling frequency of the generated points, the number of trajectories, the number of places, the starting date of the data, and the noise parameters. To facilitate the process for the user, some default values will be attributed to each of the parameters. The user will have the option to either give the specific value the parameters should take, or to indicate to the chatbot that the default value is desired, for all the parameters except the ‘Suspect ID’ parameter, as the user must specify which suspect the prediction is based on. For the remaining parameters, some default values are computed based on previous movement data extracted from the suspect while others have an established value.

Starting with the ones extracted from previous movement data, the ‘Sampling frequency’ parameter and the ‘Number of places’ parameter are both defined considering the sampling frequency of previous movement data files and the usual number of places included in each of the trajectories from previous movement data. As for the remaining parameters, the ‘Number of trajectories’ parameter takes by default the value 1, indicating that only one day should be predicted, and the ‘Noise parameters’ take by default the value of 0 in all the possible noises to be added (temporal, spatial and data loss), indicating that no noise should be simulated on the data. Finally, the ‘Date and time’ parameter combines both approaches. On one hand, the date is established by default as the very next day to the current day in which the simulation is being triggered. On the other hand, the starting time is determined by considering the usual starting time of previous movement data files. Thus, depending on whether the user has specific simulation patterns in mind, or simply wants to generate a straightforward prediction, the simulation parameters will take different values.

Based on the input given to the simulation parameter extraction task, the movement profile parameters extraction task considers different reports. More specifically, the only input given to the first task that is directly considered by this second task is the ‘Suspect ID’ parameter value. This value is used to carry out a retrieval process like the ones implemented in the previous data contextualization and interpretation module, but much simpler. The generated reports are saved under specific file names for them to be easily retrieved both by the police force as well as processes such as the one implemented in this task. The most recent report generated on the suspect with the specified ID is retrieved from the database by considering the report name. The most recent one is the one considered as it is the one with the most updated conclusions. Once this document is extracted, the process of determining each of the needed movement profile parameters starts. To do so, an already pretrained external model is used, as specified in Table 10.

Table 10. Existing tools used in the implementation of the movement profile parameters extraction task

TOOLS	DESCRIPTION	RELEVANCE	USE
Mistral 7B Instruct model	Large language model from the Mistral family, presented as a fine-tuned version of the Mistral 7B model. It has 7.3 billion parameters and approximately allows up to 32.8k tokens of context.	- Open-source model that can be run on a computer with GPU - Has a context window of 32.8k tokens, which roughly translated to 100 pages of text (ideal to process long reports) - It is optimized for data extraction from long documents and can return structured responses	To analyze conclusion reports for the movement profile parameters extraction task

The language model takes the whole report as input. Through a series of prompts, each of the needed parameters are extracted. As slightly mentioned previously, these parameters include ‘Places of

interest', 'Average speed', 'Deceleration rate', 'Acceleration rate', 'Number of different routes', 'Route probabilities', and 'Transition probabilities'. The language model extracts all the needed information to properly define each of the parameters. In the case of the 'Places of interest' parameter, the model must identify in the text the label given to each place, the coordinates that define each place, and the behavioral patterns needed to define the stationary behavior in each place, which includes the mean duration of the stay at each place, and the standard deviation from said duration at each place. As for the 'Average speed', the model must extract the patterns needed to define the speed behavior, including the mean speed of movement between places of interest, and the standard deviation from said speed. The 'Deceleration rate' and 'Acceleration rate' parameters can be extracted from the text, but also have associated default values in case the suspect usually takes means of transport in which this rate is not as relevant (e.g. walking), and has not been analyzed and included in the report. In said cases, the value for both parameters is set to 0.5 meters per second, which is the standard acceleration and deceleration rate of a walking person. Moving on to the 'Number of different routes' parameter, the model must extract from the text needed coordinates to define each of the routes must be given, and all the possible routes that the suspect has taken in the past as well. The 'Route probabilities' parameter will include, for each of the routes specified, the likelihood of the suspect taking said route to move between places of interest. Finally, the 'Transition probabilities' parameter is defined through the probabilities associated to each transition between places included in the text.

Thus, at the end of this second task, all the needed parameters will be extracted and defined with the information given by the user and the information concluded in the reports from all the previously implemented processing modules. Both things are directly given as input to the next step.

7.2.2. PARAMETER STRUCTURING STEP

Although the information extracted in the previous step is correct, the simulation does not have a process integrated at the beginning of its architecture to properly identify what information defined each of the parameters. The simulation expects the parameters to be given in a specific format, with no data preprocessing taking place before the main simulation pipeline. Therefore, the information given to this second step must undergo a structuring and reformatting process before it can be given as input to the simulator.

The main goal of this second step is to analyze all the information extracted in the previous step to define the final parameters. Although this last step is much simpler than the one implemented before, it is essential for the simulator to work properly. Thus, depending on what the parameter value is, the data given will be accordingly restructured, defining either unique values or dictionary structures (for the probabilities associated to routes and transitions), and reformatted, to transform string values into numeric or datetime format. Once this is done, each of the values are associated to the specific parameter names and directly given to the simulator. The simulator will then simulate the needed data considering only the patterns given to generate realistic and possible future movements the suspect might make with behavioral patterns the suspect might carry out.

8. RESULTS

The proposed framework implements many processing modules, each giving different outputs. Therefore, the task of evaluating the framework's performance is a complicated one, as no one way is defined to do so. With the goal of properly evaluating the reports generated by the framework, an evaluation methodology has been developed, and an evaluation scenario has been created, to evaluate how the framework would work in a real scenario, and if the results given by it are useful, coherent, and precise. In this section, the evaluation methodology and scenario will be presented and implemented on the framework, and the obtained results will be analyzed.

8.1. EVALUATION SCENARIO

To evaluate the developed framework, a simulation of a real use case is carried out. The idea behind this evaluation is to evaluate how effective the framework is at offering support in a real investigation. To do so, the possible timeline of a real investigation is designed, with the periodic intervention of the framework to draw conclusions regarding the data gathered at different stages of the investigation. The goal, as mentioned in the previous evaluation methodology section, is to evaluate firstly if the conclusions drawn from the data and gathered on the generated reports are useful, meaning that the content included in them is relevant to the investigation and can be used by the police, and if they are reliable and truthful, meaning that the content included is also true and refers to correct information.

The defined evaluation scenario goes as follows. In 2023, there was an increase in the number of theft-related crimes in the Tenderloin District area in the city of San Francisco, California in the United States of America, ranging from pickpocketing to armed robberies in the streets. Many of the robbed items were later illegally sold in areas like subways by the robbers. Even with this knowledge, the police often have a hard time catching the robbers selling stolen items, as they are quick to flee the scene when police are called. To tackle this issue, police revise the criminal database to extract information on criminals previously apprehended on robbery charges. Individuals are filtered, which have a history of related crimes in the same area. The goal of the police is to follow one or more suspects to determine whether they are guilty or related to the increase of these crimes.

To simulate this investigation, various sets of georeferenced movement data are generated using the same movement data simulator implemented in the data generation and movement prediction module (explained in detail in Section 7.1). The simulated investigation spans over 30 days. Every 10 days, the police input the gathered movement data into the system for the framework to draw conclusions regarding the suspect and his/her possible connection to the crimes. For each of these data analysis, it is expected for the framework to be able to arrive at increasingly more accurate conclusions which point to the right direction. A visual example of what the evaluation scenario timeline entails can be seen in Figure 32.

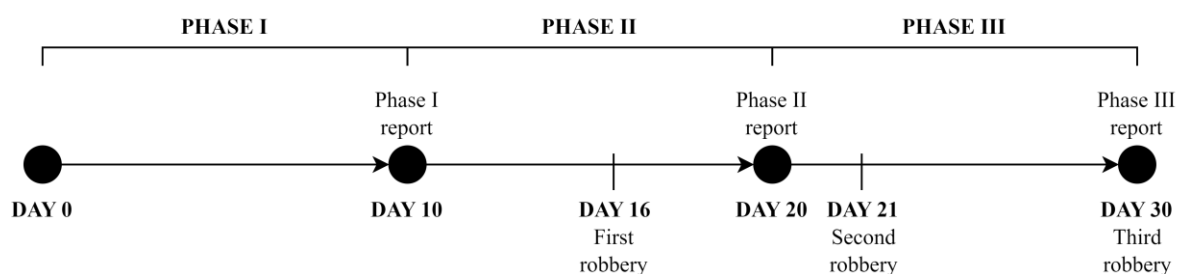


Figure 32. Evaluation timeline with real events and investigative events

This evaluation timeline is carried out based on a simulated criminal's movements. The criminal is guilty of the investigated crimes, and the police must be able to correctly deduce that based on the conclusions reached by the system using solely movement data, previously reached conclusions, and external contextual data. What the criminal does, which as mentioned should be flagged by the framework, is the following:

- Actual crimes committed during the first phase of the timeline: From the 5th of June of 2023 up until the 14th of June 2023, the criminal sells previously stolen items near the 16th St

Mission subway station in the Mission District of San Francisco, which is a common selling spot for many criminals, as it is one of the main subway stations near the high-criminal areas. The criminal does not steal anything on the first 10 days of police tracking, but rather only goes every day to sell previously acquired items. Aside from the subway station, the criminal additionally visits other places which are not related to criminal activity, but to daily errands, which includes his/her residence (a small house near Hayes Valley in San Francisco, California.), a fast food restaurant near the subway where the criminal usually eats his/her meals (in the shopping mall at 16th St in San Francisco, California), and a small supermarket where the criminal buys groceries (at the Fillmore District in San Francisco, California).

- Actual crimes committed during the second phase of the timeline: From the 15th of June 2023 to the 24th of June 2023, the criminal carries on with his/her usual schedule, trying to sell stolen items almost every day, as well as the visits to the additional non crime-related locations (home, restaurant and supermarket), but on the 20th of June 2023, in the early afternoon, the criminal commits a robbery on Ellis St., in the Tenderloin District of San Francisco. The criminal robs a laptop from a car before fleeing the scene. The criminal is not captured by the police.
- Actual crimes committed during the third phase of the timeline: From the 25th of June 2023 to the 4th of July 2023, the criminal again carries his/her usual schedule, visiting the usual places (the subway, the home, the supermarket, and the restaurant). However, during this 10-day period, the criminal commits two robberies. The first one occurs on the 30th of June 2023 on a street in Union Square, San Francisco, where the criminal steals a handbag containing a laptop, a personal phone, a set of keys, and approximately 300 dollars in cash. In this first robbery, the criminal is not caught. The second robbery in this period occurs on the 4th of July 2023 in Taylor St. in the Tenderloin District in San Francisco, where the criminal stole a black leather Coach handbag from a car, containing some personal belongings including a gold watch, a set of keys, and a series of personal documents. In this second robbery, again, the police do not capture the criminal. The criminal goes on the very same day to his/her usual selling spot to sell the gathered items.

Parallely to the actual events occurring, the police are running the mentioned investigation. The criminal explained is one of the many suspects the police consider possibly involved with theft-related crimes. Thus, we specify the steps carried out by the police at the same time as the crimes are committed. Additionally, the role the framework should have in the investigation is also specified as follows:

- Police work carried out during the first phase of the timeline: On the 5th of June of 2023, after identifying the known data associated with each of the suspects that might be connected to the crimes, the police start to remotely track the suspects through spyware on mobile phone devices. This tracking generates a series of movement tracks built with coordinates and timestamps. The police then feed these tracks every 10 days to the framework, to see if any behaviors of interest could be identified. Thus, for all the suspects, the first data input is given with the movement information gathered from the 5th of June up to the 14th of June. After this first analysis, the framework is not expected to be given major information aside from the usual behavioral and movement patterns of the suspect and why they might be of relevance in the specific use case. However, the fact that the suspect spends a long amount of time in places known for fencing stolen property should be noticed and determined as relevant.
- Police work carried out during the second phase of the timeline: From the 15th up until the 24th of June 2023, the police decide to continue the tracking activity due to the conclusions reached in the previous phase. They are called for a robbery occurring on the 20th of June on Ellis St., in the Tenderloin District of San Francisco. No robber is caught, but a police report is filled out with the data given by witnesses and the victim. On the 24th, the new data gathered is inputted into the system. An anomaly should be properly detected by the system, regarding the suspect's movements around the Tenderloin District. A correlation between the report written on the 20th and the locations visited by the suspect should also be made.
- Police work carried out during the third phase of the timeline: Finally, based on the conclusions reached up until now, which place the suspect at known stolen items selling sites,

and possibly at the site of a robbery committed 5 days before, on the 25th of June the police decides to keep tracking the suspect to see if more correlations between crimes and his/her locations can be made. They get two calls regarding theft in this time, both in the same district, one on the 30th of June, around Union Square in San Francisco, and another on the 4th of August, directly within Tenderloin District. In both crimes, a report is drafted with the known information, but no criminal is caught. Given the proximity in distance to the previous robbery, and the similar modus operandi, at the end of the 4th of July, when all the data regarding the suspect is gathered, it is inputted into the framework to see if the crimes can be correlated to the already detected possible suspect from the previous phase. The framework should detect two anomalous behaviors regarding the two robberies and correlate them with the reports written in space and time. Enough information should be available at this point in the investigation into the framework to conclude a strong correlation between the crimes and the suspect. Due to the strong conclusions gathered, the police should have enough evidence to detail the suspect. Once the data is processed, a movement prediction is done to determine the probable times and locations at which the suspect will be selling the stolen items. The police should be able to apprehend the suspect selling the reported stolen items in the next days with the predictions.

Considering this evaluation for both perspectives, the actual occurrences and crimes committed by the criminal (the ground truth) and the steps taken by the police (with predictions on what is required by the system), the evaluation can be carried out, considering the expected information that the reports include (the correlations that should be deduced at the end of each phase), and the specific details that should be included (the actual real data regarding the crimes committed), as is further explained in the next section.

8.2.EVALUATION METHODOLOGY

As investigated in depth on Section 2.3, established validation approaches for language-based models are still emerging, with many experimental and innovative solutions coming forward. Given the specific goals set for the developed framework, the texts generated must go beyond just being coherent and comprehensible, they must additionally include the necessary characteristics for them to be useful in an actual police investigation. Therefore, the evaluation of the proposed framework is not as much based on the evaluation of the individual components that make it up, but rather, how they work together to deliver the expected results. Thus, the evaluation is aimed at the result, to ensure that all the processes have returned the proper outputs.

The result is evaluated in two regards. Firstly, the report generated with the framework must contain the necessary content for the human user to be able to draw useful conclusions. Although the data tackled in this framework can branch out into various interesting topics, the reports must include the required topics regarded as relevant in the context of a police investigation. Thus, from all the output obtained from each of the components that make up the whole framework architecture, it must be first evaluated that the relevant information and topics mentioned in each one are also included in the final report, and that conclusions that branch out into trivial, non-related or non-useful topics. To carry this out, a content-based evaluation framework is developed, which evaluates whether the reports generated in each of the evaluation phases contain the topics needed for the police to draw the necessary conclusions from them.

Secondly, the report generated must contain truthful and accurate information that can be related to real data either contained in the reports or found online. None of the implemented tools should suffer from hallucinations which could jeopardize the integrity of the decisions made by the police. Thus, the data should be factual, without containing possibly biased opinions made by the tools, and accurate, containing only real and verified information. This second evaluation regard is especially relevant due to the sensitive nature of the application use in law enforcement. Thus, a second fact-based evaluation framework is developed, which evaluates whether the generated reports contain true and fact-based information to avoid disinformation and loss of trust in the results.

Furthermore, it is worth mentioning that the implemented structures that were developed in previous phases of this line of work will not be further evaluated, as they were validated when developed.

Nevertheless, the content-based evaluation methodology must ensure that the results extracted from said structures are considered in the result, and the fact-based evaluation methodology must ensure that the information regarding the output of said structures is factual and related to real data. Thus, the individual performance of the behavior analysis and anomaly detection system and the movement simulation integrated in the data generation and movement prediction module will not be evaluated, but the evaluation will ensure that the conclusions reached by both are correctly considered in the generated reports.

8.2.1. CONTENT-BASED EVALUATION METHODOLOGY

The first evaluation methodology is oriented toward the content included in the reports. The final report generated must include all the conclusions of interest reached by each of the modules in the framework for the user to have all the available content and data analysis required to deduce the needed insights into proper decision-making. The final report is defined using, as previously seen, a template-based approach, where a template is filled out using the data extracted and language model prompts, so the structure itself specifies what should be included in the report. However, we must ensure that the topics of interest are included in the report, meaning that certain topics are not simply mentioned in the titles, but are actively commented on. To carry this out, a content-based evaluation framework is developed, which takes as input the generated report, and returns a series of metrics to establish how well certain topics are included. The architecture of this evaluation approach can be seen in Figure 33.

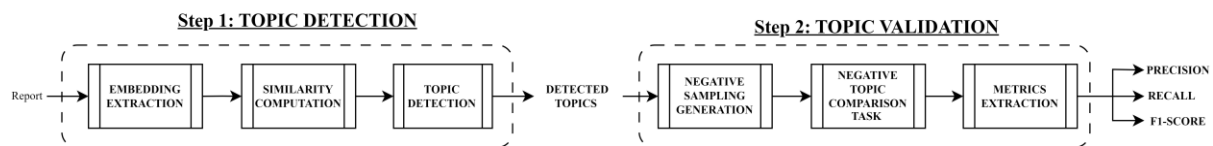


Figure 33. Content-based evaluation framework

The content-based evaluation is aimed at determining how many of the required topics are mentioned in the report considering the importance given to said topics. The process implemented is based on the comparison in semantics between the topics mentioned in the report and topics required, and the computing of the level of compliance of the text with the required topics based on the associated importance to each one. This process is carried out through two main steps, which is the topic detection step, and the topic evaluation step.

TOPIC DETECTION STEP

Starting with the first topic detection step, this step is aimed at determining which topics from the ones required are discussed in the text. However, an additional nuance is considered in this first step. We do not only want certain topics to be slightly discussed, in some cases the goal is to determine not only if the topic is mentioned, but if it is mentioned in the depth required. Depending on which report we are evaluating, the depth expected of certain topics might vary, e.g. we expect the report on the conclusions of phase 2 to go more in detail on anomalies detected (transition anomaly due to robbery committed) than the conclusions of phase 1, where anomalies are not detected as that data must be used as historic data for behavioral modelling. Therefore, we must not only ensure that the overall text contains the topics, but the topics themselves are discussed with an established detail. This is done through the specification of mention thresholds and detail level.

Mention thresholds are thresholds that establish how semantically similar a text must be to a topic to consider that the topic as mentioned in the text. Detail level are density types ('low' or 'high') of the mentioned topic in the text. Both are highly dependent on the evaluation taken place, as the ground truth of expected topics and topic densities in the text vary depending on the stage of the investigation. As previously mentioned, in each of the defined evaluation phases, what is required for the report to contain to consider that law enforcement can properly deduce the needed information is briefly mentioned. These exact expected topics, topic mention thresholds, and topic density threshold of each phase, which are given as input next to the report, are established beforehand.

Works like the one presented by Ma et al. [127], although not focused at the same goals as the ones presented in this project, do present interesting conclusions regarding which thresholds should be considered in cosine-similarity-based scenarios, which is the case of this project, as the cosine similarity measure will be the one implemented to compare the topics with the text in the reports. The mentioned work is aimed at automating the process of extracting relevant topics from complex textual data, which is not the focus of our work. However, to reach their goal, Ma et al. [127] carry out a series of evaluations over different threshold values for similarity scores. It is concluded that thresholds between 0.4 and 0.5 are efficient in the task of balancing the consideration of relevant words and the exclusion of less useful words. This can be directly translated to our use case, where thresholds between 0.4 and 0.5 are low enough to filtering out segments of the text that do not align enough with a topic for that topic to be considered present in the text while being high enough for relevant segments to not be missed. Thus, these 0.4 and 0.5 values will be used to establish the mention thresholds to detect topics on the text. Taking this result into account, the topics to consider, as well as the mention thresholds and the detail types associated with each one at each of the phases of evaluation are defined.

Starting with the topics, the goal is for all the conclusions reached by the behavioral analysis and anomaly detection system to be reflected in the reports, but with additional context regarding the thefts carried out by the followed suspect. Thus, the topics are defined accordingly, in a short manner for the comparison to be robust to differences in topic formulation in the report, and for the similarity scores to not be greatly affected by structure of the sentences, but rather by content alone. The considered topics are as follows: *Stationary behaviors*, *Usual stay duration at places*, *Moving behaviors*, *Usual speed between places*, *Usual transitioned paths*, *Duration of stay anomalies*, *Speed anomalies*, *Suboptimal path anomalies*, *Unlikely path anomalies*, *Unlikely transition anomalies*, *Theft*, and *Crime*.

As for the mention thresholds, they are defined considering the previously presented work, as well as the context behind each of the reports. Depending on the phase in which the report is generated, we expect for more information about certain topics to be known, and therefore mentioned, as the investigation progresses. Thus, in the case of newly discovered information (the behavior patterns in phase I, and the anomalies and possible crimes in phases II and III), we will take the lower end of the threshold range previously mentioned, with these topics at specific phases taking 0.4 as their mention threshold value. In the case of known information explored upon in newer phases (the behavior patterns in phases II and III), we will take the upper end of the threshold range previously mentioned, with these topics at specific phases taking 0.5 as their mention threshold value. Finally, in phase I, since nothing is yet known about the suspect, the topics regarding later reached conclusions, like the anomalies and the possible crimes, are not expected to be mentioned greatly beyond the titles. Thus, for those cases, a threshold of 0 will be established. A summary of all the mention thresholds defined for each of the topics at each of the evaluation phases can be seen in Table 11.

Table 11. Mention thresholds for topic detection in each of the phases of the evaluation scenario

	PHASE I	PHASE II	PHASE III
Stationary behaviors	0.4	0.5	0.5
Usual stay duration at places	0.4	0.5	0.5
Moving behaviors	0.4	0.5	0.5
Usual speed between places	0.4	0.5	0.5
Usual transitioned paths	0.4	0.5	0.5
Duration of stay anomalies	0.0	0.4	0.4
Speed anomalies	0.0	0.4	0.4
Suboptimal path anomalies	0.0	0.4	0.4
Unlikely path anomalies	0.0	0.4	0.4

Unlikely transition anomalies	0.0	0.4	0.4
Theft	0.0	0.4	0.4
Crime	0.0	0.4	0.4

Finally, as for the detail type associated to each of the topics, overall, the idea is for the report to mention the needed topics in enough detail for them to be relevant. This detail type label is highly connected with the pervious mention threshold value, as topics which are expected to be mentioned a lot, are equally expected to be mentioned in detail. Therefore, topics with previous ‘low’ thresholds will be associated with a low detail level expected and vice versa. However, it is important to mention that this label does not indicate the same as the previous one. While the other threshold determines how a topic is regarded as ‘mentioned’ in a statement, this label is used to indicate how the topic is detected in the text, with low-labeled topics only expected to be mentioned in general, and high-labeled topics expected to be mentioned both in general and in enough sentences to consider that the topic has been explained with enough depth. A summary of all the detail types associated with each of the topics at each of the evaluation phases can be seen in Table 12.

Table 12. Detail types for topic detection in each of the phases of the evaluation scenario

	PHASE I	PHASE II	PHASE III
Stationary behaviors	Low	High	High
Usual stay duration at places	Low	High	High
Moving behaviors	Low	High	High
Usual speed between places	Low	High	High
Usual transitioned paths	Low	High	High
Duration of stay anomalies	Low	Low	Low
Speed anomalies	Low	Low	Low
Suboptimal path anomalies	Low	Low	Low
Unlikely path anomalies	Low	Low	Low
Unlikely transition anomalies	Low	Low	Low
Theft	Low	Low	Low
Crime	Low	Low	Low

So, considering the specifications given in Tables 10 and 11, as well as the report in each case, the first topic detection step goes as follows. Firstly, an embedding task is carried out. Vector representations of the data are computed using the same all-MiniLM-L6-v2 embedding model (explained in more detail in Table 1) on three types of texts, which include the whole text from the report to get a global embedding, each of the expected topics, and each of the sentences detected on the text using a natural language toolkit (NLTK). The latter is computed for the final topic filtering task, since only considering the similarities between the global text and the topics in the detection will not consider the density of topic discussion. Thus, given these embeddings, the next similarity task is carried out both between the global text embeddings and with each of the sentence embeddings through cosine similarity, as briefly mentioned previously. At the end of this task, we will have a similarity score associated with the overall semantic similarity between the global text and the topics, as well as with all the sentences in the text and the global topic.

The final detection step considers the similarity scores, as well as the mention thresholds and detail level required from each of the topics. Starting with the mention evaluation, a topic will be considered mentioned in the text if the global text embedding has a semantic similarity to it greater than the

mention threshold associated with it. As for the detail level evaluation, if a topic has a low detail level required, only a mention will be enough to consider it detected. This means that for required low-detailed topics, if the similarity extracted is greater than the associated threshold, the topic is defined as properly explained in the text. If the topic has a high detail level required, the number of times the topic is mentioned in the text must be greater than an established minimum density threshold. This number of times is computed considering the total number of relevant sentences on the text (sentences with a similarity above an established minimum, to ensure that transition or bridge sentences, which do not give major information, are not considered), and the number of times a sentence has included a certain topic, computed again considering the similarity between the sentence embedding and the topic embedding. Thus, for required high-detailed topics, if the similarity extracted between the global text and the topic is greater than the associated threshold, and the topic density level on the text is also greater than the established minimum, then the topic is defined as detected and properly explained on the text.

TOPIC EVALUATION STEP

Moving on to the second and final step of this architecture, a topic evaluation step is carried out. The goal of this second step is to determine a series of classification metrics to evaluate the performance of the framework from a content point of view. The reason behind the usage of classification metrics is due to the nature of the problem. Although this is not a traditional classification problem, given each of the expected topics, we can consider them ‘classes’, in the sense that the prediction values will be 1 if the topic is mentioned on the text, and 0 if the topic is not mentioned on the text. However, this brings forward a series of limitations, especially in the computation of false positive-dependent metrics like precision. Since we are working with all the topics that should be mentioned, the values in the ground truth vector used in classification-based evaluation approaches will take only values of 1. Therefore, no false positives can occur in the data, as all the topic detected will automatically be a true positive. This does not hinder the computation of false positive-based metrics, but it does reduce their significance, as the evaluation value they give does not reflect the actual performance regarding false positives. The first task implemented in this second step is aimed at tackling this limitation.

The first task in the topic evaluation step is the negative sampling generation. This task is aimed at generating new synthetic topics that should not be included in the report. These topics are expected to be somewhat related to the overall theme but should not be mentioned in the actual text. This includes things like behavioral patterns not considered by the framework, anomalous behaviors not considered by the framework, or the mention of crimes not included in the timeline. To carry this out and ensure that none of the topics generated clash with the topics that are expected to appear in the report. Thus, a prompt-based approach is implemented in this task. Firstly, the number of topics required is used as the number of new topics to generate as to avoid imbalance in the evaluation (we have the same number of 1 and 0). Then, we implement a loop generation prompt that will take as input the required topics as well as the previously generated prompt to ensure two constraints are met: that the generated topic is related in theme with the required topics but are not aligned in content (are contradictory or different), and that the generated topics are not repeated in regards to previously generated ones. Thus, at the end of this first task, we have a series of topics, some of which should be considered in the report, and some which should not.

Since these new topics do not have any associated mention threshold or detail level, to evaluate whether they are in the text at all, a minor negative topic comparison task is carried out, where the global embeddings are compared to each new topic only to see if it is mentioned. If it is, then it will be considered as included in the report. This task is thus greatly interconnected with the final topic evaluation task, in which the metrics are computed. The expected values and the predicted values (of topic inclusion or not) are compared against each other and a series of metrics are computed considering the weight associated with each of the topics, as the inclusion of certain topics is more important than others. The metrics computed in this last task are defined in Table 13.

Table 13. Metrics to carry out the content-based evaluation

METRIC	USE CASE DESCRIPTION	TARGET VALUE
Precision	The precision metric defines the proportion of detected topics that are correct, meaning how many of the topics detected in the report belong to the list of required topics.	A value close to 1 means that almost all the topics included in the report are correct.
Recall	The recall metric defines the proportion of required topics that are detected, meaning how many of the required topics defined are properly included in the report.	A value close to 1 means that almost all the required topics were included in the report.
F1-Score	The F1-score metric defines the harmonic mean between the precision and recall values, meaning that it summarizes the correctness of detected topics and the completeness of topic coverage.	A value close to 1 means there is a good balance between including only required topics and not missing any of them.

Additionally, weights of importance are considered when computing the metrics. These weights are a way of adding importance to certain topics, as they are considered more by the metrics. This means that topics with a higher importance associated to them will have an impact on the metric such that if they are not detected, then the metric will be lower than the one it would have taken in a non-weighted approach, and vice versa, where if the topic is detected, then the metric will have a higher value than the one it would have taken in a non-weighted approach. However, since mention thresholds and detail types are already considered in the detection of topics, these weights are more used to tackle the issue of the lack of mention of anomalies and possible crimes in the first phase. In case the first topic detection step does not detect all the required topics on the first phase due to the lack of information, lower importance weights are given to these topics to consider that, although they are required topics, they are not expected to be fully considered in the first report. A summary of the importance weight given to each topic at each phase can be seen in Table 14.

Table 14. Importance weights for topic evaluation in each of the phases of the evaluation scenario

	PHASE I	PHASE II	PHASE III
Stationary behaviors	1	1	1
Usual stay duration at places	1	1	1
Moving behaviors	1	1	1
Usual speed between places	1	1	1
Usual transitioned paths	1	1	1
Duration of stay anomalies	0.5	1	1
Speed anomalies	0.5	1	1
Suboptimal path anomalies	0.5	1	1
Unlikely path anomalies	0.5	1	1
Unlikely transition anomalies	0.5	1	1
Theft	0.5	1	1
Crime	0.5	1	1

Thus, based on the weights, for each generated report at the end of each of the three phases of

evaluation, the numerical metrics explained previously are computed to determine the performance of the framework in the task of generating reports with the needed topics included.

8.2.2. FACT-BASED EVALUATION METHODOLOGY

The second evaluation methodology is oriented toward fact-checking and the verification of content accuracy. Even if the final report includes all the topics needed, the actual information given might be misleading and false. Although the data used is expected to come from trusted sources, many of the processes implemented during the data analysis might end up generating certain conclusions that are incorrect. Specifically, there is one main source of possible false information generation, which is the hallucinations caused by the language models. Language models, more often than not, struggle with managing hallucinations, which is when the model generates data that seems correct but is actually made up and incorrect. In the case of this framework, two main processes might be victim of this phenomenon, which is when text is generated for contextualization (e.g. when generating the synthetic report or when contextualizing the behavioral model) or when correlations are drawn for the final conclusions (e.g. when identifying interesting coincidences in space and time between a crime and the suspect's movements). The hallucination problem is directly tackled on the code, as the prompts given to these types of models are designed very specifically to avoid the model from making up erroneous and incorrect information for the sake of returning something. Nevertheless, it is important to ensure that prompts have properly carried out the task, and that none of the processes implemented in the framework have returned false information.

To carry out this evaluation, a fact-based evaluation framework is developed, which takes as input the generated final report and ensures that the statements made on the report align with the truth regarding the investigated case. Therefore, this second evaluation methodology is heavily interconnected with the evaluation scenario, as we establish the actual truth behind the crimes that were committed, and which should be reflected on the data. The architecture of this evaluation approach can be seen in Figure 34.

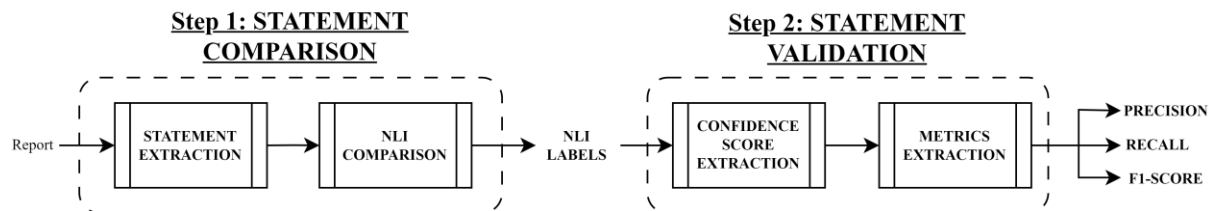


Figure 34. Fact-based evaluation framework

The fact-based evaluation is aimed at determining the relationship between the truth and the statements made in the report, to ensure that there is an entailment relationship between them, rather than a contradiction that signifies misinformation present in the report. This process is carried out through the implementation of two sequential steps: the statement comparison step and the statement evaluation step.

STATEMENT COMPARISON STEP

Natural Language Inference (NLI) is one of the main tasks in the field of natural language processing (NLP). This task is aimed at establishing the logical relationship between statements. Thus, given a reference statement, which is often referred to as the 'premise', this statement is compared to a statement to be verified, often referred to as 'hypothesis'. The idea behind NLI is to categorize the relationship of the hypothesis regarding the premise. More specifically, NLI classifies the hypothesis into three different categories, each with an associated level of confidence regarding how sure the model is that the hypothesis falls under the label. The three category types can be defined as follows:

- Entailment: The premise guarantees the hypothesis, marking it as true under the premise.
- Contradiction: The premise denies the hypothesis, marking it as false under the premise.
- Neutral: The hypothesis cannot be concluded as true or false under the premise.

Taking this approach into account, we can consider the following. If the crimes committed by the suspect are known, that knowledge can be established as the premise, and the conclusion reached by the framework as hypothesis of what could have happened. Thus, if a certain conclusion reached by the framework is entailed by any of the established premises (built from ground truth statements) with a certain level of confidence, then the conclusion can be regarded as truthful. Considering this, the main goal of this comparison statement is to extract the alignment label between each of the statements done in the conclusion report with an established ground truth of what occurs in the crime.

However, NLI models present a big limitation related to their strictness. The way in which these models establish the relationship category between premises and hypothesis is often extreme and inclined toward contradiction detection rather than entailment, where an irrelevant or small contradiction returns high contradiction scores, and partial or approximated entailments return low entailment scores. This sometimes translates into statements with similar theme and intent being classified as mostly contradictory due to differences in wording or detail. Given that our intent is to compare established statements with statements gathered from a bigger text, which often have vaguer and more stretched out wording for better reading comprehension, simply using these NLI could cause a big limitation in the proposed evaluation paradigm, as most statements could be regarded as contradictory due to their structural differences rather than their semantic ones. To tackle this issue, three measures are taken, two of them implemented in this first statement comparison step, and the remaining one in the next statement evaluation step.

The first measure is regarding the NLI model selected for this task, which is the ‘cross-encoder/nli-deberta-v3-base model’. While this model is lesser known than other alternatives (BERT, RoBERTa, facebook/bart-large-mnli, etc.), it is of special interest to our case due to its nature when evaluating the relationship between statements. Firstly, since it implements a cross-encoder approach, it identifies more subtle relationships between the premise and the hypothesis, making the model suffer less from word structure bias in the relationship analysis. Furthermore, the backbone of this model (DeBERTa v3) uses disentangled attention to better understand the context in the statements. Finally, it can be easily implemented locally and it has been specifically trained for semantic comparison in NLI tasks, providing, as mentioned, flexibility regarding differently phrased sentences that tackle the same topic.

Considering this NLI model, before carrying on with the statement comparison step, the second measure to tackle the NLI model strictness limitation is taken through the definition of the ground truth to consider as premises. Although the truth is established by the evaluation scenario, wording the truth statements as complete facts might cause trouble since the statements in the report are worded as possibilities. This, as explained, can push the model to directly consider them contradictions due to their differences in certainty. Therefore, the phrasing of the ground truth statements must be defined considering the expected phrasing in the report, rather than the straight facts. Moreover, since we desire for the reached conclusions to be closer to the actual truth as the investigation progresses, we expect the conclusions in earlier phases to mainly have a neutral stance regarding the truth statements. Nevertheless, the same general truths are established to evaluate the 3 phases as a way of also seeing how the framework progressively reaches better conclusions as more data is available. Thus, ground truth statements, considering the wording as previously said, are established for the general evaluation, without differentiating between phases. Those truths can be summarized as follows:

- *Truth statement I:* The criminal may have been involved in the theft of a laptop from a car on June 20th near the Tenderloin District in San Francisco.
- *Truth statement II:* The criminal may have been involved in the theft of a handbag containing a laptop, a phone, a set of keys, and cash on the 30th of June on a street near Union Square in San Francisco.
- *Truth statement III:* The criminal may have been involved in the theft of a handbag from a car containing a gold watch, a set of keys, and personal documents on the 4th of July on a street in the Tenderloin District in San Francisco.
- *Truth statement IV:* The criminal’s house might be near Hayes Valley in San Francisco, in the coordinates [37.77665921170173N, 122.42930492656163W].

- *Truth statement V:* The criminal might sell the stolen items at 16th St Mission subway station in the Mission District of San Francisco.
- *Truth statement VI:* The criminal might sometimes go to a restaurant in the shopping mall at 16th St in San Francisco, California, in the coordinates [37.76655660345612N, 122.40827817313549W].
- *Truth statement VII:* The criminal might sometimes go to shop to a supermarket at the Fillmore District in San Francisco, California, in the coordinates [37.77835570670887N, 122.42647155970583W].

Considering both initial measures, the statement comparison step starts by dividing the text into statements using the same approach as the previous evaluation framework, through a natural language toolkit. Once the statements are defined, the NLI model is implemented for each one of them and compared to all the defined ground truths. Moreover, for each of the three labels explained, a confidence score is associated with them, regarding the model's interpretation of the likeliness of the statement being under that label. Since one statement can be entailed to various ground truth statements, as well as neutral and contradictory, the mean of these confidence scores is computed, to determine the overall score given to each of the possible labels considering all the ground truth. Therefore, the output of this first step is a confidence level associated to each of the three possible labels associated to each of the statements extracted from the text.

STATEMENT EVALUATION STEP

A statement that is truthful is expected to be entailed with at least one ground truth and to be neutral with most of the remaining. On the other hand, a statement that is entailed with some ground truth statements but is directly contradicting others is not considered useful, as it is not desired for the report to contain contradictory statements. Therefore, both the entailment and the contradiction must be considered to properly evaluate the report and the statements made in it. This final statement evaluation step is aimed at extracting a series of metrics that help evaluate how aligned the statements in the final report are to the actual events. To carry out this evaluation step, two tasks are implemented: a confidence extraction task and a metric computation task.

The confidence extraction task computes the third remaining measure from the ones mentioned previously to tackle the NLI strictness issue. It does through in its computation of the confidence of truthfulness associated with each of the statements. As previously mentioned, it is important to consider both the entailment and contradiction scores, since a statement that aligns greatly with a true statement but also contradicts greatly another true statement showcases inconsistencies in the report, which might be a sign of false statements. However, only considering those two scores for the final confidence computation might limit the comparison insight due to the nature of NLI models previously defined. Although the model has been carefully selected and the premises carefully defined, to ensure that the comparison is not biased by structure, many sentences are still expected to have a neutral alignment with the ground truth statements. From the report, many of the statements will be the ones to give away the specific conclusions, but many others will simply connect the text or give information not regarding the ground truth. In those cases, both the entailment and contradiction scores will have low values, resulting in a low confidence score. It is crucial to consider that neutral statements do not directly signify a bad or misaligned text, and in our case it might signify the exact opposite. If a statement does not fully align with any of the ground truths, but also does not directly contradict any of them, then it does not necessarily contribute to the text's untruthfulness, but rather to its broader content outside of the established required truths. This must be considered, because a text with many entailed statements which also contains many neutral ones should have the same validity as a text with many entailed statements and fewer neutral ones, as in this evaluation we are only considering a small percentage of the total amount of information contained in the reports.

Thus, for the computation of the confidence of truthfulness value, all the labels are considered, including the entailment label, the contradiction label, and the neutrality label. The value itself is computed through a linear weighted combination of NLI scores. Firstly, the contradiction score is subtracted from the entailment score, acting as a kind of penalty. A weight is associated to this new entailment score adjusted for contradiction, which weights the relevance of the score to the final

truthiness confidence level. Secondly, the neutrality score is considered and added to this value with an additional associated weight, in this case to establish the relevance of neutral statements in the final truthiness confidence level. If the report has statements highly aligned with the truth (high entailment score), does not contradict any true statements (low contradiction score), and has a remaining number of neutral statements, then the truthiness confidence level will be high and vice versa. This truthiness confidence is computed for each of the statements.

Finally, the confidence is evaluated using numeric metrics to properly evaluate the truthiness of the overall report. This confidence is evaluated considering the overall confidence of all the statements in the text, to get an overall truthiness score of the report regarding the premise statements. Considering the confidence level, a series of metrics are computed using the same threshold considered in the previous content-based evaluation approach, 0.5, to determine the point in which it is considered that a statement is true or false. Thus, a classification-based approach is carried out again, with statements classified a true through the threshold taking the value of 1 and the ones classified as false taking the value of 0. Therefore, the metrics used are the same as the ones previously presented in Table 11, in this case defined in the following Table 15:

Table 15. Metrics to carry out the fact-based evaluation

METRIC	USE CASE DESCRIPTION	TARGET VALUE
Precision	The precision metric defines the proportion of statements that align with the truth.	A value close to 1 means that almost all the statements included in the report are aligned with the truth.
Recall	The recall metric defines the proportion of truths that are included in the report, meaning, to how many of the true conclusions does the report arrive to.	A value close to 1 means that almost all the truth statements are included in the report.
F1-Score	The F1-score metric defines the harmonic mean between the precision and recall values, meaning that it summarizes the number of true statements in the report and the completeness of the conclusions reached.	A value close to 1 means there is a good balance between having true statements and including all the necessary conclusions.

8.3.EVALUATION OF RESULTS OBTAINED

As mentioned previously, the evaluation of the proposed framework is carried out through the evaluation of the generated reports at three stages of the investigation, as to evaluate the precision of these reports as more data is uncovered. Each of the reports generated for each of the phases is evaluated for its content and its alignment with the truth.

8.3.1. CONTENT-BASED RESULTS

The reports are first evaluated for their content, ensuring that they contain all the relevant topics for the police to be able to draw useful conclusions. The results obtained for each of the reports regarding the evaluation metrics previously explained are summarized in Table 16.

Table 16. Content-based report evaluation results

	PRECISION	RECALL	F1-SCORE
Phase I report	0.944	1.0	0.971
Phase II report	0.952	0.833	0.889
Phase III report	0.952	0.833	0.889

Regarding the content of the reports, although the metrics indicate a good performance by the system, some conclusions of interest can be extracted. Firstly, regarding the report with the conclusions from the first phase of the evaluation, this phase obtained the best results as the requirements are less than with other phases due to the lack of information expected at this point in the investigation. Nevertheless, even with less requirements, the phase I report obtains a high prediction value and an even higher recall, indicating that most of the detected topics were within the required ones, and that all the required topics were included in the report. This indicates that the information that is required from this phase report, which mainly include information regarding the behavioral patterns of the suspect, is complete. This evaluates the contextualization of this data by the framework, as from very early ages in the investigation, enough additional information is integrated with the conclusions from the behavioral analysis and anomaly detection for a significant enough amount of content to be included in a report with only 10 days of data. This is also interesting in regards to the data generation and movement prediction module for two reasons. Firstly, with only 10 days of data, most of the topic requirements are met, meaning that most of the needed simulation parameters could probably be defined. This indicates that predictions can be made with few available data. Secondly, since most of the topic requirements are met, the predicted data directly integrated back to the system can still give useful insight even if only a few days are predicted.

Furthermore, both the second and third phase reports deal with more strict requirements regarding the topics to be implemented. Nevertheless, the prediction and recall results are still good. In this case, the recall is slightly lower, which can be due to several things, with the main one being that not all the anomalies might have been detected because the suspect might have not deviated in all the relevant ways, which might cause some of the anomaly related topics to not be as present in the report. Even so, the results showcase a good performance, which includes the needed topics even on a more complete use case, where the behavioral data, the anomalous data, and the crime related data had to be all correlated. This exemplifies the framework's ability to not only properly contextualize all of them, but also to draw relevant connections between them, especially when associating reported crimes with some of the anomalies detected.

8.3.2. FACT-BASED RESULTS

The reports are secondly evaluated regarding the alignment they have to the truth. Although specific content requirements are defined for each report from each phase in the content-based evaluation, for the fact-based evaluation, as previously explained, the truth is fixed and equal for all the phases. Therefore, for earlier stages, we expect the values to be slightly worse than for later stages. This is evaluated by the results obtained, which are summarized in Table 17.

Table 17. Fact-based report evaluation results

	PRECISION	RECALL	F1-SCORE
Phase I report	0.86	0.787	0.822
Phase II report	0.96	0.889	0.923
Phase III report	0.993	0.9	0.944

As expected, the precision and recall values improve as the investigation progresses, as more information is included in the reports and the conclusions reached are closer to the actual truth. Nevertheless, the results obtained, even for the first phase report, are not bad. In the first phase, since some of the conclusions are not close to being reached, like the ones regarding the specific crimes, the recall value is slightly lower, as not all the required truths are included in the report. However, due to the precision value, it seems like the truths that could be extracted in this first phase, which included those regarding the behavioral patterns (truth statements IV to VII), are included in the report, as the text seems aligned with the premises declared. This evaluates the framework in many ways, starting with the fact that even with few days of data, the framework still analyzes data in meaningful ways, enough for truthful conclusions to be reached even in earlier stages of the investigation. This also

evaluates the prediction as well, as even with few input data, the framework can integrate truthful and useful information into well-founded predictions.

Furthermore, as mentioned, as the investigation progresses, the reports align further with the truth, with the phase II report already showcasing an increase in both the prediction and recall values regarding the previous report, and the phase III report showing the best results, with the most alignment with the expected conclusions. Both are interesting, as they show that the framework can associate anomalies and crimes and speculate possible connections between them that point into the right direction. They also show that, even with more data added, the initial conclusions regarding the behavioral patterns are maintained, demonstrating a robust and consistent though process by the framework.

9. CONCLUSIONS AND FUTURE WORKS

9.1. CONCLUSIONS

The exploration and analysis of georeferenced data have become a long-standing research topic in recent years due to its increased relevance because of the equal increase of location-based services. In the work conducted for the Bachelor's thesis, an initial exploration on how this data could be leveraged in security contexts was carried out through the design and implementation of a behavioral analysis and anomaly detection system. This system analyzed georeferenced data through two of the most widely adopted processing approaches: behavior pattern extraction and modeling, and the subsequent anomaly detection. Starting with behavioral modeling, the developed system aimed at uncovering the underlying behavioral patterns in the data to build a model that could explain the movements included in it, providing a tool for raw georeferenced data understanding. As for anomaly detection, this was an approach enabled by the previous behavioral model, as it was implemented in the developed system to detect deviations from the established behavior that could be of interest in security contexts.

In this project, based on the detected limitations from the previous work carried out, mainly the reduced interpretability and transparency of the results, the simultaneous increase of language-based applications has been considered as a possible way to tackle the detected weaknesses. With the recent growth in system complexity, many models nowadays suffer from the black-box problem, which reduces transparency, understanding, and interpretability of results. Many language-based applications aim at tackling this issue but developing systems for text generation and data integration by using large language models (LLMs). Many standalone implementations of these models enable for a better understanding of processing results by human users, but in recent years, LLMs have additionally started to be integrated into more complex structures, like RAG architectures for context retrieval and AI agentic systems for task automatization, which allow to build more complex and transparent systems that better analyze and interpret data.

This work proposes an integrated and complete framework for crime analysis and suspect behavioral modeling through the integration of the previously developed system for behavioral analysis and anomaly detection with additional processing modules implementing language-based approaches to better leverage all the available data for better integration, contextualization and analysis of the detected patterns and anomalies of interest, as well as to better present the data the human user, with practicality and comprehensibility in mind, to enhance decision-making and prediction tasks.

To define the final proposed solution, multiple individual processing modules have been designed and implemented to carry out specific tasks that are later integrated in the major and general framework workflow. Starting with a main data contextualization and interpretation module, this project's work is built around this module, as it carries out the main tasks of integrating the data sources, contextualizing them with relevant information, interpreting them considering the security context, and presenting them to the user in a structured way. This is done through the integration of RAG-based architecture with AI agentic systems. Firstly, the available data sources (movement data, behavioral model, anomalies detected, and previous police reports) are analyzed and contextualized by using an automatized context search agentic system. Then, they are integrated as contextual documents in a retrieval-augmented generation structure, in which, given a prompt by the user, the relevant domain-specific information is extracted and given to a language model which interprets it and structures the conclusions accordingly. For the implementation of this module, an additional police report simulation module is designed and implemented, which combines language models with data processing algorithms to generate realistic and context-aware reports that simulate real police reports. Finally, an additional data generation and movement prediction module is designed to further enhance decision-making by predicting the possible future movements of a suspect based on the conclusions reached on the first data contextualization and interpretation module. This is carried out through the integration of language models for text analysis and generation with an existing movement data simulator developed with the previously mentioned behavioral analysis and anomaly detection system as part of the Bachelor's thesis work. All the developed modules are integrated with

the previously defined system on a unified framework aimed at georeferenced data analysis and contextualization for crime investigation.

Two additional evaluation frameworks have been defined to tackle the existing limitation in language-based validation methodologies, which include a lack of standardized metrics and algorithms to properly evaluate the performance of language-based systems. These frameworks are aimed at evaluating the generated reports based on their included context, and their veracity, to ensure that the information given to the user is trustworthy and complete. The context-based evaluation has demonstrated that the generated reports contain all the relevant topics needed for the user to draw relevant and informed conclusions due to the inclusion and mention of all the information gathered at various stages of the investigation. Furthermore, the fact-based evaluation demonstrated that the framework does gradually reach the correct conclusions as more information is uncovered, with the latest phases of the investigation obtaining conclusions well-aligned with the truth, which give a good picture of the actual crimes that have occurred and their possible connection with certain suspects.

Given these results, this framework could be implemented as a supporting tool to aid law enforcement in georeferenced data and crime analysis tasks, due to the integration of relevant data source available to them and the interpretation and structuring of the correlations found and the conclusions reached by the framework.

With the work carried out in this project, and the results obtained, the goals proposed for this Master's thesis have been fulfilled, with the main contributions summarized as follows:

- The integration of an existing behavioral analysis and anomaly detection system, as well as a movement data simulator, into a complete framework for better interpretation and contextualization using domain-specific knowledge to leverage the reached conclusions by using emerging language-based systems and architectures.
- The implementation of a complete data contextualization and interpretation module which properly leverages external data sources and internal contextual information to give meaning to raw movement data, behavioral patterns, and behavioral deviations in the context of a police investigation. The module implements rising approaches, like AI agentic architectures for external data extraction, and RAG architectures for contextualized prompts. The module additionally structures the conclusions reached by the data integration and interpretation processes implemented for better understanding by the human user.
- The generation of a police report simulation which considers real crimes, as well as movement and behavioral patterns, to generate data-specific police reports that can be correlated back to the movement data by the modules aimed at data correlation and interpretation implemented in the framework. The police reports are structured following usual information contained in real reports.
- The implementation of a data generation and movement prediction module which enables the user to leverage the conclusions reached by the framework to predict possible future movements and crimes for its implementation in decision-making. The movement prediction module properly integrates new language-based models with previously developed movement simulators to obtained the required results, which are directly inputted back into the system for proper interpretation.
- The validation of the proposed framework through the design of novel evaluation frameworks aimed at evaluating the results based on the relevant characteristics in the security context of this work, including the proper and complete content of the report, and the true and correct conclusions included in the reports.

9.2.FUTURE WORKS

Due to the evolving nature of language-based solutions, many challenges are still present in the literature landscape of these approaches. Starting with the data contextualization and interpretation module, although the results are promising, it is desired to implement more advanced structures, with a fine-tuned RAG architecture, as well as more complex agentic systems. The desire for the fine-tuned RAG is to implement models, both the retrieval and the generator models, that have been specifically trained for security-based tasks, so that the retrieval knows what relevant information to look for

when retrieving specific documents, and the generator can both respond to questions in the chatbot and generate reports with a crime context background. In the future, a more defined template for the generated reports is also desired. Furthermore, as for the more complex agent systems, the desire is to define a more robust agents which can learn and perform tasks by themselves, without the need of multiple instruction specifications, following a learning agent structure.

It is also desired to improve the developed movement simulator module, to better predict possible future movements with associated probabilities and even the consideration of anomalies to properly predict not only future movements, but also future possible crimes. To do so, a possible fine-tune of the implemented LLM is considered, to better extract the relevant information from the sources to define the needed parameters, as well as a more refined structure around the simulator, in which the parameters are not directly inputted from the report analysis component, but an additional process is carried out to add logic to the simulations, with associated characteristics, anomalies, and probabilities of occurrence associated to each one.

It is also desired to define a more robust validation approach to properly evaluate the framework's performance, as well as the individual performance of each of the modules that make it up, for a better understanding of the processes being carried out, and the limitations they present, with the idea of properly tackling those limitations in the future as well. To do so, an improvement of the current frameworks is desired for a more robust analysis of the text, as well as the design of further frameworks that validate different aspects not considered in this work. Furthermore, as mentioned, the design of new frameworks aimed at validating other component's performance directly (and not through the final output) is also interesting, especially for the RAG architecture component and the AI agentic system, which have limited validation approaches in the literature.

The main idea behind the proposed framework is to flesh it out into a complete tool in the future that could be used by police forces in real use cases.

10. BIBLIOGRAPHY

- [1] R. L. Emmet, . K. Hunt, R. Jennings, K. Daniel and D. A. Abreu, "Evaluating a Method for Georeferencing Agricultural Fields," *Journal of Data Science*, vol. 22, no. 3, p. 423–435, 2024.
- [2] M. F. F. de Oliveira, P. H. Melo Albuquerque, Y. H. Peng and P. A. Henrique, "Mapping regional business opportunities using geomarketing and machine learning," *Gestão & Produção*, vol. 27, 2020.
- [3] S. Bertocci, F. Cioli and A. Cottini, "Unlocking cultural heritage: leveraging georeferenced tools and open data for enhanced cultural tourism experiences.," in *Proceedings of the 20th International Conference on Culture and Computer Science: Code and Materiality*, 2023.
- [4] N. Monir, A. R. Abdul Rasam, R. Ghazali, H. F. Suhandri and A. Cahyono, "Address Geocoding Services in Geospatial-based Epidemiological Analysis: A Comparative Reliability for Domestic Disease Mapping.," *International Journal of Geoinformatics*, vol. 17, no. 5, 2021.
- [5] S. Felício, J. Hora, M. C. Ferreira, D. Abrantes, P. D. Costa, C. Dangelo, J. Silva and T. Galvão, "Handling OpenStreetMap georeferenced data for route planning," *Transportation Research Procedia*, vol. 62, pp. 189-196, 2022.
- [6] L. Yin, J. Chen, H. Zhang, Z. Yang, Q. Wan, L. Ning, J. Hu and Q. Yu, "Improving emergency evacuation planning with mobile phone location data," *Environment and Planning B: Urban Analytics and City Science*, vol. 47, no. 6, pp. 964-980, 2020.
- [7] Z. Chu, H. Guo, X. Zhou, Y. Wang, F. Yu, H. Chen, W. Xu, X. Lu, Q. Cui, L. Li, J. Zhou and S. Li, "Data-centric financial large language models," *arXiv*, 2023.
- [8] N. Agarwal, D. Thakur, K. Krishna, D. P. Goel and D. S. P. Singh, "LLMS for Data Analysis and Client Interaction in MedTech," 2021.
- [9] Y. Zhang, M.-P. Kwan and L. Fang, "An LLM driven dataset on the spatiotemporal distributions of street and neighborhood crime in China," *Scientific Data*, vol. 12, no. 1, p. 467, 2025.
- [10] R. Chivoreanu, A. Cosma, A. Catruna, R. Rughinis and E. Radoi, "Aligning actions and walking to LLM-generated textual descriptions," in *2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*, 2024.
- [11] S. A. Imran, M. N. H. Khan, S. Biswas and B. Islam, "Llasa: Large multimodal agent for human activity analysis through wearable sensors," *arXiv*, vol. 3, no. 4, 2024.
- [12] J. Feng, Y. Du, J. Zhao and Y. Li, "Agentmove: Predicting human mobility anywhere using large language model based agentic framework," *arXiv*, 2024.
- [13] K. Meduri, H. Gonaygunta, G. S. Nadella, P. P. Pawar and D. Kumar, "Adaptive intelligence: Gpt-powered language models for dynamic responses to emerging healthcare challenges," *IJARCCCE*, vol. 13, pp. 104-109, 2024.
- [14] H. Jung, Y. Kim, H. Choi, H. Seo, M. Kim, J. Han, G. Kee, S. Park, S. Ko, B. Kim, S. Kim, T. J. Jun and Y.-H. Kim, "Enhancing clinical efficiency through llm: Discharge note generation for cardiac patients," *arXiv*, 2024.
- [15] R. E. Atencio-González, D. E. Bonilla-Ron, M. V. Miles-Flores and S. Á. López-Zavala, "Chat GPT como Recurso para el Aprendizaje del Pensamiento Crítico en Estudiantes Universitarios," *Cienciamatria. Revista Interdisciplinaria de Humanidades, Educación, Ciencia y Tecnología*,

vol. 9, no. 17, pp. 36-44, 2023.

- [16] M. Imran and N. Almusharraf, "Google Gemini as a next generation AI educational tool: a review of emerging educational technology," *Smart Learning Environments*, vol. 11, no. 1, p. 22, 2024.
- [17] E. Halford and A. Webster, "Using chat GPT to evaluate police threats, risk and harm," *International Journal of Law, Crime and Justice*, vol. 78, 2024.
- [18] X. Xing and P. Chen, "Entity extraction of key elements in 110 police reports based on large language models," *Applied Sciences*, vol. 14, no. 17, 2024.
- [19] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," 2018.
- [20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1-67, 2020.
- [21] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," *arXiv*, 2019.
- [22] Z. Fu, W. Lam, Q. Yu, A. M.-C. So, S. Hu, Z. Liu and N. Collier, "Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder," *arXiv*, 2023.
- [23] T. Wang, A. Roberts, D. Hesslow, T. Le Scao, H. W. Chung, I. Beltagy, J. Launay and C. Raffel, "What language model architecture and pretraining objective works best for zero-shot generalization?," in *International Conference on Machine Learning*, 2022.
- [24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019.
- [25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems 33*, 2020.
- [26] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altschmidt, S. Altman and S. Anadkat, "Gpt-4 technical report," *arXiv*, 2023.
- [27] C. E. Haupt and M. Marks, "AI-generated medical advice—GPT and beyond," *Jama*, vol. 329, no. 16, pp. 1349-1350, 2023.
- [28] R. Khamidullin, "Forensic Support for the Use of Artificial Intelligence Technology in the Detection and Investigation of Crimes," *Russ. Jurid. J. Elec. Supp.*, p. 19, 2024.
- [29] G. G. T. R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth and K. Millican, "Gemini: a family of highly capable multimodal models," *arXiv*, 2023.
- [30] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker and Y. Du, "Lamda: Language models for dialog applications," *arXiv*, 2022.
- [31] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton and S. Gehrmann, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1-113, 2023.

- [32] L. A. Kadiyala, O. Mermer, D. J. Samuel, Y. Sermet and I. Demir, "The implementation of multimodal large language models for hydrological applications: A comparative study of GPT-4 vision, gemini, LLaVa, and multimodal-GPT," *Hydrology*, vol. 11, no. 9, p. 148, 2024.
- [33] D. J. Samuel, I. Skarga-Bandurova, D. Sikolia and M. Awais, "AgroLLM: Connecting Farmers and Agricultural Practices through Large Language Models for Enhanced Knowledge Transfer and Practical Application," *arXiv*, 2025.
- [34] K. Saab, T. Tu, W.-H. Weng, R. Tanno, D. Stutz, E. Wulczyn, F. Zhang, T. Strother, C. Park and E. Vedadi, "Capabilities of gemini models in medicine," *arXiv*, 2024.
- [35] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro and F. Azhar, "Llama: Open and efficient foundation language models," *arXiv*, 2023.
- [36] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang and A. Fan, "The llama 3 herd of models," *arXiv*, 2024.
- [37] M. A. "The llama 4 herd: The beginning of a new era of natively multimodal ai innovation," 5 April 2025. [Online]. Available: <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>.
- [38] C. Curto, D. Giordano, D. G. Indelicato and V. Patatu, "Can a Llama Be a Watchdog? Exploring Llama 3 and Code Llama for Static Application Security Testing," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2024.
- [39] K. I. Roumeliotis, N. D. Tselikas and D. K. Nasiopoulos, "LLMs in e-commerce: A comparative analysis of GPT and LLaMA models in product review evaluation," *Natural Language Processing Journal*, vol. 6, 2024.
- [40] N. Satterfield, P. Holbrook and T. Wilcox, "Fine-tuning Llama with Case Law Data to Improve Legal Domain Performance," *OSF Preprints*, 2024.
- [41] H. Wang, C. Liu, N. Xi, Z. Qiang, S. Zhao, B. Qin and T. Liu, "HuaTuo: Tuning LLaMA Model with Chinese Medical Knowledge," *arXiv*, 2023.
- [42] G. Michelet and F. Breitingner, "ChatGPT, Llama, can you write my report? An experiment on assisted digital forensics reports written using (local) large language models," *Forensic Science International: Digital Investigation*, vol. 48, 2024.
- [43] M. A. t. "Mistral 7B," 27 September 2023. [Online]. Available: <https://mistral.ai/news/announcing-mistral-7b>.
- [44] M. Ikonić Nešić, S. Petalinkar, M. Škorić, R. Stanković and B. Rujević, "Advancing Sentiment Analysis in Serbian Literature: A Zero and Few-Shot Learning Approach Using the Mistral Model," in *Proceedings of the Sixth International Conference on Computational Linguistics in Bulgaria (CLIB 2024)*, Sofia, Bulgaria, 2024.
- [45] Y. Jiang, Q. Chao, Y. Chen, X. Li, S. Liu and G. Cong, "UrbanLLM: Autonomous Urban Activity Planning and Management with Large Language Models," *arXiv*, 2024.
- [46] X. Guo, Q. Zhang, J. Jiang, M. Peng, H. F. Yang and M. Zhu, "Towards responsible and reliable traffic flow prediction with large language models," *SSRN*, 2024.
- [47] M. B. Mutanga, J. Msane, T. N. Mndaweni, B. B. Hlongwane and N. Z. Ngcobo, "Exploring the Impact of LLM Prompting on Students' Learning," *Trends in Higher Education*, vol. 4, no. 3, p. 31, 2025.
- [48] Q. Zhao and M. Zhang, "Elimination-based reasoning with LLM for multiple-choice educational question answering," *Journal of King Saud University Computer and Information*

Sciences, vol. 37, no. 7, pp. 1-17, 2025.

- [49] P. Sarzaeim, "Design and development of an LLM-based framework for crime classification and prediction," 2024.
- [50] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani and S. Brahma, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1-53, 2024.
- [51] D. Huang and Z. Wang, "Logical reasoning with LLMs via few-shot prompting and fine-tuning: a case study on turtle soup puzzles," in *2025 IEEE Symposium on Computational Intelligence in Natural Language Processing and Social Media (CI-NLPSoMe Companion)*, 2025.
- [52] R. S. de Padua, I. Qureshi and M. U. Karakaplan, "Gpt-3 models are few-shot financial reasoners," *arXiv*, 2023.
- [53] J. Bommarito, M. Bommarito, D. M. Katz and J. Katz, "GPT as knowledge worker: a zero-shot evaluation of (AI) CPA capabilities," *arXiv*, 2023.
- [54] N. Gruver, M. Finzi, S. Qiu and A. G. Wilson, "Large language models are zero-shot time series forecasters," *Advances in Neural Information Processing Systems*, vol. 36, pp. 19622-19635, 2023.
- [55] C. Beneduce, B. Lepri and M. Luca, "Large language models are zero-shot next location predictors," *IEEE Access*, 2025.
- [56] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han and F. Huang, "Qwen technical report," *arXiv*, 2023.
- [57] A. Mon, S. Fenollosa and J. Lecumberri, "From BERT to Qwen: Hate Detection across architectures," *arXiv*, 2025.
- [58] T.-G. Marchitan, C. Creanga and L. P. Dinu, "Few-shot text-based emotion detection," *arXiv*, 2025.
- [59] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824-24837, 2022.
- [60] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22199-22213, 2022.
- [61] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv*, 2022.
- [62] Z. Guan, Z. Wu, Z. Liu, D. Wu, H. Ren, Q. Li, X. Li and N. Liu, "Cohortgpt: An enhanced gpt for participant recruitment in clinical study," *arXiv*, 2023.
- [63] M. Zhang, S. R. Ahmed and J. H. Martin, "Ftg-cot at semeval-2024 task 9: Solving sentence puzzles using fine-tuned language models and zero-shot cot prompting," in *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, 2024.
- [64] M. Lu, F. Gao, X. Tang and L. Chen, "Analysis and prediction in SCR experiments using GPT-4 with an effective chain-of-thought prompting strategy," *Iscience*, vol. 27, no. 4, 2024.
- [65] Q. T. "Qwen2 technical report," *arXiv*, 2024.
- [66] X. Luo, F. Ding, Y. Song, X. Zhang and J. Loo, "Pkrd-cot: A unified chain-of-thought prompting for multi-modal large language models in autonomous driving," in *International*

Conference on Neural Information Processing, 2024.

- [67] S. Tahmid and S. Sarker, "Qwen2. 5-32b: Leveraging self-consistent tool-integrated reasoning for bengali mathematical olympiad problem solving," *arXiv*, 2024.
- [68] K. Guu, K. Lee, Z. Tung, P. Pasupat and M.-W. Chang, "REALM: Retrieval-Augmented Language Model Pre-Training," *arXiv*, 2020.
- [69] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih and T. Rocktäschel, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459-9474, 2020.
- [70] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv*, 2019.
- [71] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen and W.-t. Yih, "Dense Passage Retrieval for Open-Domain Question Answering.," *EMNLP*, pp. 6769-6781, 2020.
- [72] J. P. Nayinzira and M. Adda, "SentimentCareBot: Retrieval-augmented generation chatbot for mental health support with sentiment analysis," *Procedia Computer Science*, vol. 251, pp. 334-341, Procedia Computer Science.
- [73] A. Banerjee, A. Satish and W. Wörndl, "Enhancing tourism recommender systems for sustainable city trips using retrieval-augmented generation," in *International Workshop on Recommender Systems for Sustainability and Social Good*, 2024.
- [74] K. K. Kemboi, "Applying Retrieval Augmented Generation (Rag) With Llm-based Chatbot to Enhance Patient-facing Health Information Tools," 2024.
- [75] M. Raja and E. Yuvaraajan, "A rag-based medical assistant especially for infectious diseases," in *2024 International Conference on Inventive Computation Technologies (ICICT)*, 2024.
- [76] J. Yang, X. Dai, Y. Lv, L. Kovacs and F.-Y. Wang, "TransRAG for parallel transportation: toward reliable and trustworthy transportation systems via retrieval-augmented generation," *Frontiers of Information Technology & Electronic Engineering*, vol. 26, no. 1, pp. 20-26, 2025.
- [77] N. S. Amarnath and R. Nagarajan, "An Intelligent Retrieval Augmented Generation Chatbot for Contextually-Aware Conversations to Guide High School Students," in *2024 4th International Conference on Sustainable Expert Systems (ICSES)*, 2024.
- [78] Z. Liu, "AcademicRAG: Knowledge Graph Enhanced Retrieval-Augmented Generation for Academic Resource Discovery," 2025.
- [79] J. Fang, Z. Meng and C. Macdonald, "TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation," *arXiv*, 2024.
- [80] K. Rangan and Y. Yin, "A fine-tuning enhanced RAG system with quantized influence measure as AI judge," *Scientific Reports*, vol. 14, no. 1, 2024.
- [81] X. Chen, L. Wang, W. Wu, Q. Tang and Y. Liu, "Honest AI: Fine-Tuning" Small" Language Models to Say" I Don't Know", and Reducing Hallucination in RAG," *arXiv*, 2024.
- [82] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," *arXiv*, 2024.
- [83] C.-W. Sung, Y.-K. Lee and Y.-T. Tsai, "A new pipeline for generating instruction dataset via RAG and self fine-tuning," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2024.

- [84] P. Béchard and O. M. Ayala, "Multi-task retriever fine-tuning for domain-specific and efficient RAG," *arXiv*, 2025.
- [85] T. Zhang, Z. Jiang, S. Bai, T. Zhang, L. Lin, Y. Liu and J. Ren, "RAG4ITOps: A supervised fine-tunable and comprehensive RAG framework for IT operations and maintenance," *arXiv*, 2024.
- [86] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115-152, 1995.
- [87] N. J. Nilsson, "Shakey the robot," 1984.
- [88] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Hoboken, NJ, USA: Pearson, 2020.
- [89] D. Andre and S. Russell, "Programmable reinforcement learning agents," *Advances in neural information processing systems*, vol. 13, 2000.
- [90] E. Gelenbe, E. Seref and Z. Xu, "Simulation with learning agents," *Proceedings of the IEEE*, vol. 89, no. 2, pp. 148-157, 2001.
- [91] P. G. Balaji, X. German and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177-188, 2010.
- [92] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv*, 2016.
- [93] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023.
- [94] Z. Q. Shen, R. Gay and X. Tao, "Goal-based intelligent agents," *International Journal of Information Technology*, vol. 9, no. 1, pp. 19-30, 2003.
- [95] Z. Yao, X. Li, B. Lang and M. C. Chuah, "Goal-lbp: Goal-based local behavior guided trajectory prediction for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 6770-6779, 2023.
- [96] P. Rizzo, "Goal-based personalities and social behaviors in believable agents," *Applied artificial intelligence*, vol. 13, no. 3, pp. 239-271, 1999.
- [97] F. Tung, J. S. Zelek and D. A. Clausi, "Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance," *Image and Vision Computing*, vol. 29, no. 4, pp. 230-240, 2011.
- [98] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *arXiv*, 2018.
- [99] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy and S. Bowman, "SuperGlue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.
- [100] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv*, 2020.
- [101] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu and A. Kumar, "Holistic evaluation of language models," *arXiv*, 2022.
- [102] A. Srivastava, A. Rastogi, A. Rao, A. A. Shueb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta and A. Garriga-Alonso, "Beyond the imitation game: Quantifying and extrapolating the

- capabilities of language models," *Transactions on machine learning research*, 2023.
- [103] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002.
 - [104] C.-Y. Lin, Rouge: A package for automatic evaluation of summaries, 2004.
 - [105] S. Banerjee and A. Lavie, METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, 2005.
 - [106] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv*, 2019.
 - [107] K. J. N. G. R. Reddy, "Novel Approach to Validate the Content Generated by LLM," 2025.
 - [108] M. Estevez, N. Singh, L. Dyson, B. Adamson, Q. Yuan, M. W. Hildner, E. Fidyk, O. Mbah, F. Khan and K. Seidl-Rathkopf, "Ensuring Reliability of Curated EHR-Derived Data: The Validation of Accuracy for LLM/ML-Extracted Information and Data (VALID) Framework," *arXiv*, 2025.
 - [109] M. Kuchnik, V. Smith and G. Amvrosiadis, "Validating large language models with relm," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 457-476, 2023.
 - [110] S. Es, J. James, L. E. Anke and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024.
 - [111] C. D. Manning, Introduction to information retrieval, Syngress Publishing, 2008.
 - [112] E. M. Voorhees, The trec-8 question answering track report., 1999.
 - [113] R. Pradeep, N. Thakur, S. Upadhyay, D. Campos, N. Craswell, I. Soboroff, H. T. Dang and J. Lin, "The great nugget recall: Automating fact extraction and rag evaluation with large language models," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.
 - [114] E. M. Voorhees, "Overview of TREC 2003.," in *Trec*, 2003.
 - [115] S. Simon, A. Mailach, J. Dorn and N. Siegmund, "A methodology for evaluating rag systems: A case study on configuration dependency validation," *arXiv*, 2024.
 - [116] R. Friel, M. Belyi and A. Sanyal, "Ragbench: Explainable benchmark for retrieval-augmented generation systems," *arXiv*, 2024.
 - [117] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men and K. Yang, "Agentbench: Evaluating llms as agents," *arXiv*, 2023.
 - [118] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk and D. Fried, "Webarena: A realistic web environment for building autonomous agents," *arXiv*, 2023.
 - [119] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso and R. Salakhutdinov, "Minerl: A large-scale dataset of minecraft demonstrations," *arXiv*, 2019.
 - [120] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou and J. Schrittwieser, "Starcraft ii: A new challenge for reinforcement learning," *arXiv*, 2017.
 - [121] K. Wei, S. Guth, G. Wu and P. Paskov, "Methodological Challenges in Agentic Evaluations of

- AI Systems," in *ICML Workshop on Technical AI Governance (TAIG)*, 2025.
- [122] D. J. Jilk, Limits to verification and validation of agentic behavior, Chapman and Hall/CRC, 2018.
- [123] Z. Fu, Z. Tian, Y. Xu and C. Qiao, "A two-step clustering approach to extract locations from individual GPS trajectory data," *ISPRS International Journal of Geo-Information*, vol. 5, no. 10, p. 166, 2016.
- [124] P. Brantingham and P. Brantingham, Crime pattern theory, Willan, 2013.
- [125] L. E. Cohen and M. Felson, Routine activity theory: A routine activity approach, Routledge, 2015.
- [126] D. Weisburd, The law of crime concentration and the criminology of place, Wiley Online Library, 2015.
- [127] Y. Ma, C. Xiao, C. Yuan, S. N. Van Der Veer, L. Hassan, C. Lin and G. Nenadic, "CAST: Corpus-Aware Self-similarity Enhanced Topic modelling," *arXiv*, 2024.

APPENDIX A: ETHICAL, ECONOMIC, SOCIETAL AND ENVIRONMENTAL ASPECT

A.1 INTRODUCTION

As mentioned throughout this work, the main goal behind the development of the proposed framework is for it to be implemented in a security-context, specifically as tool to aid law enforcement in crime investigation tasks. However, due to the sensitive nature of both the data used for this framework, as well as its implementation in security and crime analysis applications, a series of legal and ethical impacts and limitations need to be carefully considered and reviewed, to ensure that the proposed tool does not endanger the integrity and privacy of any human being and can be implemented in an ethical and legal way. Firstly, regarding the nature of the data, even though the data is not considered sensitive according to Article 9 of the European Union's General Data Protection Regulation (GDPR), georeferenced and location data of individuals is considered personal data by the same regulations, so the treatment of it might align with the personal data treatment principles established by the GDPR to ensure legal and ethical compliance. Secondly, regarding the nature of the context in which this tool is implemented, law enforcement authorities are subject to specific conditions regarding the use of personal data under the GDPR, as well as the use of AI systems that are under the label of 'high-risk' under the European's Union AI Act. Therefore, those specific conditions had to also be considered in the development of this framework to also ensure compliance in the framework's expected implementation.

Aside from the legal and ethical impacts surrounding this framework, it is important to consider its main goal. The framework is developed as a tool to improve and refine the current existing data processing tools available to law enforcement, to allow for a more efficient data processing, and a more enhanced data interpretation, to improve the tasks of decision-making and crime prediction. Therefore, this framework seeks to aid law enforcement for a more specialized, integrated, and evidence-based information handling to support police investigative work. It is important to also consider the social, environmental, and labor-based impacts a framework like this could have.

This first appendix focuses on a study of the main impacts of this framework in various contexts, including social, economic, and environmental contexts. It also carried out an in-depth study of the legal and ethical requirements of the framework due to its contextual nature, and how these requirements are satisfied.

A.2 DESCRIPTION OF RELEVANT IMPACTS RELATED TO THE PROJECT

An analysis of the main impacts of the work presented in this Master's thesis has been carried, identifying in each case the opportunities that it could provide in the various possible impacted contexts, and the risks that it presents. This analysis can be seen summarized in Table 18.

Table 18. Analysis of the impacts of the developed framework in social, ethical, legal, and environments contexts

IMPACT TYPE	OPPORTUNITIES	RISKS
Labor and working conditions	Complementation and enhancement of existing suspect and crime analysis tasks, facilitating decision-making and data interpretation to efficiently carry out law enforcement operations.	Challenges in scaling computational resources due to the use of heavy models.
Labor and working conditions	Reduction of analysis and computation time in pattern analysis, anomaly detection, and data integration and	Possible delays when dealing with large amounts of contextual data.

	correlation tasks, which improves satisfaction by reducing tedious tasks and manual labor.	
Social	Contributions to the understanding of criminal behaviors and correlations. Acceleration of investigative processes and other police operations.	Possible prolonged operation and processing times due to issues with components in the overall framework.
Ethical and legal	The data analysis processes implemented are free from discriminatory bias due to the lack of sensitive information handling, reducing the bias in police investigation and contributing to societal equity.	Private location data is used by an automated high-risk AI framework, which could lead to an unintended breach in the privacy and rights of certain individuals.
Ethical and legal	The conclusions reached by the framework are only build on true and verified information from trustworthy sources (law enforcement archives), avoiding the unintended correlation of the crimes with irrelevant or unrelated data that might cause bias in the investigation.	Data integration and correlations made by an automated high-risk AI framework could lead to unintended connections and false associations, which might cause obstacles in police work.
Environmental	The optimization approach of the developed systems might potentiate processing efficiency and encourage the development of productive green implementations.	Processing of large amount of data and context by computationally intensive models might leave a noticeable carbon footprint.
Economic	Reduction of costs associated with manual processing methods and time-consuming analysis approaches.	System maintenance and resources for its implementation are required and might be costly.

A.3 ETHICAL AND LEGAL REQUIREMENTS AND IMPACTS OF THE PROPOSED FRAMEWORK

As previously mentioned, due to the nature of the data used, the technical algorithms and models implemented, and the context of the framework's development, a series of legal and ethical requirements analysis must be carried to ensure compliance. More specifically, the developed framework makes use of personal movement data, automated AI systems and pretrained models, and is aimed for a security-based purpose. Thus, the legal and ethical constraints directly directed at the mentioned cases must be thoroughly analyzed to ensure that the needed practices are in motion to guarantee compliance with the legal and ethical directives and guidelines established by the European Union, as well as the engineering norms, standards, and responsibilities defined.

Starting with the required legal compliance, the main structures to consider for AI implementations developed within the European Union are the proposed EU Artificial Intelligence Act (EU AI act), and the General Data Protection Regulations (GDPR) imposed. The EU AI act, formally introduced in

2021 for its approval by the European Parliament and the Council of the EU, is the first proposal for AI regulation by the European Union. Its main goal is to establish the needed legal framework to ensure the safe, transparent, and responsible use and development of AI systems within the European Union. This act has a series of articles of interest for our use case, starting with Annex III, which refers to high-risk systems mentioned in Article 6(2), has specific points which discuss the definition of high-risk systems under law enforcement. It specifically established the systems that can be used if it is permitted under Union or national law, with the most relevant systems being defined as follows:

- *Annex 3(6)(c)* of the proposed European Commission Artificial Intelligence Regulation defined high-risk systems as AI systems which are intended to ‘be used by or on behalf of law enforcement authorities, or by Union institutions, bodies, offices, or agencies, in support of law enforcement authorities to evaluate the reliability of evidence during the investigation or prosecution of criminal offences’, which could apply to the integration and interpretation processes carried out by the framework.
- *Annex 3(6)(d)* of the proposed European Commission Artificial Intelligence Regulation defined high-risk systems as AI systems which are intended to ‘be used by law enforcement authorities or on their behalf or by Union institutions, bodies, offices or agencies in support of law enforcement authorities for assessing the risk of a natural person offending or re-offending not solely on the basis of the profiling of natural persons as referred to in Article 3(4) of Directive (EU) 2016/680, or to assess personality traits and characteristics or past criminal behavior of natural persons or groups’, which could also apply to various processes implemented by the framework, including the building of behavioral model, the assessment of a suspect past reported crimes, and the prediction of possible future movements and crimes.

However, considering Article 6 of the act, the following is mentioned, which exempts the developed framework from the high-risk label:

- *Article 6(3)(b)* of the proposed European Commission Artificial Intelligence Regulation establishes that a high-risk system defined in *Annex 3* shall not be considered as one if ‘the AI system is intended to improve the result of a previously completed human activity’, which could be considered the case, as many data integration and processing tasks in police investigations are carried out manually.
- *Article 6(3)(c)* of the proposed European Commission Artificial Intelligence Regulation establishes that a high-risk system defined in *Annex 3* shall not be considered as one if ‘the AI system is intended to detect decision-making patterns or deviations from prior decision-making patterns and is not meant to replace or influence the previously completed human assessment, without proper human review’, which is definitely the case of the proposed framework, as it is aimed to be an aid for the existing processes of data analysis, and not replace the human decision-making or influence it towards a biased outcome.

Therefore, considering that the framework, legally, is exempt from the high-risk label. Nevertheless, other articles like Article 5 must be considered, as it defines prohibited AI practices for all systems. More specifically, the articles that could cause trouble from the developed framework, and the reasoning behind the compliance with them, can be summarized as follows:

- *Article 5(1)(c)* of the proposed European Commission Artificial Intelligence Regulation establishes that the following is prohibited: the ‘placing on the market, the putting into service or the use of AI systems for the evaluation or classification of natural persons or groups of persons over a certain period based on their social behavior or known, inferred, or predicted personal or personality characteristics’. However, this does not directly affect to our framework, as it specifies that this practice is only prohibited is the social score of these inferred characteristics lead to ‘detrimental or unfavorable treatment of certain natural persons or groups of persons in social contexts that are unrelated to the contexts in which the data was originally generated or collected’, which does not affect our case as data is strictly gathered and used for security applications only, or if it leads to ‘detrimental or unfavorable treatment of certain natural persons or groups of persons that is unjustified or disproportionate to their

social behavior or its gravity’, which also does not apply given the context of the application of the framework.

- *Article 5(1)(d)* of the proposed European Commission Artificial Intelligence Regulation establishes that the following is prohibited: the ‘the placing on the market, the putting into service for this specific purpose, or the use of an AI system for making risk assessments of natural persons in order to assess or predict the risk of a natural person committing a criminal offence, based solely on the profiling of a natural person or on assessing their personality traits and characteristics’. However, it is also specified that ‘this prohibition shall not apply to AI systems used to support the human assessment of the involvement of a person in a criminal activity, which is already based on objective and verifiable facts directly linked to a criminal activity’, which is the case of the developed framework, as all the reached conclusions can be directly associated with true and verifiable data and the sole purpose of the framework is to support existing law enforcement processes.

Finally, the last article to consider by the AI act is Article 50, which refers to the transparency obligations of certain system which encapsulate the ones implemented in this framework. However, there is a compliance to the needed paragraphs as summarized:

- *Article 50(1)* of the proposed European Commission Artificial Intelligence Regulation establishes that ‘providers shall ensure that AI systems intended to interact directly with natural persons are designed and developed in such a way that the natural persons concerned are informed that they are interacting with an AI system, unless this is obvious from the point of view of a natural person who is reasonably well-informed, observant and circumspect, taking into account the circumstances and the context of use’. However, it also specifies that ‘this obligation shall not apply to AI systems authorized by law to detect, prevent, investigate or prosecute criminal offences, subject to appropriate safeguards for the rights and freedoms of third parties, unless those systems are available for the public to report a criminal offence’. Although the developed chatbot implementation, which directly interacts with natural persons is designed in an obvious way and informs the user of their interaction with an AI system, the framework still falls under the systems authorized by law to not comply with this statement, as the framework is solely developed for law enforcement use and used to investigate and detected criminal offences.
- *Article 50(2)* of the proposed European Commission Artificial Intelligence Regulation establishes that ‘providers of AI systems, including general-purpose AI systems, generating synthetic audio, image, video or text content, shall ensure that the outputs of the AI system are marked in a machine-readable format and detectable as artificially generated or manipulated. Providers shall ensure their technical solutions are effective, interoperable, robust, and reliable as far as this is technically feasible, considering the specificities and limitations of various types of content, the costs of implementation and the generally acknowledged state of the art, as may be reflected in relevant technical standards’. This is relevant in our case, as text content is produced by the language models. However, like with the previous paragraph, even though our system complies with the legal obligations, it is technically not required as the paragraph states that ‘this obligation shall not apply to the extent the AI systems perform an assistive function for standard editing or do not substantially alter the input data provided by the deployer or the semantics thereof, or where authorized by law to detect, prevent, investigate or prosecute criminal offences’.

Continuing with the legal requirements, the other regulatory document to consider, as previously mentioned, are the General Data Protection Regulations (GDPR). The GDPR regulates how personal data should be compiled, processed, and stored to ensure the protection of privacy and the fundamental rights of people regarding their data. Due to the use of location data, which is considered private, two main articles are to be considered to ensure compliance, which include Article 5, which gathers the principles relating to personal data processing (all considered in the framework by design), and Article 6, which gathers the principles of lawful personal data processing (all considered in the framework by design).

Finally, regarding the ethical principles to consider by the system, the main regulations are in the form of guidelines and requirements extracted from a set of deliverables produced by the High-Level Expert Group on Artificial Intelligence (AI HLEG) appointed by the European Commission. These guidelines ensure the ethical development and deployment of trustworthy Artificial Intelligence. Although all the ethical principles have been considered from design to ensure compliance, some stood out as especially relevant in this work's AI system. These relevant principles, and their compliance measures, can be defined as follows:

- *The principle of respect for human autonomy*: The tool does not trick, manipulate, or guide the user toward a certain unjustified conclusion, as it always maintains and ensures human autonomy and liberty. Additionally, the tool is implemented as an aid to existing processes, and is not aimed at substituting the human in decision making.
- *The principle of prevention of harm*: The tool does not harm nor endanger the integrity and dignity of both neither the user nor the suspect, as it is not implemented with malicious intent.
- *The principle of fairness*: The tool avoids discriminatory and unfair bias through the implementation of systems which do not rely on sensitive data. The human user, additionally, is not limited or tricked by the framework onto making certain decisions.
- *The principle of explicability*: All the processes carried out by the framework, decisions made, and data sources used can be traced back to their origins for transparency and explicability. Furthermore, every one of the technical components that make up the framework have been thoroughly documented and explained in a coherent and transparent manner.

A.4 CONCLUSIONS

The developed framework, and all the processing modules that make it up, brings forward a series of opportunities in various contexts ranging from economic, to social, to even environmental. However, it also assumes a series of risks to be considered. In the case of this Master's thesis, the impact of the project must be analyzed considering the context of implementation, the nature of the data used, and the nature of the systems developed. Considering all of them, it has been determined that the most relevant and main impact of this project is the legal and ethical concerns that arise due to the processing of personal data with automated AI systems in a security-based system.

An in-depth analysis has been carried out regarding the legal and ethical principles the framework must comply with for its development under European Union law, and it has been concluded that the framework complies with all the needed requirements, and could be implemented as tool in a real-life scenario.

APPENDIX B: FINANCIAL BUDGET

The budget for this Project can be summarized as follows:

Table 19. Project's financial budget

PERSONNEL COSTS (direct costs)

Hours	Hourly cost	Total
360	21 €	7.560 €

MATERIAL RESOURCES COSTS (direct costs, DC)

	Purchase price	Months of use	Depreciation (in years)	Total
Personal computer, including software	1.500,00 €	6	5	150,00 €
Server with GPU framework development	5.760,86 €	5	5	480,07 €
Server for routing motor (OSRM)	500,00 €	6	5	50,00 €

TOTAL COSTS OF MATERIAL RESOURCES	680,07 €
--	-----------------

GENERAL COSTS (indirect costs, IC)	15%	on DC	1.236,01 €
INDUSTRIAL BENEFIT	6%	on DC+IC	568,56 €

CONSUMABLES

Office supplies	30,00 €
-----------------	----------------

SUBTOTAL BUDGET	10.074,64 €
------------------------	--------------------

VAT	21%	2.115,67 €
------------	-----	-------------------

TOTAL BUDGET	12.190,31 €
---------------------	--------------------