



Hellenic Open University
School of Science and Technology
Master in Information Systems (PLS)

Postgraduate Dissertation

**Design and Development of a Standalone Offline AI System for the
Analysis and Visualization of Intelligence Reports**

George Sevastelis

Supervisor: Efthimios Tambouris

Patras, Greece, September 2026

Theses / Dissertations remain the intellectual property of students (“authors/creators”), but in the context of open access policy they grant to the HOU a non-exclusive license to use the right of reproduction, customisation, public lending, presentation to an audience and digital dissemination thereof internationally, in electronic form and by any means for teaching and research purposes, for no fee and throughout the duration of intellectual property rights. Free access to the full text for studying and reading does not in any way mean that the author/creator shall allocate his/her intellectual property rights, nor shall he/she allow the reproduction, republication, copy, storage, sale, commercial use, transmission, distribution, publication, execution, downloading, uploading, translating, modifying in any way, of any part or summary of the dissertation, without the explicit prior written consent of the author/creator. Creators retain all their moral and property rights.



Postgraduate Dissertation

**Design and Development of a Standalone Offline AI System for the
Analysis and Visualization of Intelligence Reports**

George Sevastelis

Supervising Committee

Supervisor:

Efthimios Tambouris

Professor, University of Macedonia

Co-Supervisors:

Vassilios S. Verykios

Professor, Hellenic Open University

Vasileios Zafeiropoulos

Hellenic Open University

Patras, Greece, September 2026

Acknowledgments

The author wishes to thank the supervising professor, Efthimios Tambouris, for his guidance and detailed feedback throughout the preparation of this dissertation, and the members of the examination committee, Vassilios S. Verykios and Vasileios Zafeiropoulos, for the time they devoted to its assessment.

Statement on the use of generative AI tools

During the development of the prototype described in Chapter 4 the author used generative AI coding assistants — Claude (Anthropic), ChatGPT (OpenAI) and GitHub Copilot — as programming aids. The extent of that use is recorded here so that the reader can judge the author’s own contribution to the software.

The source code of the Kalhas prototype was written by the author with the assistance of those tools. The author specified the architecture, the module boundaries, the models, the database schema and the intended behaviour of every component reported in Section 4.4; the assistants were used to draft code against those specifications, to propose alternatives, to explain the behaviour of the libraries involved and to help locate faults. Every fragment they produced was read, run, revised where necessary and integrated by the author, who is responsible for the result. The evaluation harness reproduced in Appendix D (`evaluate.py`) is the one component whose code was generated in its entirety by a single assistant, Claude. The author specified what the script had to do — import the production classes without modification, ingest the declared corpus, execute the declared query set, compute the stated measures and record the execution environment — and then reviewed the code, executed it on the target host and interpreted its output, but did not write its code by hand. That output (`results.json`) is reported in Chapter 5 as emitted, including one measure that the script computes incorrectly and which is therefore not reported (Section 5.3.2).

This statement concerns software development only. The prototype makes no use of these assistants or of any other external service at run time, and no document of the validation corpus, no query and no result was processed by them.

Abstract

Defence and security intelligence organisations receive large volumes of unstructured textual reports on a daily basis. In the organisational setting that motivates this dissertation, the annual inflow exceeds twenty thousand pages of bulletins and briefs covering political, military, social and technological topics. These documents are currently archived chronologically, without conceptual indexing, semantic search or automated analysis. The consequences are delayed retrieval, an inability to detect trends across documents, heavy dependence on manual reading by analysts, and an elevated risk of omission under information overload.

The aim of this dissertation is to design a standalone, offline information system that applies open-source natural language processing to the analysis and visualisation of such reports, without transmitting any content to external or cloud-based services. The work is delimited to the design and to a Minimum Viable Product prototype; it does not cover production deployment, classified data or organisational change management.

The methodology follows a structured, sequential system-development process. It comprises a focused literature review of the relevant research areas, elicitation and specification of functional and non-functional requirements, architectural design of a layered system, implementation of a Minimum Viable Product, and evaluation of that prototype against a quality model together with a set of task-level performance measures. The design integrates named-entity recognition, relation and event extraction, automatic summarisation, natural-language search and event-timeline visualisation over heterogeneous document formats.

The contribution is twofold: a reference architecture for offline, AI-assisted processing of high-volume reports under strict data-sovereignty constraints, and an evaluated prototype that establishes the technical feasibility of one pillar of that architecture. The prototype demonstrates that retrieval-augmented generation over a corpus of long documents, with locally hosted models, no outbound connectivity and complete retrieval provenance, is achievable on commodity hardware without a GPU. On a 283-page validation corpus, measured inside its deployed containers without a GPU, the prototype ingested every document without error at 0.85 seconds per page, returned the correct document at rank one for all fifteen known-item queries ($\text{Recall@5} = \text{MRR@5} = 1.000$) in a median of 114 milliseconds, and composed an answer with a local 3.8-billion-parameter model in a median of 41 seconds. It does not implement the structured extraction of relations and events, the event-timeline visualisation, or the access-control layer, and no feasibility claim is made for those. The isolation constraint proved not to be the obstacle; the engineering effort demanded by structured extraction, visualisation and security did.

Keywords

Information system design; offline natural language processing; named entity recognition; event-timeline visualisation; intelligence analysis; software quality evaluation.

Design and Development of a Standalone Offline AI System for the Analysis and Visualization of Intelligence Reports

Γεώργιος Σεβαστέλης

Περίληψη

Οι υπηρεσίες πληροφοριών του χώρου της άμυνας και της ασφάλειας λαμβάνουν καθημερινά μεγάλο όγκο αδόμητων κειμενικών αναφορών. Στο οργανωσιακό πλαίσιο που αποτελεί την αφορμή της παρούσας εργασίας, ο ετήσιος όγκος υπερβαίνει τις είκοσι χιλιάδες σελίδες δελτίων και ενημερωτικών σημειωμάτων, τα οποία καλύπτουν πολιτικά, στρατιωτικά, κοινωνικά και τεχνολογικά θέματα. Τα έγγραφα αυτά αρχειοθετούνται σήμερα αποκλειστικά με χρονολογική σειρά, χωρίς εννοιολογική επισήμανση, χωρίς σημασιολογική αναζήτηση και χωρίς αυτοματοποιημένη ανάλυση. Οι συνέπειες είναι η καθυστερημένη ανάκτηση της πληροφορίας, η αδυναμία εντοπισμού τάσεων μεταξύ εγγράφων, η έντονη εξάρτηση από τη χειρωνακτική ανάγνωση εκ μέρους των αναλυτών και ο αυξημένος κίνδυνος παραλείψεων υπό συνθήκες υπερφόρτωσης πληροφορίας.

Βασικός στόχος της εργασίας είναι η σχεδίαση ενός αυτόνομου, εκτός δικτύου πληροφοριακού συστήματος, το οποίο αξιοποιεί τεχνολογίες επεξεργασίας φυσικής γλώσσας ανοικτού κώδικα για την ανάλυση και οπτικοποίηση των εν λόγω αναφορών, χωρίς μεταφορά περιεχομένου σε εξωτερικές υπηρεσίες ή υπηρεσίες νέφους. Το αντικείμενο της εργασίας περιορίζεται στη σχεδίαση και στην ανάπτυξη ενός πρωτοτύπου ελάχιστου βιώσιμου προϊόντος και δεν περιλαμβάνει παραγωγική εγκατάσταση, διαβαθμισμένα δεδομένα ή διαχείριση οργανωσιακής αλλαγής.

Η μεθοδολογία ακολουθεί δομημένη, διαδοχική διαδικασία ανάπτυξης συστήματος. Περιλαμβάνει στοχευμένη βιβλιογραφική επισκόπηση των σχετικών ερευνητικών περιοχών, άντληση και προδιαγραφή λειτουργικών και μη λειτουργικών απαιτήσεων, αρχιτεκτονική σχεδίαση στρωματοποιημένου συστήματος, υλοποίηση ελάχιστου βιώσιμου προϊόντος και αξιολόγηση του πρωτοτύπου έναντι μοντέλου ποιότητας λογισμικού και μετρικών επίδοσης ανά εργασία. Η σχεδίαση ενοποιεί αναγνώριση οντοτήτων, εξαγωγή σχέσεων και γεγονότων, αυτόματη περίληψη, αναζήτηση με φυσική γλώσσα και οπτικοποίηση χρονογραμμών γεγονότων σε ετερογενείς μορφές εγγράφων.

Η συνεισφορά είναι διττή: μια αρχιτεκτονική αναφοράς για εκτός δικτύου επεξεργασία αναφορών υψηλού όγκου υπό αυστηρούς περιορισμούς κυριαρχίας δεδομένων, και ένα αξιολογημένο πρωτότυπο που τεκμηριώνει την τεχνική εφικτότητα ενός εκ των πυλώνων της. Το πρωτότυπο δείχνει ότι η ανάκτηση με επαυξημένη παραγωγή (retrieval-augmented generation) πάνω σε σώμα μακρών εγγράφων, με τοπικά μοντέλα, χωρίς καμία εξερχόμενη σύνδεση και με πλήρη ιχνηλασιμότητα ανάκτησης, είναι εφικτή σε συμβατικό υλικό χωρίς GPU. Σε σώμα επικύρωσης 283 σελίδων, χωρίς GPU, το πρωτότυπο επεξεργάστηκε κάθε έγγραφο χωρίς σφάλμα (0,85 s ανά σελίδα), επέστρεψε το σωστό έγγραφο στην πρώτη θέση και για τα δεκαπέντε ερωτήματα ($\text{Recall}@5 = \text{MRR}@5 = 1,000$) σε 114 ms και συνέθεσε απάντηση με τοπικό μοντέλο 3,8 δισ. παραμέτρων σε 41 s. Δεν υλοποιεί τη δομημένη εξαγωγή σχέσεων και γεγονότων, την οπτικοποίηση χρονογραμμής, ούτε το επίπεδο ελέγχου πρόσβασης, και δεν διατυπώνεται ισχυρισμός εφικτότητας γι' αυτά. Ο περιορισμός της απομόνωσης δεν αποδείχθηκε το εμπόδιο· η μηχανική προσπάθεια που απαιτούν η δομημένη εξαγωγή, η οπτικοποίηση και η ασφάλεια, ναι.

Λέξεις – Κλειδιά

Σχεδίαση πληροφοριακού συστήματος· επεξεργασία φυσικής γλώσσας εκτός δικτύου· αναγνώριση ονομαστικών οντοτήτων· οπτικοποίηση χρονογραμμής γεγονότων· ανάλυση πληροφοριών· αξιολόγηση ποιότητας λογισμικού.

Table of Contents

Acknowledgments.....	i
Statement on the use of generative AI tools.....	ii
Abstract.....	iii
Περίληψη.....	iv
Table of Contents.....	vi
List of Figures.....	viii
List of Tables.....	x
Abbreviations and Acronyms.....	xi
1. Introduction.....	1
1.1. Problem description.....	1
1.2. Research objectives and scope.....	2
1.3. Structure of the thesis.....	2
2. Literature Review.....	3
2.1. Introduction.....	3
2.2. Current state of practice.....	3
2.3. Foundations of natural language processing for unstructured text.....	4
2.4. Named entity recognition.....	6
2.5. Relation and event extraction.....	7
2.6. Text summarisation.....	7
2.7. Visualisation and natural-language interfaces.....	8
2.8. Offline deployment, security and software quality.....	8
2.9. Applications in public security, defence and intelligence.....	10
2.10. Research gaps.....	11
2.11. Summary.....	11
3. Methodology.....	13
4. System Development.....	16
4.1. Introduction.....	16
4.2. Analysis.....	16
4.3. Design.....	19
4.4. Implementation.....	26
4.4.1. Environment.....	26
4.4.2. Document ingestion and text extraction.....	28
4.4.3. Chunking and embedding.....	28
4.4.4. Entity extraction.....	29
4.4.5. Retrieval and the vector index.....	29
4.4.6. Answer generation.....	30
4.4.7. The analyst interface.....	31
4.4.8. Security controls.....	33
4.4.9. What was built and what was not.....	33
4.5. Summary.....	34
5. Evaluation and Results.....	36
5.1. Introduction.....	36
5.2. Evaluation framework.....	36
5.2.1. The evaluation corpus.....	37
5.3. Results.....	37
5.3.1. Functional validation of the ingestion path.....	37
5.3.2. Measured performance on the target host.....	38

5.4. Quality evaluation against ISO/IEC 25010.....	42
5.5. Usability assessment.....	43
5.6. Summary.....	44
6. Discussion.....	45
6.1. Introduction.....	45
6.2. Key findings.....	45
6.3. Limitations.....	46
6.4. Implications.....	48
6.5. Summary.....	48
7. Conclusions and Future Work.....	49
References.....	50
Appendix A: Requirements elicitation record.....	56
Appendix B: Minimum Viable Product scope definition.....	61
Appendix C: Implementation plan.....	65
Appendix D: Evaluation harness.....	67

List of Figures

Figure 1 — Retrieval-augmented generation. Diagram redrawn by the author from the architecture described by Lewis et al. (2020); the annotations stating the consequences for the present design are the author's own.....	6
Figure 2 — The seven-stage reference model for visualisation-oriented natural-language interfaces. Diagram redrawn by the author from the seven stages identified by Shen et al. (2023); the note on how the model is used in this dissertation is the author's own.....	8
Figure 3 — Threat model: covert channels for exfiltration from an air-gapped host. Diagram constructed by the author from the six categories of covert channel surveyed by Hussain et al. (2025) and the taxonomy of Carrara and Adams (2016); the RAMBO instance is taken from Guri (2024) and the term bridgeware from Guri and Elovici (2018).	9
Figure 4 — Zero-trust core components applied to the security layer. Diagram redrawn by the author from the core components defined by Rose et al. (2020) in NIST Special Publication 800-207; the mapping of the resource onto the report corpus is the author's own.....	10
Figure 5 — Use case diagram of the proposed system. Author's own diagram, derived from the use cases of Table 3.....	17
Figure 6 — Layered architecture of the proposed standalone system. Author's own diagram, derived from the layer responsibilities of Table 5.....	20
Figure 7 — Component view of the proposed system. Author's own diagram, derived from the layer responsibilities of Table 5.....	21
Figure 8 — Package structure and permitted dependencies. Author's own diagram.....	22
Figure 9 — Domain model of the data layer, as a UML class diagram. Author's own diagram.....	23
Figure 10 — Activity: end-to-end processing of an uploaded report. Author's own diagram.....	23
Figure 11 — State machine: lifecycle of an ingested report. Author's own diagram.....	24
Figure 12 — Sequence: ingestion and extraction of a report. Author's own diagram.....	24
Figure 13 — Sequence: natural-language query grounded in the report corpus. Author's own diagram; the retrieve-then-generate pattern follows Lewis et al. (2020) and the stage sequence follows Shen et al. (2023).....	25
Figure 14 — Deployment: the standalone, air-gapped installation. Author's own diagram; the countermeasure position follows Saeed et al. (2025).....	26
Figure 15 — Deployment as built: the Kalhas Docker Compose installation. Author's own diagram, drawn from docker-compose.yml, the two Dockerfiles and the setup scripts.....	27
Figure 16 — Implemented module structure of the backend. Author's own diagram; every box corresponds to a file in the repository and the names are those of the actual classes and methods.....	28
Figure 17 — Implemented database schema. Author's own diagram, drawn from backend/db/models.py.....	30
Figure 18 — Sequence as built: natural-language query in the Kalhas MVP. Author's own diagram, traced from api/routes/query.py, retrieval/retriever.py and nlp/generator.py.....	31

Figure 19 — Upload and Process view of the Kalhas front end, captured while guri2024_rambo.pdf is being processed. Screenshot of the deployed Streamlit application.	32
Figure 20 — Chat view after a grounded answer. The generated text is followed by the three source attributions with their similarity scores. Screenshot of the deployed Streamlit application.....	32
Figure 21 — Chat view with the source panel expanded: the retrieved passages are shown verbatim beneath their similarity scores. Screenshot of the deployed Streamlit application.	33
Figure 22 — Use-case coverage: the designed scope of Table 3 set against the implemented artefact. Author's own analysis of the repository.....	34
Figure 23 — Wall-clock time of the ingestion stages for each document of the validation corpus, second run of 19 September 2026 (total 241.7 s). Author’s own chart from results.json; text extraction and chunking are below the resolution of the bar for all but the longest document.....	39

List of Tables

Table 1 — Abbreviations and acronyms.....	xi
Table 2 — Categories of existing AI systems for report analysis and their deployment models.....	4
Table 3 — Functional requirements expressed as use cases.....	17
Table 4 — Non-functional requirements mapped to ISO/IEC 25010.....	18
Table 5 — Layered architecture of the proposed system.....	20
Table 6 — Functional validation of the ingestion path on the target host: text extracted, chunks produced, entities recognised and wall-clock time per stage for each document of the technical validation corpus, using the production classes at the configured window of 512 characters with 50 characters of overlap (second run of 19 September 2026, results.json).....	38
Table 7 — Latency by stage on the target host, second run of 19 September 2026. Ingestion stages are per document (n = 12); the query stages are per query over 75 executions (15 queries × 5 repetitions) against the 2 147-vector index; generation is per answer over the first five queries (phi3:mini, temperature 0.3, 200-token ceiling).....	38
Table 8 — Known-item retrieval on the declared query set of Table D.1, top-k = 5. A retrieved chunk counts as relevant if its source document is the document declared for the query. Recall@5 = 1.000, MRR@5 = 1.000; identical in both runs.....	40
Table 9 — Descriptive profile of the entities recognised by en_core_web_sm over the validation corpus, by spaCy label (9 052 entities, 18 labels, 32.0 per page). Counts only; no accuracy is implied.....	41
Table 10 — Status of each planned measure after the execution of the harness of Appendix D on the target host.....	42

Abbreviations and Acronyms

Table 1 — Abbreviations and acronyms

Term	Full name
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
DLP	Data Leakage Prevention
DocRE	Document-level Relation Extraction
EE	Event Extraction
FAISS	Facebook AI Similarity Search (vector index library)
GPU	Graphics Processing Unit
HOU	Hellenic Open University
ICS	Industrial Control System
IE	Information Extraction
LLM	Large Language Model
LoRA	Low-Rank Adaptation
MRR	Mean Reciprocal Rank
MVP	Minimum Viable Product
nDCG	Normalised Discounted Cumulative Gain
NER	Named Entity Recognition
NLP	Natural Language Processing
OSINT	Open Source Intelligence
RAG	Retrieval-Augmented Generation
RE	Relation Extraction
SLR	Systematic Literature Review
SRE	Software Requirements Engineering
V-NLI	Visualization-oriented Natural Language Interface
ZTA	Zero Trust Architecture

1. Introduction

1.1. Problem description

Organisations that produce and consume intelligence reports operate in an environment of sustained information overload. The volume of data that must be stored, transmitted and processed grows at an exponential rate, and in open-source intelligence the scale of the available sources is itself a problem for investigations — one that AI can mitigate by reducing analyst workload (Browne et al., 2024). The same pressure is documented across the wider public-security and cybersecurity domains, where systematic reviews report that the growth of unstructured data is driving a demand for automated text analytics and identify public-security agencies as major beneficiaries of it (de Carvalho & Costa, 2023; Ignaczak et al., 2021).

The concrete problem addressed by this dissertation arises in a defence intelligence directorate whose daily inflow consists of bulletins and briefs, typically longer than twenty pages each and delivered in heterogeneous formats. The cumulative annual volume exceeds twenty thousand pages and spans political, military, social and technological subject matter. This figure is drawn from the requirements-elicitation material produced by the author for the present study and reproduced in Appendix A; it characterises the specific organisational setting under study and is not offered as a general statistic. The reports are archived chronologically. There is no conceptual indexing, no semantic search, no automated extraction of entities or events, and no mechanism for detecting trends across documents.

Three consequences follow, and each is well attested in the research literature. **First**, retrieval is slow and incomplete: without entity- or event-level indexing, analysts must read linearly, and information extraction from unstructured narrative text is precisely the task that named-entity recognition, relation extraction and event extraction were developed to automate (Q. Li et al., 2024; Zhang et al., 2025). **Second**, cross-document analysis is effectively absent: the construction of event timelines from text requires temporal information extraction and reasoning, capabilities that no chronological file system provides (Leeuwenberg & Moens, 2020). **Third**, the burden of routine reading falls on human analysts, which both consumes expert capacity and introduces the risk of omission; the argument that withholding decision-support automation from military and intelligence processes is itself an ethical failure has been made explicitly in the literature on responsible AI in the military (Meerveld et al., 2023), and the same work treats the intelligence cycle as a sub-process of the military decision-making process in which every step lends itself to AI-based automation (Meerveld et al., 2023); the conditions under which that support remains defensible are analysed separately (Probasco et al., 2025).

The problem is significant because the cost of an omission in this domain is not a delayed report but a degraded decision. At the same time, the obvious remedy — sending documents to a commercial large-language-model service — is unavailable: the material is sensitive, and the privacy and leakage risks of large language models and of retrieval-augmented architectures are documented in detail (Das et al., 2025; Zeng et al., 2024). Any acceptable solution must therefore run entirely offline, which reframes the problem from one of model selection into one of system design under a hard data-sovereignty constraint.

1.2. Research objectives and scope

The main objective of this dissertation is to design a standalone, offline information system that applies open-source natural language processing to the analysis and visualisation of high-volume intelligence reports, and to validate that design through the implementation and evaluation of a Minimum Viable Product prototype.

The sub-objectives of the dissertation are the following:

- To review, critically and in a focused manner, the research areas on which such a system depends: information extraction from unstructured text, summarisation, visualisation of extracted knowledge, and offline deployment of AI under security constraints.
- To elicit and specify the functional and non-functional requirements of the system, with explicit attention to the offline constraint.
- To design a modular, layered architecture that satisfies those requirements.
- To implement a Minimum Viable Product covering the highest-priority functions.
- To evaluate the prototype against a recognised software quality model and against task-level performance measures for the extraction and retrieval components.

The scope of the study is delimited as follows. The dissertation covers the **analysis, design, implementation** and **evaluation** of a **prototype** operating on unclassified, representative documents. It does not cover deployment in a production or accredited environment, does not process classified material, does not address the organisational and procedural change required to adopt such a system, and does not attempt to advance the state of the art in any individual NLP model. The contribution lies in the integration and in the design of the system under the offline constraint, not in model novelty. Predictive analytics and automated alerting are explicitly excluded from the Minimum Viable Product and are treated as future work.

1.3. Structure of the thesis

The remainder of this dissertation is organised as follows. **Chapter 2** presents the literature review. It establishes the research areas on which the system depends, surveys the current state of practice in comparable commercial and open-source systems, and **identifies the gaps that motivate the proposed design**. **Chapter 3** sets out the methodology: it justifies the chosen system-development approach against the alternatives and specifies, as a sequence of steps and sub-steps, the work that will be carried out. **Chapter 4** reports the system development itself, structured as analysis, design and implementation. **Chapter 5** presents the evaluation framework and the results obtained from applying it to the prototype. **Chapter 6** discusses those results, states the limitations of the study, and considers the implications for practice. **Chapter 7** concludes and identifies directions for future work. The requirements-elicitation material, the Minimum Viable Product scope definition and the implementation plan produced during the study are reproduced in Appendices A, B and C respectively. The evaluation harness, its query set and the record of its execution are reproduced in Appendix D.

2. Literature Review

2.1. Introduction

This chapter reviews the research literature that underpins the design of an offline information system for the analysis and visualisation of intelligence reports. Following the guidance that a literature review should first establish whether the ground has already been covered, the review begins from existing systematic reviews in each area and uses them as entry points into the primary literature.

The scope of the review is determined by the functional decomposition of the problem set out in Chapter 1. Processing a report end-to-end requires, in order: converting unstructured text into structured representations; identifying the entities named in the text; identifying the relations and events that connect them, including their temporal ordering; condensing long documents into readable summaries; presenting the extracted knowledge in a form an analyst can interrogate; and doing all of the above on hardware that is physically isolated from any network. Each of these steps corresponds to an established research area, and the review is organised accordingly:

- Section 2.2 surveys the current state of practice, that is, the commercial and open-source systems that already address parts of this problem.
- Section 2.3 covers the foundations of **natural language processing for unstructured text**, and establishes why **transformer-based models** rather than earlier approaches form the basis of the design.
- Section 2.4 covers named entity recognition, the component that produces the primary structured output.
- Section 2.5 covers relation extraction and event extraction, including temporal information extraction, which together supply the material for event timelines.
- Section 2.6 covers text summarisation, which addresses the length of individual reports.
- Section 2.7 covers visualisation and natural-language interfaces, which determine how extracted knowledge reaches the analyst.
- Section 2.8 covers offline and air-gapped deployment, the security threats specific to it, and the software quality model used to evaluate the resulting system.
- Section 2.9 covers applications of these techniques in public security, defence and intelligence, which is the application domain of this dissertation.
- Section 2.10 states the research gaps that follow from the review, and Section 2.11 summarises the chapter.

This ordering is not arbitrary: it follows the data flow through the proposed system, so that each section reviews the literature for one stage of the processing pipeline that Chapter 4 will implement.

2.2. Current state of practice

A number of commercial platforms already apply artificial intelligence to the analysis of security-relevant reports. The systematic review of AI for open-source intelligence by Browne et al. (2024) identifies one hundred and sixty-three research articles in this space and characterises the state of the field: research is concentrated on classification models serving

the collection, processing and analysis phases of the intelligence cycle, but the review explicitly identifies the incorporation of pre-existing OSINT tools with AI as an open research gap, and names multi-lingual support and integration with live applications among its directions for future research. Ignaczak et al. (2021), whose systematic protocol identified two thousand one hundred and ninety-six studies on text mining in cybersecurity, of which eighty-three were reviewed in depth, likewise identify the analysis of non-English content as an open gap. ***These two gaps — multilingual support and integration — are directly relevant to the present work.***

Table 2 summarises the categories of system that occupy this space. The entries are characterised by function rather than by vendor claim, because vendor documentation does not constitute an academic source and no such source was available for the individual products at the time of writing. The essential and consistent property of this class of system, for the purposes of this dissertation, is that it is predominantly cloud-based or hybrid, which is the property that makes it unusable in the setting described in Chapter 1.

Table 2 — Categories of existing AI systems for report analysis and their deployment models

System category	Typical capability	Deployment model
Cyber threat intelligence platforms	Behavioural anomaly detection over network and endpoint telemetry; automated incident report generation and visualisation.	Predominantly cloud or hybrid
OSINT link-analysis platforms	Entity extraction from open sources, link analysis, graph-based visualisation of relationships.	Cloud, hybrid, or on-premise with external enrichment
Competitive and market intelligence platforms	AI-assisted search and summarisation over document corpora; trend visualisation.	Cloud
Security operations automation platforms	Automated triage and report analysis; agentic workflows over security data.	Cloud

The literature that motivates the offline requirement is unambiguous. Das et al. (2025), surveying the security and privacy challenges of large language models, catalogue jailbreaking, data-poisoning and personally-identifiable-information leakage attacks against deployed models. Zeng et al. (2024) demonstrate empirically that retrieval-augmented generation systems can be induced to leak the contents of their private retrieval database — the precise architecture that a naive implementation of the system proposed here would adopt. Taken together with the cloud-dependence of the existing platforms, this establishes the gap that the present design addresses: ***none of the surveyed classes of system is designed to operate with no external connectivity at all.***

2.3. Foundations of natural language processing for unstructured text

The processing of unstructured text has passed through three broadly successive paradigms, and the design decisions taken in Chapter 4 depend on which of them is adopted. **Symbolic and rule-based methods encode linguistic knowledge explicitly.** They are properly described as a knowledge-engineering approach to language processing rather than as a machine-learning technique, and they remain in use where precision and auditability outweigh coverage. **Statistical and classical machine-learning methods learn from annotated corpora but rely on manually designed, task-specific features (Q. Li et al., 2024), and the resulting models are specific to the task for which they were trained.**

The third paradigm, and the one adopted here, is the transformer. Vaswani et al. (2017) introduced an architecture based entirely on *attention mechanisms*, dispensing with recurrence and convolution, and demonstrated both superior quality and substantially greater parallelisability. Devlin et al. (2019) built on this with BERT, which pre-trains deep bidirectional representations on unlabelled text and can then be fine-tuned with a single additional output layer to reach state-of-the-art performance across eleven natural language processing tasks. The significance of this for the present work is architectural rather than merely empirical: a single pre-trained encoder can be adapted to entity recognition, relation extraction and classification alike, which means that one model family can serve several of the required subsystems.

Three further results bear directly on the offline constraint. Sanh et al. (2019) show, with DistilBERT, that knowledge distillation during pre-training can reduce model size by forty per cent while retaining ninety-seven per cent of language-understanding capability and running sixty per cent faster, and they demonstrate this explicitly for on-device computation. Dettmers et al. (2023) show, with QLoRA, that back-propagating gradients through a frozen, four-bit quantised model into low-rank adapters permits the fine-tuning of a sixty-five-billion-parameter model on a single forty-eight-gigabyte GPU while preserving full sixteen-bit fine-tuning performance. **Together these results establish that capable models can be run and adapted on a single self-contained machine, which is the technical precondition for everything that follows.** Lewis et al. (2020) introduce retrieval-augmented generation, combining a parametric seq2seq model with a non-parametric dense vector index; this is the mechanism by which the proposed system will ground natural-language queries in the report corpus, subject to the privacy caveat established in Section 2.2.

Figure 1 redraws that architecture in the form in which it is adopted here. The upper part of the figure is the architecture described by Lewis et al. (2020); the consequences stated beneath it are the present author's, and they follow from the offline constraint of Section 1.1 together with the privacy result of Zeng et al. (2024) discussed in Section 2.2.

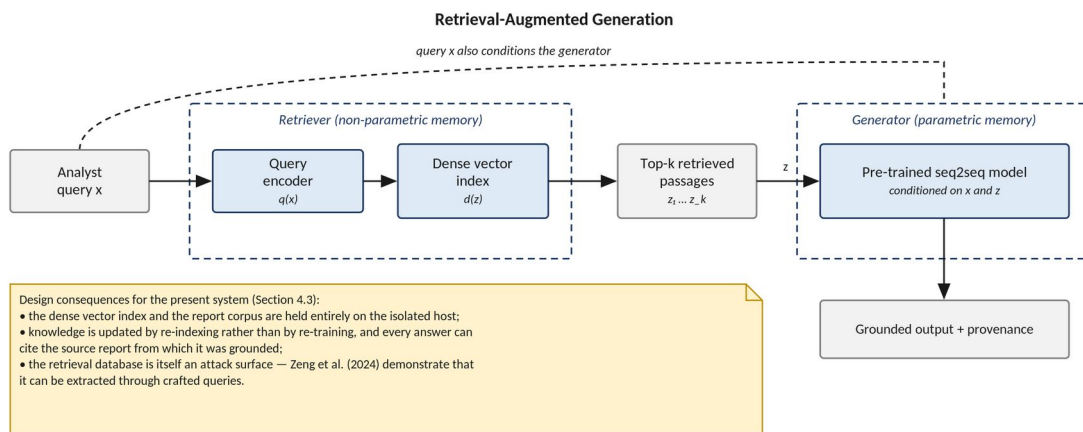


Figure 1 — Retrieval-augmented generation. Diagram redrawn by the author from the architecture described by Lewis et al. (2020); the annotations stating the consequences for the present design are the author's own.

The language of the corpus is a first-order design constraint rather than a detail. Koutsikakis et al. (2020) present GREEK-BERT, a monolingual model for modern Greek, and show that it achieves state-of-the-art results on part-of-speech tagging, named entity recognition and natural language inference; on the two more demanding of these benchmarks, named entity recognition and natural language inference, it outperforms the multilingual models M-BERT by roughly four to five points and XLM-R by roughly one point, and shallower word-embedding baselines by five to ten points. Voukoutis et al. (2024) present Meltemi, a seven-billion-parameter Greek large language model obtained by continued pre-training of Mistral on a forty-billion-token Greek corpus, released under a permissive licence. Both are relevant because both can be hosted locally, and because the reports in the target corpus are not in English.

2.4. Named entity recognition

Named entity recognition supplies the primary structured output of the proposed system: the names, locations, dates and organisations around which an intelligence report is organised. **The task admits three variants** — flat, nested and discontinuous entity recognition — and the principal weakness of the sequence-labelling models that dominated the field is that they are task-specific and cannot address all three concurrently (Mo et al., 2023).

Generative, sequence-to-sequence formulations unify the three variants in a single framework. Mo et al. (2023) observe that such models are nonetheless ineffective at two things: detecting entity boundaries, because token-relations are not captured by auto-regressive decoding, and estimating entity types. They propose a multi-task transformer that adds an entity-boundary-detection task, realised by classifying the relations between tokens within a sentence, and that uses an external knowledge base to compute prior entity-type distributions. Mo and Li (2024) develop a related multiple-attention framework that injects entity-type information into the encoder self-attention and the decoder self- and cross-attention, and word–word relation representations into the decoder cross-attention, and report that the method significantly improves on, or matches, the best generative NER models across four flat and two long-entity benchmarks.

The domain-adaptation question is the one that matters most here, and it has been answered in a closely analogous setting. Xing and Chen (2024) apply lightweight open-source large language models of six to seven billion parameters to the extraction of key elements from Chinese police reports. Prompt engineering alone improved the F1 score by approximately three per cent; LoRA fine-tuning improved it by ten to fifty per cent; a zero-shot data-augmentation method yielded a further four per cent. Against a fine-tuned universal information extraction model, their domain-specific model improved per-element F1 by approximately five per cent, and by forty-two per cent for the organisation element. This is the closest published analogue to the present problem — short, formulaic, high-volume, non-English operational reports — and it establishes that the approach is viable with models small enough to run on a single machine. Adhikary et al. (2024) reach a compatible conclusion in the legal domain, showing that a few-shot learning approach based on large language models can extract structured attributes from criminal case proceedings within a sequence-labelling framework, and that the extracted attributes are useful in downstream tasks.

2.5. Relation and event extraction

Entities alone do not answer analytical questions. The relations between entities, and the events in which they participate, are what an analyst actually needs, and they are what makes a timeline possible.

Document-level relation extraction is the relevant formulation, because in a multi-page report the entities involved in a relation rarely occur in the same sentence. Zhang et al. (2025) identify two obstacles in existing large-language-model approaches: candidate-relation filtering is performed at document level, which yields insufficient correlation between candidate relations and specific entity pairs, and the large volume of no-relation data creates a data-imbalance problem. Their entity-pair-guided relation summarisation and retrieval model shifts filtering to the entity-pair level and reports state-of-the-art results among large-language-model approaches. Dagdelen et al. (2024) take a different and, for an engineering project, highly practical route: they fine-tune GPT-3 and Llama-2 for joint named-entity recognition and relation extraction, and have the model return records directly as structured JSON objects suitable for insertion into a database. They describe the approach as simple, accessible and highly flexible, which is exactly the property required by a prototype built by a single developer.

Event extraction is surveyed by Q. Li et al. (2024), who classify the deep-learning literature by task definition and summarise the available benchmarks and evaluation metrics. The temporal dimension is treated separately by Leeuwenberg and Moens (2020), who survey three decades of temporal reasoning for temporal information extraction and show how symbolic temporal reasoning is integrated with machine-learning-based extraction so that a coherent global temporal view of a text can be constructed. This is the theoretical basis for the event-timeline component of the proposed system. Olsen et al. (2024) survey the datasets available for socio-political events of conflict and unrest, and identify discrepancies that are directly relevant here: political-science event databases encode aggregated, real-world events reconstructed from an entire document and from expert knowledge, whereas NLP event datasets annotate specific and more atomic text spans — what Olsen et al. call the abstraction gap. The proposed system needs the former but can only be built from models trained on the latter, and this gap is a genuine constraint on what it can achieve.

2.6. Text summarisation

Individual reports in the target corpus exceed twenty pages, which places them squarely in the long-document regime. Sharma et al. (2024) address exactly this case, proposing an unsupervised, extractive, hierarchical graph-based ranking algorithm that exploits the discourse structure of long documents, and demonstrating on the ScisummNet dataset that leveraging structural information outperforms comparable unsupervised algorithms. Zhao et al. (2024) propose HAEsum, which models a document both locally, through a heterogeneous graph over intra-sentence relations, and globally, through a hypergraph self-attention layer over high-order inter-sentence relations, and argue that existing methods err by modelling only one of the two.

A third strand is directly aligned with the architecture proposed in Chapter 4. Kushk and Kochut (2021) propose ESDL, which summarises entities rather than documents, using deep learning to rank the triples that constitute an entity's description in a knowledge base. Since the system under design already extracts entities, entity-level summarisation offers a summary that is organised around the analyst's actual unit of interest rather than around the document.

2.7. Visualisation and natural-language interfaces

Extraction is worthless if its output cannot be interrogated. Two bodies of work govern this stage.

The first concerns the representation of the extracted structure. Beck et al. (2017) provide a taxonomy and survey of dynamic graph visualisation and identify the representation of time as the major distinguishing feature: dynamic graphs are shown either as animated diagrams or as static charts on a timeline. Their evaluation evidence on dynamic stability and the preservation of the viewer's mental map bears directly on the choice made for the event-timeline component. Sarikaya et al. (2019) survey the design space of dashboards and show that the term now covers a far wider range of artefacts than the classical monitoring display, extending to interactive interfaces with multiple views and purposes; their design dimensions provide the vocabulary in which the analyst-facing interface of the proposed system can be specified. The value of entity-centric visual analysis in precisely this application domain is established by the Jigsaw system, which extracts entities from document collections and presents their connections across coordinated list, graph and timeline views in support of investigative analysis (Stasko et al., 2008); the decade of deployment experience reported by its authors documents both the analytical value of the approach and the sustained engineering effort that such tools demand (Görg et al., 2014). The second concerns how the analyst poses questions. Shen et al. (2023) survey visualisation-oriented natural language interfaces and decompose them into seven stages — query interpretation, data transformation, visual mapping, view transformation, human interaction, dialogue management and presentation. This decomposition is adopted in Chapter 4 as the reference model for the natural-language query subsystem, because it separates the language problem from the visualisation problem in a way that maps cleanly onto a modular implementation.

Figure 2 sets out that decomposition. It is reproduced here because it is adopted, unchanged, as the reference model for the natural-language query subsystem designed in Section 4.3.

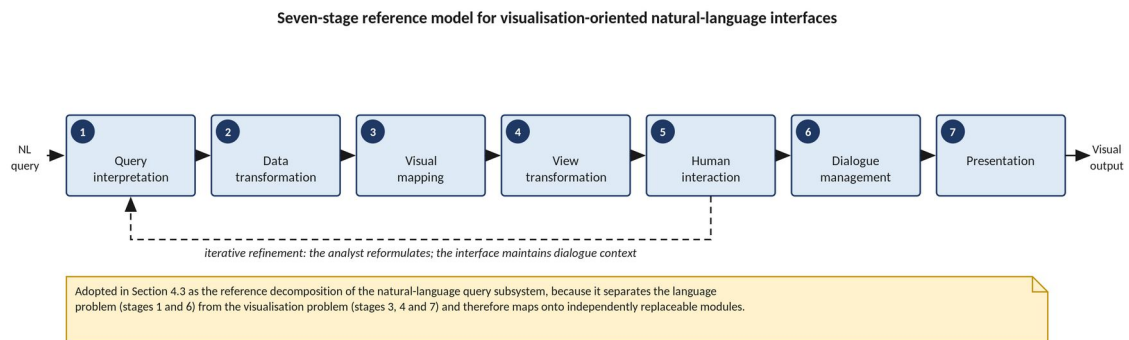


Figure 2 — The seven-stage reference model for visualisation-oriented natural-language interfaces. Diagram redrawn by the author from the seven stages identified by Shen et al. (2023); the note on how the model is used in this dissertation is the author's own.

2.8. Offline deployment, security and software quality

The offline constraint is the defining feature of the proposed system, and it must be understood as a security posture rather than merely an absence of connectivity.

Physical isolation does not confer security by itself. Guri and Elovici (2018) use the term *bridgware* for malware designed to bridge air-gap isolation in order to leak data, and note

that supply-chain attacks and infected removable media have successfully compromised air-gapped networks in the past. Guri (2024) demonstrates the RAMBO attack, in which malware generates radio signals from the memory bus and exfiltrates data at up to one thousand bits per second to a software-defined radio receiver located a few metres away. Carrara and Adams (2016) provide the general taxonomy of out-of-band covert channels, and Hussain et al. (2025) survey the acoustic, electromagnetic, optical, magnetic, power and thermal categories of attack against industrial air-gapped systems. Saeed et al. (2025) review the same threat landscape and go further, proposing a new ISO 27001 clause specifically for securing air-gapped systems. The design implication is that isolation must be accompanied by controls, not treated as a substitute for them.

Figure 3 collects these results into the threat model that the design of Section 4.3 must answer. It is not a hypothetical construction: every branch of it corresponds to a class of attack demonstrated in the cited literature.

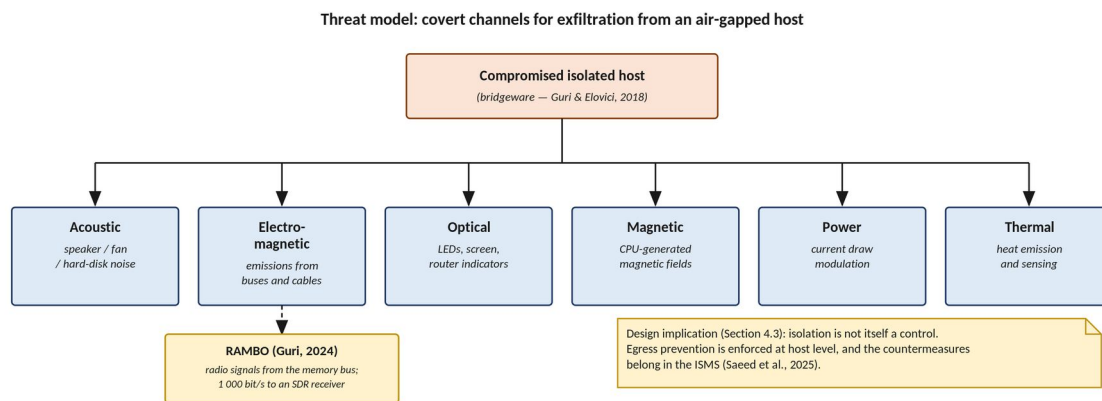


Figure 3 — Threat model: covert channels for exfiltration from an air-gapped host. Diagram constructed by the author from the six categories of covert channel surveyed by Hussain et al. (2025) and the taxonomy of Carrara and Adams (2016); the RAMBO instance is taken from Guri (2024) and the term bridgware from Guri and Elovici (2018).

Two complementary frameworks supply those controls. Rose et al. (2020), in NIST Special Publication 800-207, define zero trust architecture: no implicit trust is granted to an asset or user account on the basis of physical or network location, and authentication and authorisation are discrete functions performed before any session to a resource is established. Alneyadi et al. (2016) survey data leakage prevention systems, which detect and prevent the leakage of confidential data in use, in transit and at rest. Applied to the present system, these two together specify the security layer of the architecture presented in Section 4.3.

Figure 4 shows the zero-trust core components as they apply to the security layer of the proposed system.

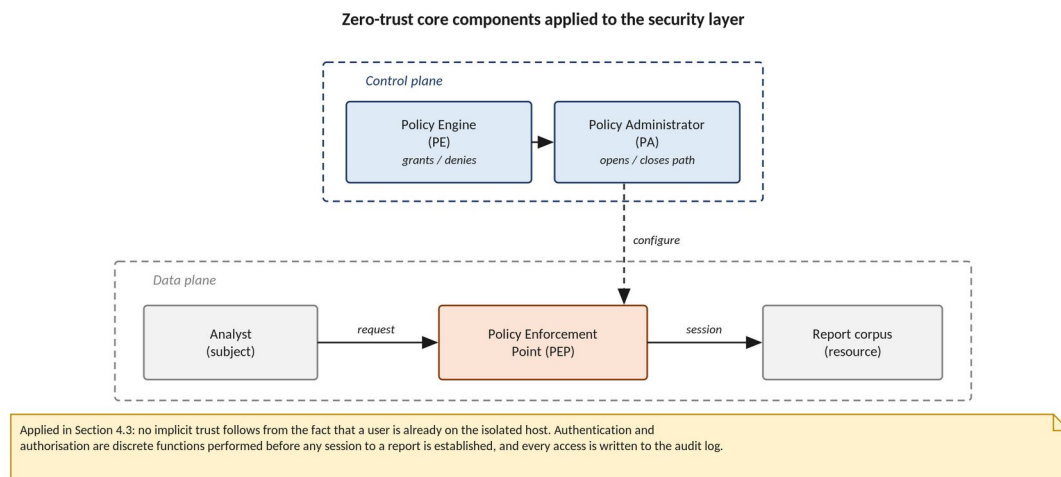


Figure 4 — Zero-trust core components applied to the security layer. Diagram redrawn by the author from the core components defined by Rose et al. (2020) in NIST Special Publication 800-207; the mapping of the resource onto the report corpus is the author's own.

Quality must be evaluated against a recognised model rather than asserted. Rojas et al. (2025) map the evolution of ISO/IEC 25010 through a bibliometric analysis of one thousand six hundred and thirteen articles and document its international significance and global adoption as a standard for software quality evaluation. Trichkova-Kashamova (2021) applies the ISO/IEC 25010 quality models specifically to the assessment of information systems, framing evaluation as an optimisation problem whose purpose is to minimise the subjectivity of the assessing expert. Barletta et al. (2023) demonstrate a concrete instantiation, proposing measures for each quality-in-use characteristic — effectiveness, efficiency, satisfaction, freedom from risk and context coverage — and combining them through the Analytic Hierarchy Process. This provides the evaluation instrument used in Chapter 5.

Two further engineering-method sources inform Chapters 3 and 4. Lenarduzzi and Taibi (2016) establish, through a systematic mapping study, the common definitions of the Minimum Viable Product, and Alonso et al. (2023) map the software-engineering practices actually used to develop MVPs across thirty-three papers, noting in particular that research on MVP technical-feasibility assessment remains limited — which is precisely the contribution a prototype of this kind can make. On the requirements side, Necula et al. (2024) systematically review the use of natural language processing in software requirements engineering, and T. Li et al. (2024) review machine learning for requirements engineering across two hundred and seven academic papers. S. Li et al. (2021) survey the quality attributes of microservices architecture and identify scalability, performance, availability, monitorability, security and testability as the six most frequently addressed attributes together with nineteen architectural tactics that address them; these inform the layered decomposition adopted in Section 4.3.

2.9. Applications in public security, defence and intelligence

The application domain has its own literature, and it is a literature of caution as much as of capability.

On the capability side, de Carvalho and Costa (2023) systematically review text mining and analytics for public security across four major bibliographic databases and identify the main application areas and the most recurrent tools. Ignaczak et al. (2021) do the same for cybersecurity, proposing a taxonomy of the activities that text mining supports. Browne et

al. (2024) do so for open-source intelligence. Deng and Ning (2021) survey societal event forecasting with deep learning, covering civil unrest and crime, and set out how event-forecasting problems are formulated as machine-learning prediction tasks — the natural extension of the present system once entity and event extraction are in place, and therefore the basis of the future-work proposals in Chapter 7.

On the caution side, Meerveld et al. (2023) argue that the ethical debate on military AI has been distorted by an excessive focus on lethal autonomous weapons, and that decision-support optimisation across the military decision-making process presents opportunities that cannot responsibly be ignored; their framing of the intelligence cycle is the one adopted in this dissertation. Probasco et al. (2025) analyse the advantages and risks of AI for military decision-making and are explicit about the failure modes. Rivera et al. (2024) provide the strongest empirical warning: in simulated wargames, all five off-the-shelf large language models studied exhibited forms of escalation and difficult-to-predict escalation patterns, tending to develop arms-race dynamics. The design consequence is stated directly in Chapter 4: the proposed system extracts, summarises and visualises, and it does not recommend, score or decide. That boundary is a deliberate design decision grounded in this literature.

2.10. Research gaps

The review supports four gaps, each traceable to a specific source.

- Deployment under isolation is under-researched. Browne et al. (2024) name integration with live applications among the future directions of AI-based OSINT research, and the security literature (Das et al., 2025; Zeng et al., 2024) documents the leakage risks that make the cloud-based integration path unacceptable in this domain. The intersection — capable NLP under complete network isolation — is not addressed.
- Non-English operational text is under-served. Ignaczak et al. (2021) name the analysis of non-English content as an open gap, and Browne et al. (2024) likewise point to multi-lingual support. Greek-language resources exist (Koutsikakis et al., 2020; Voukoutis et al., 2024) but have not been applied to this task.
- Domain adaptation for short operational reports is demonstrated only in adjacent domains. Xing and Chen (2024) show it for police reports and Adhikary et al. (2024) for legal documents; the defence intelligence report has not been treated.
- MVP technical-feasibility assessment is itself an acknowledged gap in the software-engineering literature (Alonso et al., 2023), which is what makes an evaluated prototype a contribution rather than merely an artefact.

2.11. Summary

This chapter established the research areas on which the proposed system depends and reviewed each in turn. Transformer-based models provide a single architecture that can serve entity recognition, relation extraction and summarisation alike, and distillation and quantisation make them deployable on a single isolated machine. Document-level relation extraction and temporal information extraction supply the material for event timelines. Established taxonomies exist for the visualisation of dynamic graphs, for dashboards and for natural-language interfaces to visualisation. The security literature shows that air-gapping is a necessary but not sufficient control, and ISO/IEC 25010 provides the instrument against which the resulting system can be evaluated. The domain literature supports the use of AI for

decision support in the intelligence cycle while warning explicitly against extending it to recommendation or decision. Four gaps follow from this review, and they define what the remainder of the dissertation attempts.

3. Methodology

This chapter states how the research problem will be investigated. It justifies the chosen approach against the alternatives, delimits its applicability, and then specifies the exact steps and sub-steps that will be followed. In accordance with the dissertation template, this chapter contains neither an introductory nor a concluding sub-section, and it reports no results: the outcome of executing each step below is reported in Chapters 4 and 5.

Choice of research approach. The research question of this dissertation is a design question — whether a system with a given set of properties can be built and made to work under a given constraint — rather than an explanatory or predictive one. Three approaches were considered.

The first was a purely analytical or conceptual study, producing an architecture and a requirements specification without an artefact. Its advantage is that it permits broader coverage of the design space within the available time; its decisive weakness is that it cannot answer the feasibility question, and Alonso et al. (2023) identify technical-feasibility assessment as one of the two under-researched aspects of MVP work. It was rejected on that ground.

The second was an empirical comparative study, benchmarking existing NLP models on a corpus of intelligence reports. Its advantage is methodological rigour and a clean quantitative result; its weakness is that it would evaluate models rather than a system, and the contribution claimed in Chapter 1 is a system-level one — the integration under the offline constraint. It would also require a substantial annotated corpus, which does not exist for this domain and language. It was rejected on both grounds, although its evaluation techniques are retained within the approach that was adopted.

The third approach, and the one adopted, is structured system development culminating in an evaluated Minimum Viable Product. It is capable of answering the feasibility question, it accommodates the model-level evaluation of the second approach as a sub-component, and it produces an artefact against which a recognised quality model can be applied. Its limitations must be stated plainly: a single artefact evaluated by a single developer supports claims about feasibility and about the specific quality characteristics measured, but it does not support statistical generalisation, and the absence of a comparison system means that no claim of superiority over existing tools can be made. These limitations are revisited in Chapter 6.

Nature of the literature review. Following the guidance of the dissertation template, the distinction between a simple and a systematic literature review was considered. A full systematic review following a formal protocol was judged disproportionate to the weight of the theoretical component in a design-oriented dissertation. A focused review was therefore adopted, with one methodological safeguard borrowed from systematic practice: the search began from existing systematic reviews in each area (Alonso et al., 2023; Browne et al., 2024; de Carvalho & Costa, 2023; Ignaczak et al., 2021; S. Li et al., 2021; T. Li et al., 2024; Necula et al., 2024; Rojas et al., 2025) and proceeded outwards from them, so that established ground was not re-covered.

Steps of the method. The following steps will be carried out.

Step 1: Literature review of the constituent research areas

- Sub-step 1.1: Decompose the problem into its functional stages and derive the search terms from that decomposition.
- Sub-step 1.2: Search for existing systematic literature reviews in each area, in Scopus, Web of Science, IEEE Xplore, the ACM Digital Library and Google Scholar.
- Sub-step 1.3: Search for primary studies in each area, using the reviews found in Sub-step 1.2 as entry points and applying forward and backward snowballing.
- Sub-step 1.4: Apply inclusion criteria — peer-reviewed, published from 2016 onwards unless foundational, and directly relevant to one of the identified stages — and exclusion criteria.
- Sub-step 1.5: Synthesise the retained studies thematically, by processing stage rather than by author, and record the research gaps.

Step 2: Requirements elicitation and specification

- Sub-step 2.1: Elicit the current state of practice, the document volumes and the pain points from the operational setting, and record them.
- Sub-step 2.2: Identify the stakeholder groups and their goals.
- Sub-step 2.3: Derive functional requirements and express them as use cases.
- Sub-step 2.4: Derive non-functional requirements, mapped onto the characteristics of ISO/IEC 25010.
- Sub-step 2.5: Prioritise the requirements to determine the scope of the Minimum Viable Product, following the MVP definitions of Lenarduzzi and Taibi (2016).

Step 3: Architectural design

- Sub-step 3.1: Select the architectural style and justify it against the quality attributes identified in Sub-step 2.4, drawing on the quality-attribute tactics catalogued by S. Li et al. (2021).
- Sub-step 3.2: Decompose the system into layers and specify the responsibility and the interface of each.
- Sub-step 3.3: Design the data model for documents, extracted entities, relations and events.
- Sub-step 3.4: Design the security layer, applying the zero-trust principles of Rose et al. (2020) and the data-leakage controls surveyed by Alneyadi et al. (2016), and taking account of the air-gap threat model established in Section 2.8.
- Sub-step 3.5: Design the analyst-facing interface, using the dashboard design dimensions of Sarikaya et al. (2019) and the seven-stage natural-language-interface model of Shen et al. (2023).

Step 4: Implementation of the Minimum Viable Product

- Sub-step 4.1: Establish the offline development environment and verify that every dependency can be installed and every model obtained without network access at run time.
- Sub-step 4.2: Implement document ingestion and text extraction for the required formats.
- Sub-step 4.3: Implement the entity-recognition and relation-extraction components, selecting locally hostable models and fine-tuning them for the domain and language following the procedure demonstrated by Xing and Chen (2024).
- Sub-step 4.4: Implement the summarisation component.
- Sub-step 4.5: Implement the search component over a local vector index.

- Sub-step 4.6: Implement the event-timeline and dashboard visualisations.
- Sub-step 4.7: Implement authentication, authorisation and audit logging.

Step 5: Evaluation

- Sub-step 5.1: Assemble an evaluation corpus of representative, unclassified documents and annotate a subset of it to serve as ground truth.
- Sub-step 5.2: Define the task-level metrics: precision, recall and F1 for entity recognition and relation extraction; a recall-oriented overlap metric for summarisation; retrieval metrics for search.
- Sub-step 5.3: Measure the performance of each component against the ground truth.
- Sub-step 5.4: Evaluate the system against the ISO/IEC 25010 quality model, instantiating the quality-in-use characteristics as measures following Barletta et al. (2023), and weighting them by expert judgement as in Trichkova-Kashamova (2021).
- Sub-step 5.5: Conduct a usability assessment with representative users.

Step 6: Analysis and synthesis

- Sub-step 6.1: Analyse the results of Step 5 against the research objectives stated in Section 1.2.
- Sub-step 6.2: State the limitations of the study and the threats to the validity of its conclusions.
- Sub-step 6.3: Derive recommendations and identify future work.

4. System Development

4.1. Introduction

This chapter reports the execution of Steps 2, 3 and 4 of the methodology. Section 4.2 presents the analysis: the stakeholders, the functional requirements expressed as use cases, and the non-functional requirements mapped onto ISO/IEC 25010. Section 4.3 presents the design: the architectural style, the layered decomposition, the data model and the security layer. Section 4.4 presents the implementation of the Minimum Viable Product. Section 4.5 summarises the chapter. The evaluation of what is built here is deferred to Chapter 5.

4.2. Analysis

The analysis is grounded in the requirements-elicitation exercise carried out by the author in the operational setting described in Chapter 1. The full elicitation record is reproduced in Appendix A. The findings summarised in this section are the author's own analysis of that setting and are presented as such; they are not attributed to any published source, because none exists for this organisation.

Stakeholders. Four groups were identified.

- Intelligence analysts, who are subject-matter experts and who currently perform all knowledge extraction manually. They are the primary users.
- Operational and command decision-makers, who consume synthesised intelligence and who are currently deprived of cross-document synthesis.
- The information-technology and support function, which must operate and maintain the system without external vendor access.
- End users with limited technical training, whose ability to query the system determines whether it is adopted at all.

Functional requirements. Fourteen use cases were derived and are listed in Table 3; each is stated as an actor-goal pair together with its priority. Figure 5 presents the same use cases in UML notation, showing the three actor roles, the system boundary, and the associations between them. The three use cases drawn with a dashed outline are those deferred from the Minimum Viable Product, for the reasons given below.

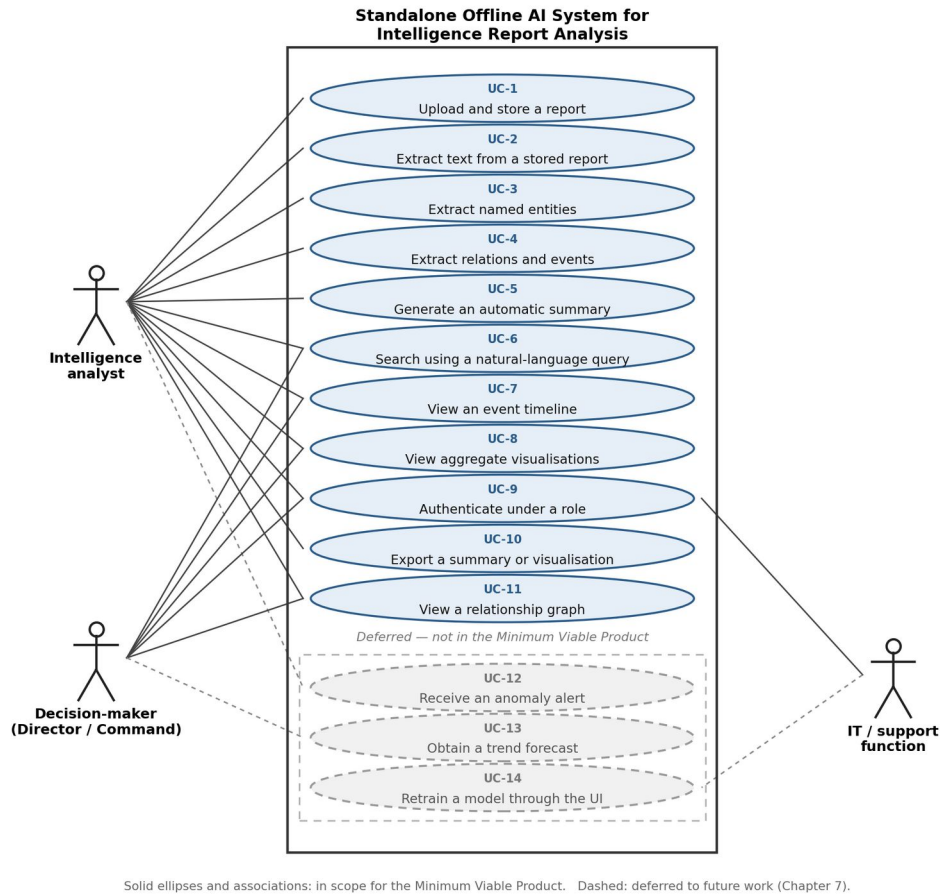


Figure 5 — Use case diagram of the proposed system. Author's own diagram, derived from the use cases of Table 3.

Table 3 — Functional requirements expressed as use cases

ID	Use case	Primary actor	Priority
UC-1	Upload and store a report in any supported format	Analyst	High
UC-2	Extract text from a stored report	System	High
UC-3	Extract named entities from a report	System	High
UC-4	Extract relations and events between entities	System	High
UC-5	Generate an automatic summary of a report	System	High
UC-6	Search the corpus using a natural-language query	Analyst	High

UC-7	View an event timeline derived from extracted dates and events	Analyst	High
UC-8	View aggregate visualisations of the corpus	Analyst, Decision-maker	High
UC-9	Authenticate and operate under a role-based permission set	All users	High
UC-10	Export a generated summary or visualisation	Analyst	Medium
UC-11	View a relationship graph across documents	Analyst	Medium
UC-12	Receive an alert on an anomalous pattern	Analyst	Low — excluded from MVP
UC-13	Obtain a forecast of a trend	Decision-maker	Low — excluded from MVP
UC-14	Retrain a model through the user interface	IT function	Low — excluded from MVP

Use cases UC-12 through UC-14 are deferred. The exclusion of UC-12 and UC-13 is not merely a matter of effort: the literature on AI in military decision-making warns specifically about the risks of systems that recommend or predict (Rivera et al., 2024; Probasco et al., 2025), and the design decision taken here is that the system extracts, summarises and visualises, but does not score, rank or recommend. This boundary is defended in Section 6.3.

Non-functional requirements. These are expressed against the characteristics of ISO/IEC 25010, whose international standing Rojas et al. (2025) document and which Trichkova-Kashamova (2021) applies specifically to information systems.

Table 4 — Non-functional requirements mapped to ISO/IEC 25010

ISO/IEC 25010 characteristic	Requirement	Rationale and source
Security	The system shall operate with no outbound network connectivity. Authentication shall precede every resource access; all actions shall be logged.	Zero trust principles (Rose et al., 2020); LLM and RAG leakage risks (Das et al., 2025; Zeng et al., 2024); air-gap threat model (Guri & Elovici, 2018; Hussain et al., 2025; Saeed et al., 2025)
Performance efficiency	Model inference shall complete within an	Feasibility of single-machine inference and fine-

	operationally acceptable time on a single workstation-class machine with one GPU.	tuning established by distillation and quantisation (Sanh et al., 2019; Dettmers et al., 2023)
Usability	Analysts without technical training shall be able to query the corpus in natural language.	Seven-stage classification of natural-language interfaces to visualisation (Shen et al., 2023); dashboard design dimensions (Sarıkaya et al., 2019)
Maintainability	The system shall be decomposed into independently replaceable modules, so that a model can be substituted without changing the surrounding system.	Quality-attribute tactics for microservices architecture, adapted to a layered monolith (after S. Li et al., 2021)
Functional suitability	Entity extraction shall achieve a level of accuracy sufficient for analyst use on the target corpus.	Threshold to be set empirically in Chapter 5; comparable domain results in Xing and Chen (2024)
Reliability	Failure of any single processing component shall not result in loss of a stored document.	Author's design decision

4.3. Design

Architectural style. A modular, layered architecture was selected. The alternative considered was a microservices decomposition; S. Li et al. (2021) identify scalability as the most widely acknowledged benefit of microservices, alongside the independent deployment and distribution of small services; none of these is a driver in a single-machine, single-tenant, air-gapped installation, while the associated monitorability and testability costs are real. A layered monolith with strictly separated modules therefore retains the maintainability benefit — the ability to replace a model without disturbing its neighbours — without incurring the operational overhead. This is a design decision, and it is revisited in Section 6.3.

Layers. Figure 6 presents the architecture. The analyst interacts only with the presentation layer; every subsequent stage — application, AI processing, data and security — executes on the isolated host, and the dashed boundary in the figure marks the extent of that host. No arrow crosses that boundary outwards, which is the diagrammatic expression of the offline requirement established in Section 2.8. Table 5 states the responsibility, the technology and the justification of each layer.

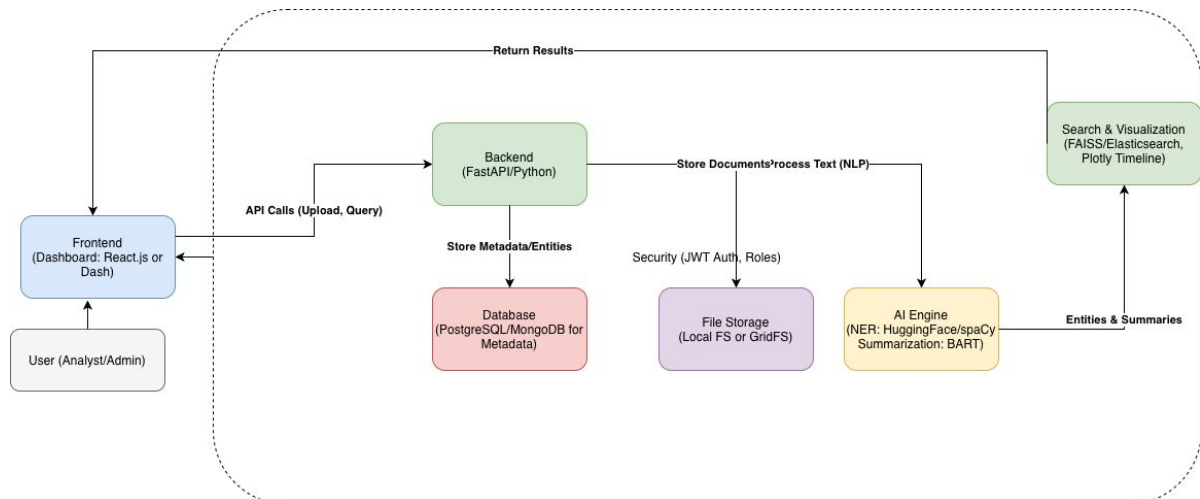


Figure 6 — Layered architecture of the proposed standalone system. Author's own diagram, derived from the layer responsibilities of Table 5.

Table 5 — Layered architecture of the proposed system

Layer	Responsibility	Technology	Justification
Presentation	Upload, natural-language query, timeline and dashboard views	Local web dashboard	Analyst-facing views specified using the dashboard design dimensions of Sarikaya et al. (2019); query interface follows the seven-stage model of Shen et al. (2023)
Application	Request routing, orchestration of the processing pipeline	Python web framework, served locally	Author's design decision; no external dependency at run time
AI processing	Text extraction, named entity recognition, relation and event extraction, summarisation, embedding	Locally hosted transformer models	Transformer architecture (Vaswani et al., 2017; Devlin et al., 2019); local feasibility (Sanh et al., 2019; Dettmers et al., 2023); Greek-language models (Koutsikakis et al., 2020; Voukoutis et al., 2024)
Data	Storage of documents, extracted entities,	Local relational store and local vector index	Retrieval grounding follows Lewis et al. (2020), with the

	relations, events and vector index		retrieval database held entirely on the isolated host and role filtering applied at retrieval time in view of Zeng et al. (2024)
Security	Authentication, role-based authorisation, audit logging, egress prevention	Local identity store; host-level egress controls	Zero trust architecture (Rose et al., 2020); data leakage prevention (Alneyadi et al., 2016)

Components and interfaces. Figure 7 decomposes each layer of Figure 6 into the components that realise it. The decomposition is driven by the maintainability requirement of Table 4: each component encapsulates one model or one store, so that either may be replaced without disturbing its neighbours.

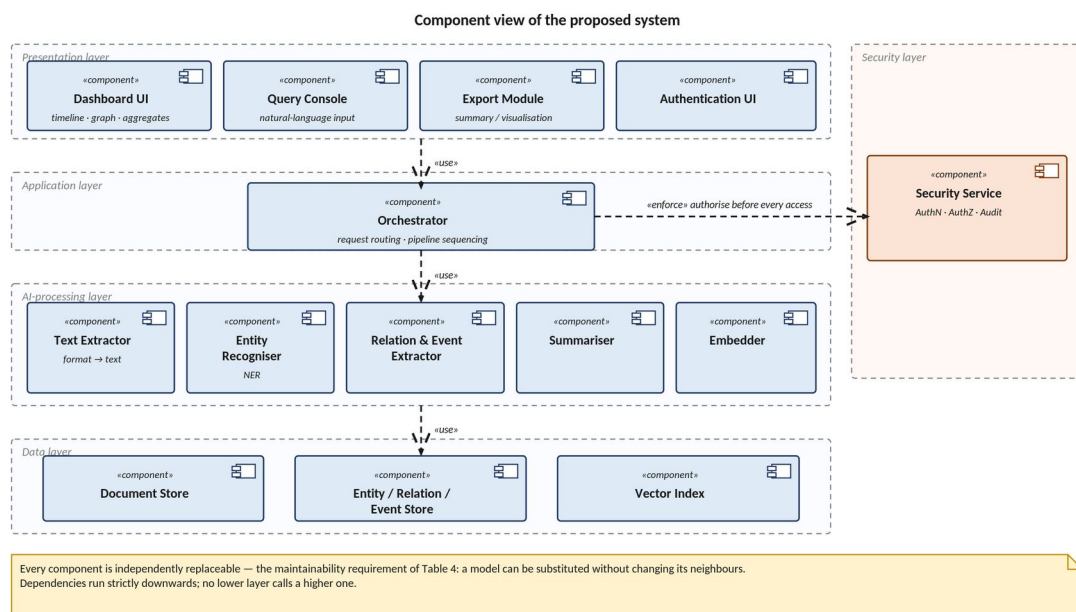


Figure 7 — Component view of the proposed system. Author's own diagram, derived from the layer responsibilities of Table 5.

Figure 8 states the same decomposition as a package structure and makes the dependency rule explicit: dependencies run strictly downwards, and no lower package imports a higher one. This is what makes the layering enforceable rather than merely descriptive, and it is what allows each layer to be tested in isolation.

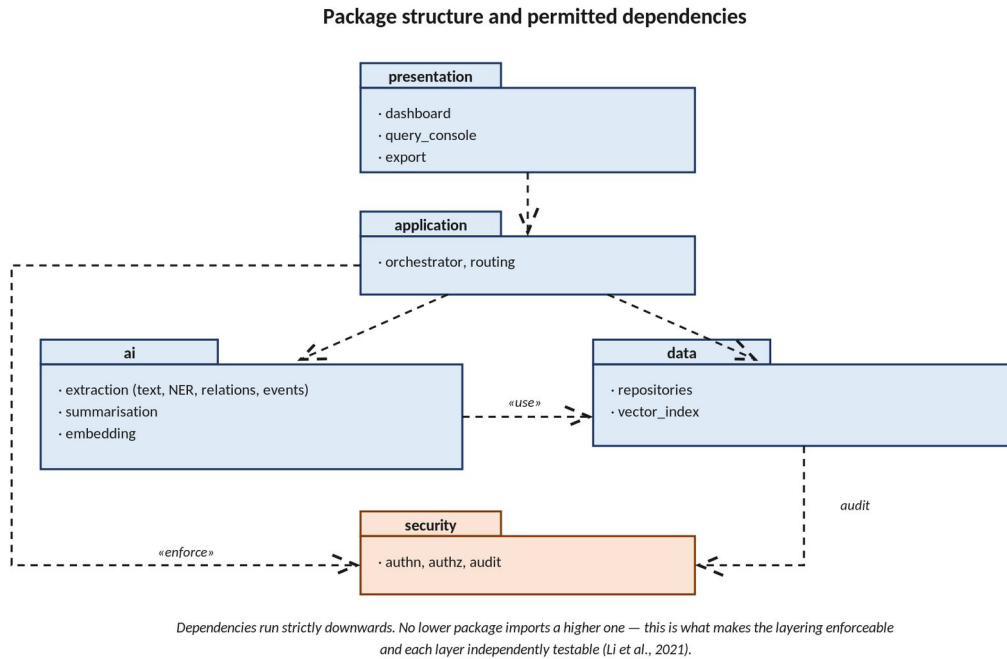


Figure 8 — Package structure and permitted dependencies. Author's own diagram.

Data model. The data model records, for each ingested document, its provenance and format; for each document, the entities extracted from it, each with a type, a character offset and a confidence score; for each pair of entities, any relation extracted between them; and for each event, its participants and its temporal anchoring. The separation of entities, relations and events follows the task decomposition established in Sections 2.4 and 2.5, and the temporal anchoring of events follows Leeuwenberg and Moens (2020). Extracted records are stored in a structured form suitable for direct database insertion, following the approach demonstrated by Dagdelen et al. (2024).

Figure 9 presents that model as a UML class diagram. The attributes shown are exactly those stated in the preceding paragraph; no attribute is asserted that the design does not require.

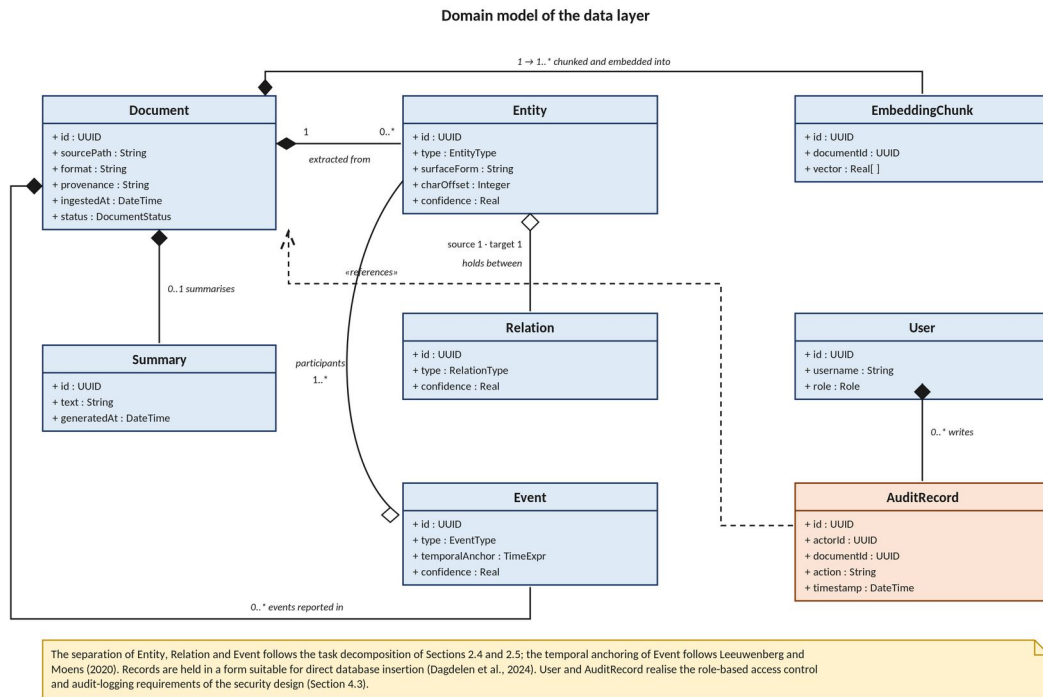


Figure 9 — Domain model of the data layer, as a UML class diagram. Author's own diagram. Processing. Figure 10 gives the order in which those records are produced. The sequence of stages is the data flow adopted in Chapter 2 — text extraction, entity recognition, relation and event extraction, summarisation, embedding — and the swimlanes show which layer of Figure 6 is responsible for each. Authorisation precedes storage, in accordance with the zero-trust principle of Figure 4.

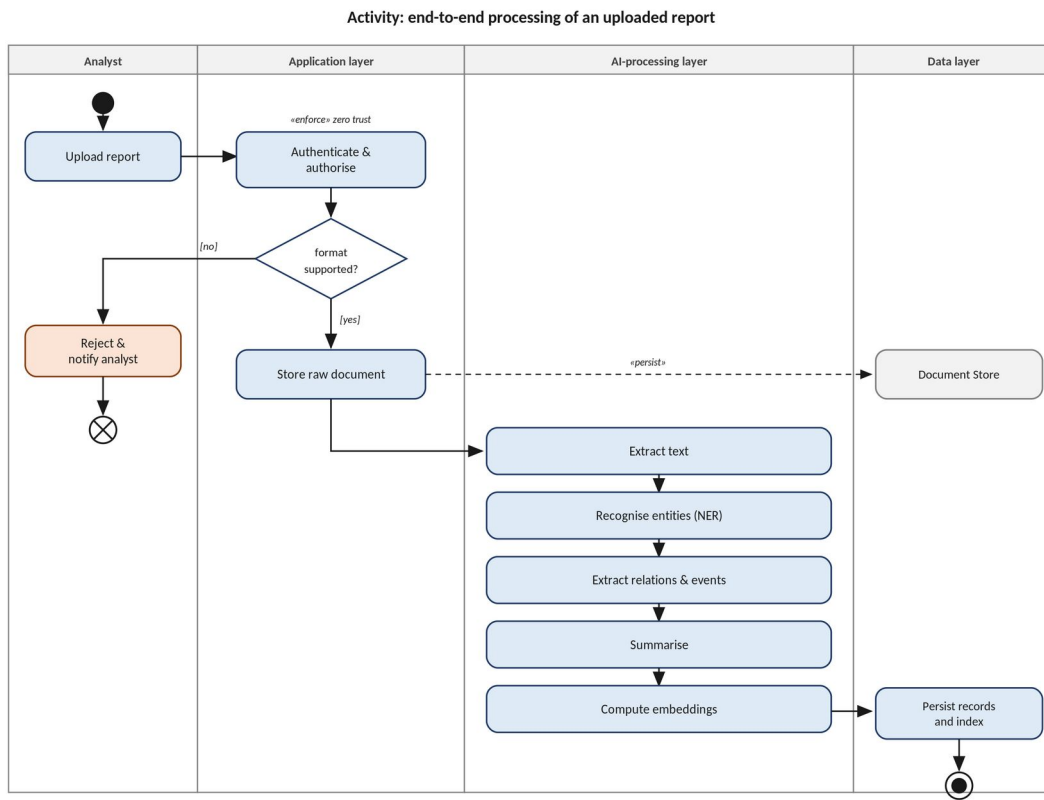
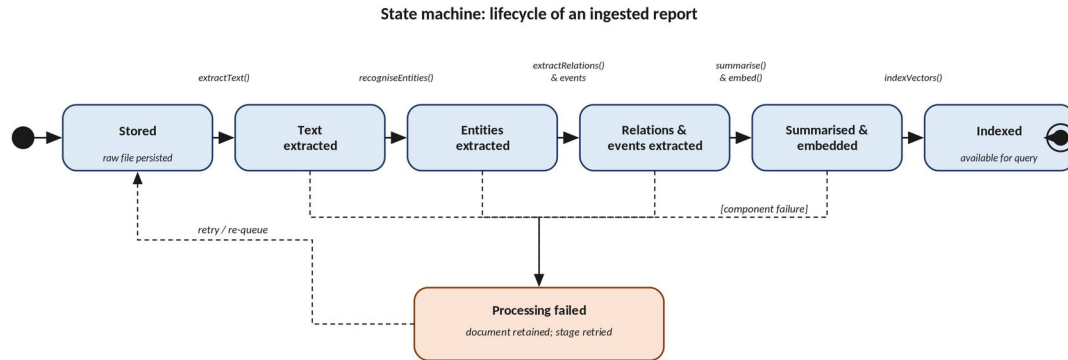


Figure 10 — Activity: end-to-end processing of an uploaded report. Author's own diagram.

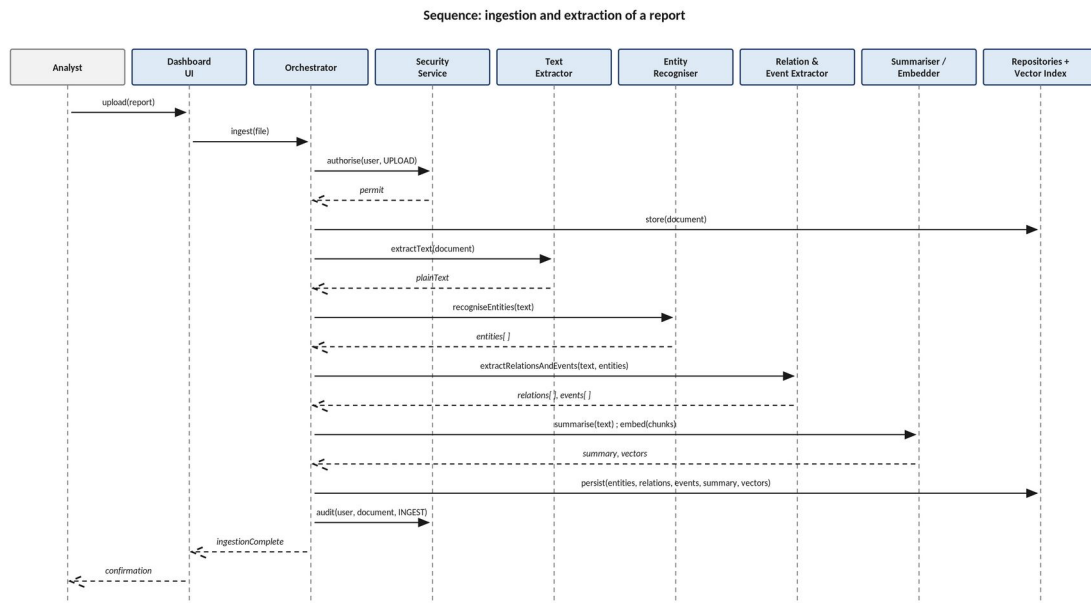
Figure 11 states the same process as the lifecycle of a single document. It is included because it makes one requirement visible that the activity diagram does not: the failed state satisfies the reliability requirement of Table 4, which is that the failure of any single processing component must not cause the loss of a stored document. The document is retained and the failing stage alone is retried.



The failed state satisfies the reliability requirement of Table 4: failure of any single processing component must not result in the loss of a stored document. The document remains in the Document Store and the failing stage alone is retried; no earlier stage is repeated.

Figure 11 — State machine: lifecycle of an ingested report. Author's own diagram.

Interactions. Figures 12 and 13 show the two interactions that characterise the system. Figure 12 is the ingestion path: the analyst uploads a report, authorisation is obtained before the document is stored, each processing stage is invoked through the orchestrator rather than by its neighbour, and the ingestion is written to the audit log.



Authorisation precedes every resource access, and the ingestion is written to the audit log — the zero-trust and accountability requirements of Table 4 (Rose et al., 2020; Meerveld et al., 2023). Each processing stage is invoked through the orchestrator, so that any model may be substituted without altering the components on either side of it.

Figure 12 — Sequence: ingestion and extraction of a report. Author's own diagram.

Figure 13 is the query path. It instantiates the retrieval-augmented generation pattern of Figure 1 within the seven-stage model of Figure 2. Two design decisions are visible in it. The role filter is applied at retrieval time rather than after generation, because Zeng et al. (2024) show that the retrieval database can itself be extracted through crafted queries; and the generated answer carries its provenance, so that the analyst can always reach the source report behind a statement.

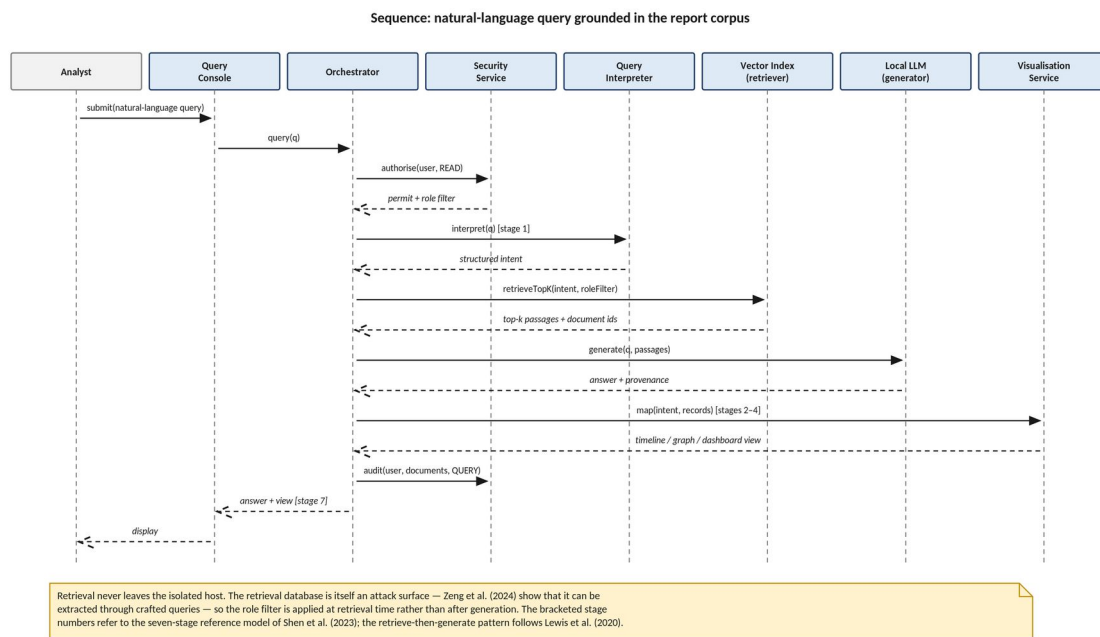


Figure 13 — Sequence: natural-language query grounded in the report corpus. Author's own diagram; the retrieve-then-generate pattern follows Lewis et al. (2020) and the stage sequence follows Shen et al. (2023).

Security design. The security layer implements three controls. Access control follows the role-based model of Sandhu et al. (1996) and is enforced before any resource is reached, in accordance with the zero-trust principle that no implicit trust follows from network location (Rose et al., 2020). Egress prevention is enforced at host level, on the basis that isolation must be an enforced control rather than an assumption; the air-gap attack literature (Carrara & Adams, 2016; Guri, 2024; Hussain et al., 2025) establishes that a compromised isolated host can exfiltrate data through covert physical channels, and Saeed et al. (2025) locate the corresponding countermeasures in the organisation's information-security management system, proposing them as an ISO 27001 control — which places them outside the application. Audit logging records every action against every document, supporting the explainability and meaningful-human-control requirements that the responsible-AI literature places on military decision support (Meerveld et al., 2023).

Deployment. Figure 14 shows the installation. Every component executes on a single isolated workstation; models and dependencies are obtained and verified on a provisioning host before isolation, which is the procedure specified in Sub-step 4.1, and no arrow crosses the boundary outwards at run time. The threats of Figure 3 are not eliminated by that boundary — they are addressed by the host-level egress controls named in the figure and, following Saeed et al. (2025), at the level of the organisation's information-security management system, outside the application.

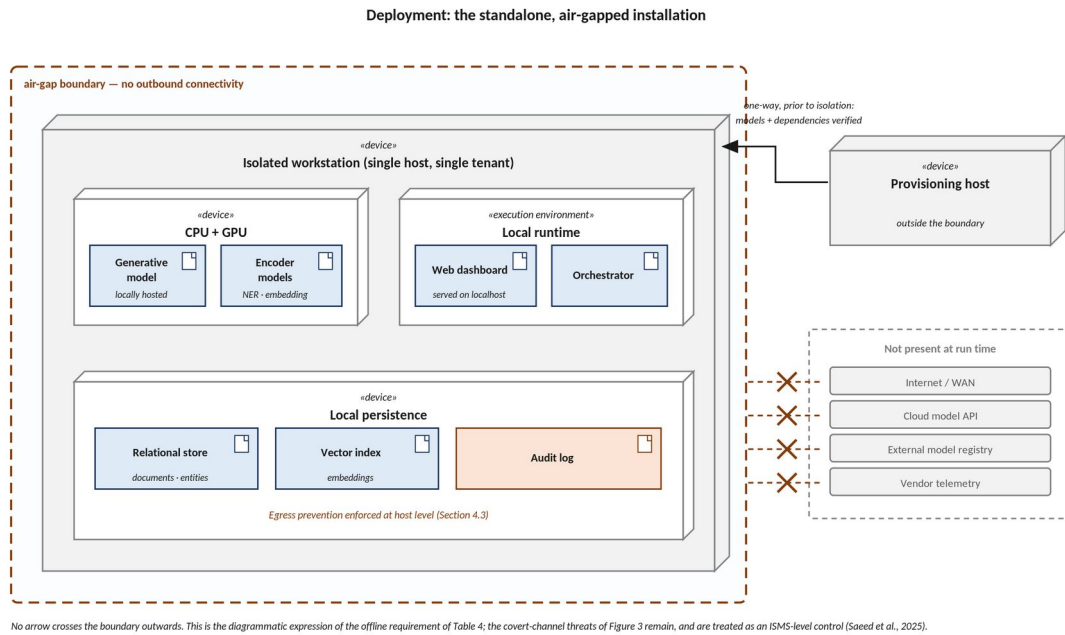


Figure 14 — Deployment: the standalone, air-gapped installation. Author's own diagram; the countermeasure position follows Saeed et al. (2025).

4.4. Implementation

This section reports the implementation of the Minimum Viable Product. The scope of the Minimum Viable Product was determined in Sub-step 2.5 by prioritising the use cases of Table 3; the full scope definition is reproduced in Appendix B, and the implementation plan in Appendix C. The artefact is a working system, held in a private Git repository under the name Kalhas, and every statement made in this section is traceable to a file in that repository. Where the artefact departs from the design of Section 4.3, the departure is stated rather than concealed; Section 4.4.9 summarises those departures and Figure 22 presents them in full. The code was written with the assistance of generative AI coding tools; the extent of that assistance, and the one component that was generated entirely by one of them, are declared in the statement that precedes the Abstract.

4.4.1. Environment

The system is delivered as a Docker Compose composition of three containers, all built for the linux/arm64 platform and executed on an Apple Silicon host. Figure 15 shows the deployment as it was actually built. The rag-frontend container runs a Streamlit application on port 8501; the rag-backend container runs a FastAPI application, written in Python (van Rossum & the Python development team, 2018) and served by uvicorn on port 8000; the rag-ollama container runs an Ollama server on port 11434 and performs all generative inference. The three containers communicate over a single Docker bridge network. Persistence is provided by four bind-mounted volumes: ./data holds the uploaded documents and the SQLite database, ./embeddings holds the FAISS index and its metadata, ./models holds the cached sentence-transformers and spaCy models, and ./ollama holds the language-model weights.

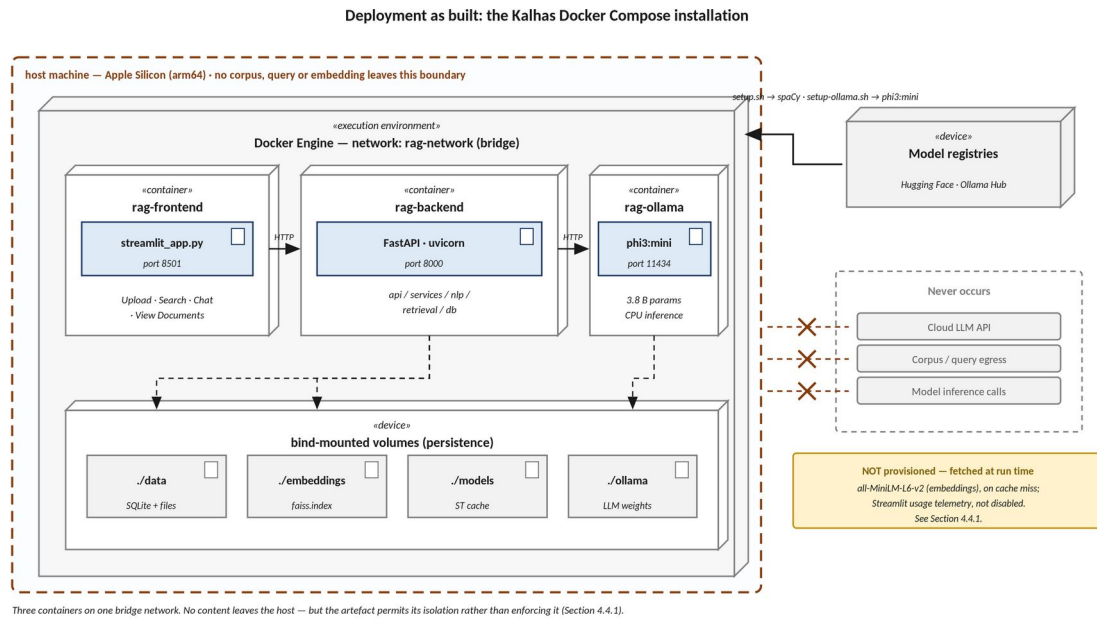


Figure 15 — Deployment as built: the Kalhas Docker Compose installation. Author's own diagram, drawn from docker-compose.yml, the two Dockerfiles and the setup scripts.

Inference is performed on the CPU throughout. No GPU is requested by any container and none is used. This is a departure from the non-functional requirement of Table 4, which anticipated a workstation-class machine with one GPU, and it is the consequence of selecting models small enough not to need one — a tactic supported by Sanh et al. (2019), who establish the feasibility of running distilled models on constrained hardware, and by Dettmers et al. (2023), who establish that even large models can be adapted on a single GPU.

Two setup scripts obtain models before the system is isolated. `setup.sh` installs the spaCy English model from a pinned release URL into the backend container; `setup-ollama.sh` pulls the `phi3:mini` weights through the Ollama client into the `./ollama` volume. This is the provisioning procedure specified in Sub-step 4.1 and drawn in Figure 14, and to that extent the artefact confirms that it is workable.

The provisioning is nonetheless incomplete, and the gap is the most important defect found in the artefact. Neither setup script, neither Dockerfile and no entry in the composition obtains the embedding model. The `EmbeddingGenerator` class constructs its sentence-transformer with the `./models` directory as a cache folder, and with an empty cache the library resolves that miss by downloading the model from a remote model registry — at run time, on the first document processed or the first query asked. A second and lesser instance of the same class of defect is that the Streamlit front end is started without disabling its usage telemetry and the repository contains no configuration file that would disable it.

The consequence must be stated precisely, because it is easy to overstate in either direction. The artefact does not leak the corpus: no report, no chunk, no query and no embedding is transmitted anywhere, and the retrieval index, the database and the language model are all local. What the artefact fails to do is enforce its own isolation. It permits the offline property rather than guaranteeing it: on a host with no route to the internet, the two outbound calls simply fail — the embedding model then cannot load at all, and the pipeline silently degrades to zero vectors — whereas on a host that still has a route, they succeed. This is the exact failure mode against which the air-gap literature warns. Guri and

Elovici (2018) observe that air-gapped networks have repeatedly been compromised despite their physical isolation, and Saeed et al. (2025) argue that the enforcing controls belong in the organisation's information-security management system. In this artefact they are not present at the level of the application either. Closing the gap is a small change — warm the embedding cache in setup.sh and disable telemetry in the front-end container — and Section 6.3 records it as a limitation rather than a design flaw, because the architecture is not what is wrong.

4.4.2. Document ingestion and text extraction

Ingestion is exposed as POST /api/upload. The router writes the uploaded file to the ./data volume, records a Document row with its filename, path, size, content type and a status of “uploaded”, and returns the assigned identifier. Processing is a separate, explicit call, POST /api/process, which is what allows the analyst to upload a batch and then process it. Text extraction is implemented in the TextExtractor class and dispatches on the file extension. PDF is handled by PyMuPDF, which iterates the pages and concatenates the extracted text; DOCX is handled by python-docx, which joins the paragraph texts; plain text and Markdown are read directly. Four formats are therefore supported: PDF, DOCX, TXT and MD. An unsupported extension is logged as a warning and yields an empty string rather than an exception, so that a malformed document cannot abort a batch — the behaviour required by the reliability requirement of Table 4 and drawn as the failed state of Figure 11.

Figure 16 sets out the implemented module structure. It is the as-built counterpart of the component view of Figure 7, and the correspondence is close: the layers survived contact with the implementation, and no module crosses a layer boundary that the design did not permit.

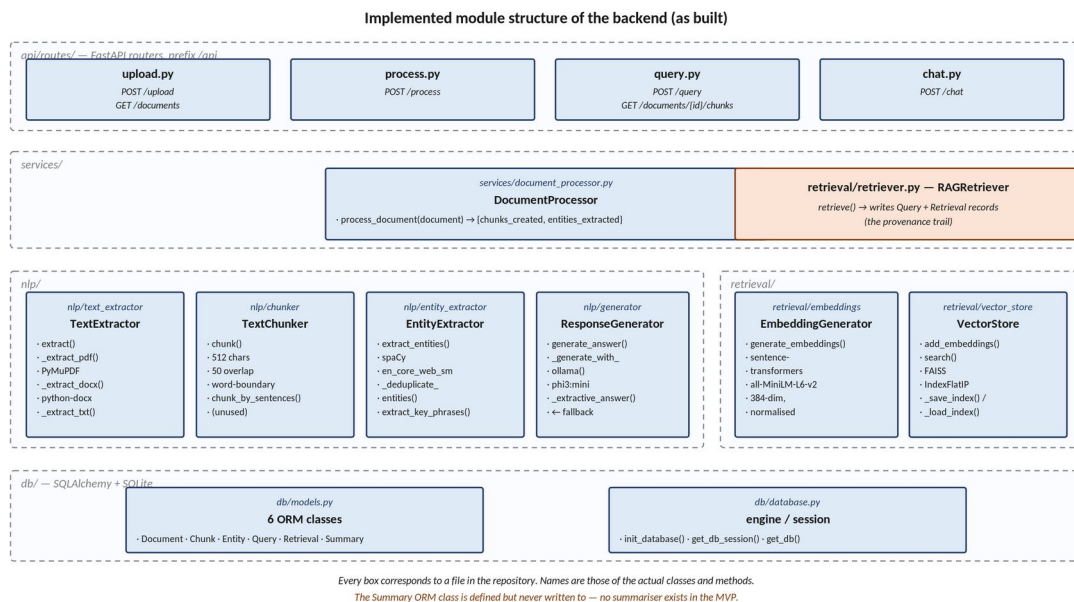


Figure 16 — Implemented module structure of the backend. Author's own diagram; every box corresponds to a file in the repository and the names are those of the actual classes and methods.

4.4.3. Chunking and embedding

The TextChunker class produces overlapping windows of 512 characters with an overlap of 50 characters. The window boundary is moved back to the last space before the cut, so that a chunk never ends in the middle of a word. Each chunk carries its character offsets within the source document, and those offsets are persisted; this is the mechanism by which a retrieved passage can be located in the original report. A second, sentence-based strategy is present in the class but is not called by the pipeline.

Embeddings are produced by the EmbeddingGenerator class, which wraps the sentence-transformers model all-MiniLM-L6-v2. The model returns 384-dimensional vectors and is instructed to normalise them, which is what makes an inner-product index equivalent to cosine similarity. The model is cached in the ./models volume, and if it cannot be loaded the class returns zero vectors rather than raising — a degradation that keeps the pipeline running but silently destroys retrieval quality, and which is noted here as a defect rather than a feature.

4.4.4. Entity extraction

The EntityExtractor class wraps the spaCy pipeline en_core_web_sm and is applied to the full extracted text of each document, up to a limit of one million characters. It returns, for each entity, its spaCy label, its surface form, and fifty characters of surrounding context on each side. Entities are then deduplicated on the pair (label, lower-cased text), so that a person named twenty times in a report is stored once. The confidence field is written as a constant 1.0 because spaCy does not expose a confidence score for its entity recogniser; the field is therefore present in the schema but carries no information, and should not be read as a measure of certainty.

Two consequences must be stated plainly. First, the model is English. The corpus that motivates this work is not in English, and the Greek-language models identified in Section 2.3 as the correct choice — GREEK-BERT (Koutsikakis et al., 2020) and Meltemi (Voukoutis et al., 2024) — are not used by the artefact. Second, the extracted entities are persisted but are never surfaced: no endpoint returns them and no view in the Streamlit application displays them. Use case UC-3 is therefore implemented in its extraction half and not in its presentation half.

4.4.5. Retrieval and the vector index

The VectorStore class wraps a FAISS IndexFlatIP index. IndexFlatIP performs an exhaustive inner-product search, which over the normalised vectors produced by the embedding stage is exact cosine similarity: there is no approximation and therefore no recall loss from the index itself. The index is written to disk after every insertion, together with a pickled metadata structure that maps each position in the index to the chunk identifier and document identifier it came from. It is this mapping that allows a FAISS hit to be resolved back to a passage and a source report.

The RAGRetriever class implements the query path. It encodes the query with the same embedding model used for the documents, searches the index, and — before returning anything — writes a Query row and one Retrieval row per hit, each recording the chunk identifier, the similarity score and the rank. The retrieval provenance of every answer the system has ever given is therefore reconstructable from the database. This exceeds what the design of Section 4.3 required and is the strongest property of the artefact; it is what makes the explainability that Meerveld et al. (2023) require of military decision support —

knowing what a decision was based on — auditable in practice rather than merely asserted.

Figure 17 gives the implemented schema. Six tables are defined. Five are written to; the sixth, summaries, is defined and never populated.

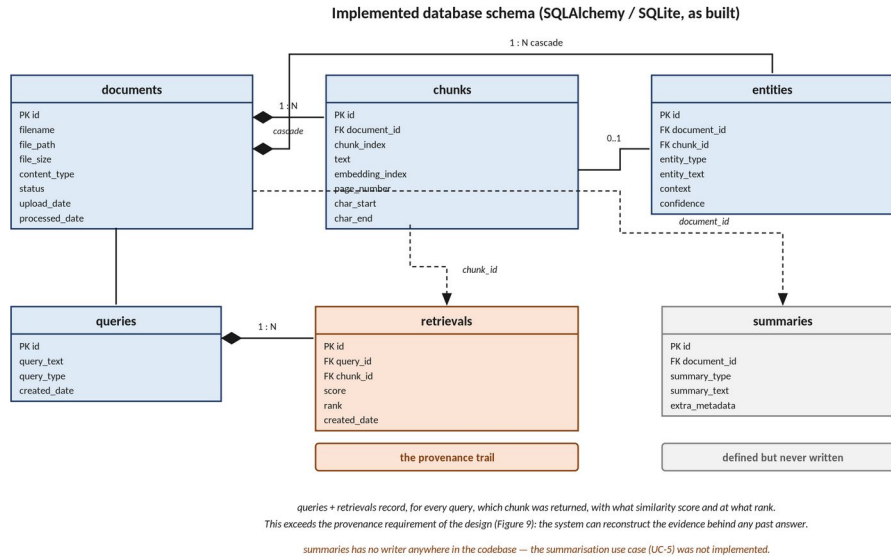


Figure 17 — Implemented database schema. Author's own diagram, drawn from backend/db/models.py.

An optional document-identifier filter is applied after the FAISS search rather than within it. The index is searched for twice the requested number of results and the filter is then applied to the hits, which means that a sufficiently selective filter can return fewer than the requested number of chunks. This is a known limitation of the implementation and is recorded as such.

4.4.6. Answer generation

The ResponseGenerator class calls the local Ollama server. The default model is phi3:mini, a 3.8-billion-parameter model chosen because it runs on the CPU of the target machine. The retrieved chunks are concatenated up to a ceiling of 2 000 characters and inserted into a prompt that instructs the model to answer only from the supplied context, to be specific, and to say so when the context is insufficient. Generation runs at temperature 0.3 with a limit of 200 predicted tokens. The source documents and their similarity scores are appended to the generated answer. The fallback path appends the source document name but not the score, and the empty-result path returns a bare message with no attribution; the provenance shown to the analyst is therefore complete on the generative path and partial on the other two, although the full record is written to the database in every case.

The class implements an explicit fallback. If the Ollama client cannot be constructed, or if generation raises, the class calls `_extractive_answer`, which returns the highest-ranked retrieved chunk verbatim together with its source attribution. The system therefore degrades from abstractive to extractive answering rather than failing, and in doing so it degrades towards the safer behaviour: an extracted passage cannot hallucinate. Figure 18 traces this path through the code.

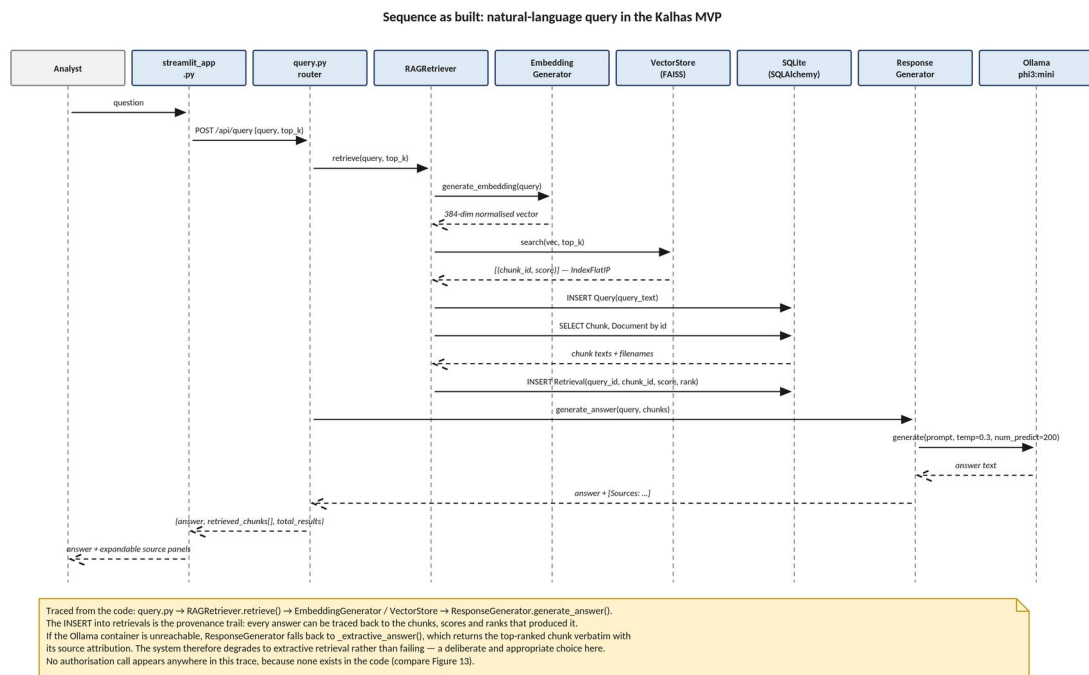


Figure 18 — Sequence as built: natural-language query in the Kalhas MVP. Author's own diagram, traced from api/routes/query.py, retrieval/retriever.py and nlp/generator.py.

4.4.7. The analyst interface

The Streamlit application presents four views: Upload and Process, Search Documents, Chat Interface, and View Documents. The search view returns the generated answer together with an expandable panel for each retrieved chunk showing the source document, the similarity score and the passage text. The chat view maintains a conversation history in the session and displays the top three sources beneath each response. The document view lists the corpus with each document's status.

Figures 19 to 21 show the application as deployed, with a document of the validation corpus of Section 5.2.1 loaded. Figure 19 is the upload view, captured while a document is being processed: the file has been stored and the processing call of Section 4.4.2 is running. Figure 20 is the chat view after a question has been answered. The generated text is followed, in the same message, by the names of the three source documents and their similarity scores, which is the attribution described in Section 4.4.6. Figure 21 shows the same view with the source panel expanded, in which the retrieved passages are displayed verbatim beneath their scores, so that the analyst can read the evidence rather than only its citation. The display title of the Streamlit application is “RAG System”; Kalhas is the name of the repository.

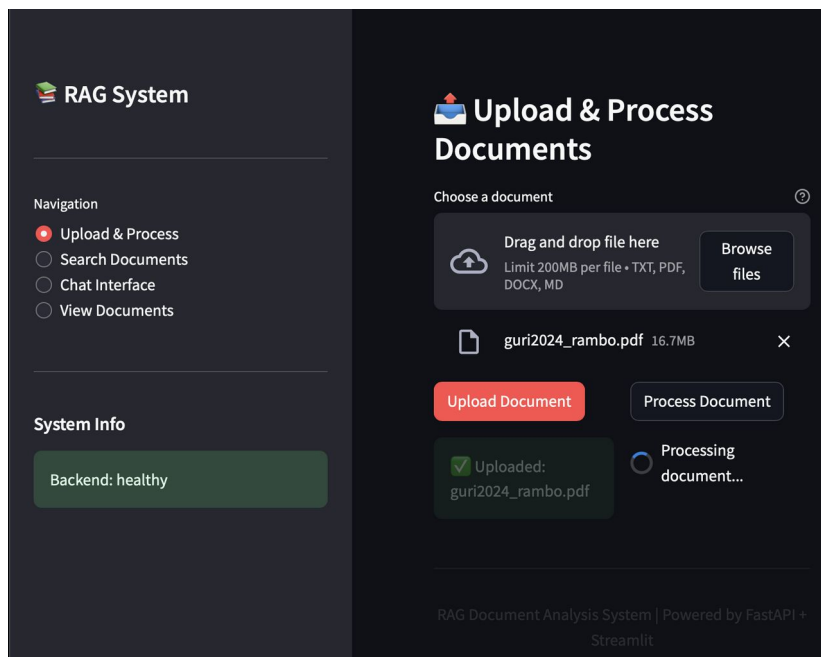


Figure 19 — Upload and Process view of the Kalhas front end, captured while guri2024_rambo.pdf is being processed. Screenshot of the deployed Streamlit application.

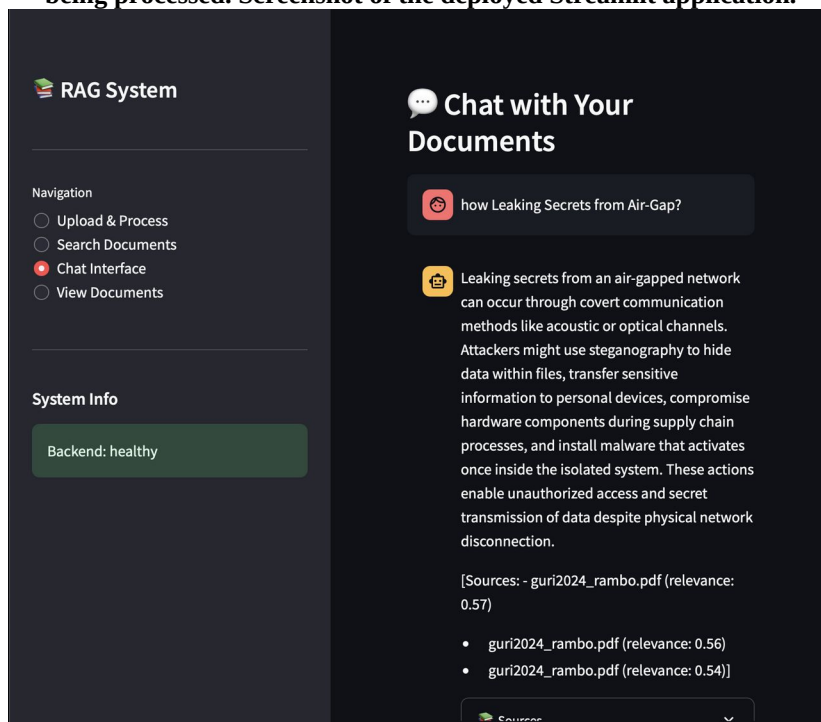


Figure 20 — Chat view after a grounded answer. The generated text is followed by the three source attributions with their similarity scores. Screenshot of the deployed Streamlit application.

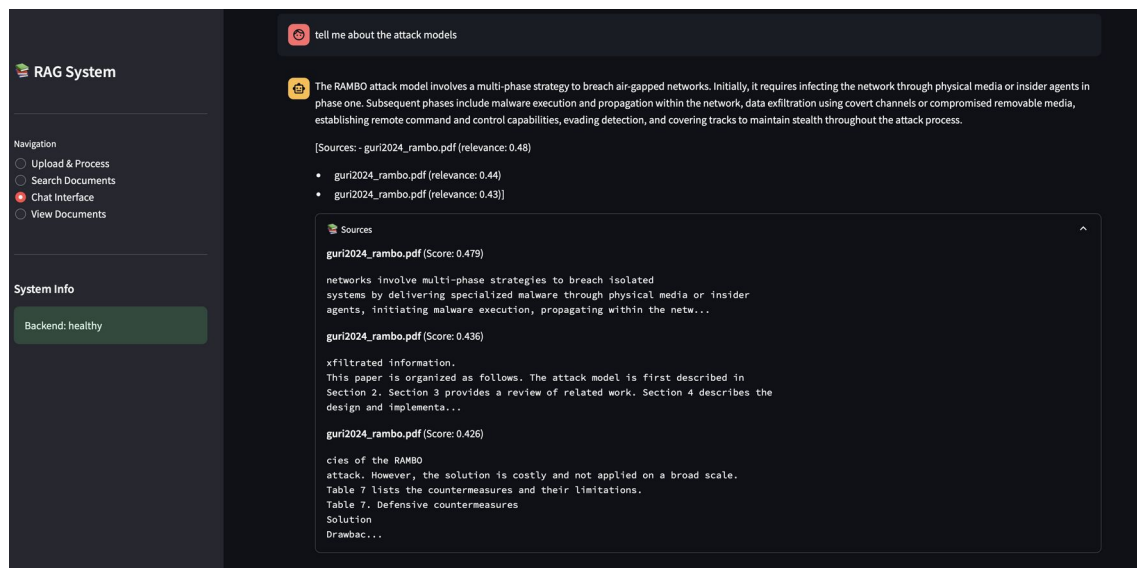


Figure 21 — Chat view with the source panel expanded: the retrieved passages are shown verbatim beneath their similarity scores. Screenshot of the deployed Streamlit application.

What the interface does not contain must be stated, because it is the largest gap between the design and the artefact. There is no event timeline, no entity graph, no aggregate visualisation and no dashboard. The design of Section 4.3 specified all four, drawing on the dashboard design dimensions of Sarikaya et al. (2019) and the timeline-versus-animation analysis of Beck et al. (2017); none was built. The visualisation pillar of the design — one of the three named in the title of this dissertation — is not present in the Minimum Viable Product.

4.4.8. Security controls

The security layer designed in Section 4.3 was not implemented. There is no authentication, no authorisation and no audit logging in the artefact. No endpoint requires a credential; the FastAPI application is configured with a permissive cross-origin policy that accepts any origin; and no User, Role or AuditRecord table exists in the schema. The project's own validation document records the position explicitly: its checklist lists the absence of authentication, of encryption and of rate limiting as accepted properties of the Minimum Viable Product, and places audit logging among the requirements deferred to a production system.

The zero-trust principles of Rose et al. (2020), drawn in Figure 4 and specified in Section 4.3, are therefore not realised by the artefact. What the artefact does provide is the property that motivated them: the corpus never leaves the host. No component makes an outbound call at run time, the retrieval index and the report corpus are held entirely on the isolated machine, and the generative model is served locally. The corpus is therefore protected against leakage to any party outside the host; it is not protected against the query-based extraction of the retrieval database that Zeng et al. (2024) demonstrate, which any user able to query the system could attempt, nor against any actor with access to the host. This is a coherent position for a feasibility prototype and an untenable one for a deployed system, and Section 6.3 treats it as such.

4.4.9. What was built and what was not

Figure 22 sets the fourteen use cases of Table 3 against the artefact. Of the eleven use cases placed within the scope of the Minimum Viable Product, three are fully implemented — upload and storage (UC-1), text extraction (UC-2), and natural-language search over the corpus (UC-6) — and one is partially implemented: named entity recognition (UC-3) extracts and persists entities but never presents them. Seven are not implemented: relation and event extraction (UC-4), summarisation (UC-5), the event timeline (UC-7), aggregate visualisations (UC-8), authentication and role-based permissions (UC-9), export (UC-10), and the relationship graph (UC-11). The three use cases deferred from the Minimum Viable Product by design (UC-12 to UC-14) remain deferred.

Use-case coverage: designed scope versus implemented artefact

ID	Use case	Priority	Status	Evidence in the code
UC-1	Upload and store a report	High	implemented	/api/upload
UC-2	Extract text from a stored report	High	implemented	nlp/text_extractor.py
UC-3	Extract named entities	High	partial	spaCy en_core_web_sm — stored, never displayed
UC-4	Extract relations and events	High	not implemented	no module exists
UC-5	Generate an automatic summary	High	not implemented	summaries table never written
UC-6	Search the corpus in natural language	High	implemented	/api/query - /api/chat (RAG)
UC-7	View an event timeline	High	not implemented	not implemented
UC-8	View aggregate visualisations	High	not implemented	not implemented
UC-9	Authenticate; role-based permissions	High	not implemented	no auth; CORS allow_origins = "
UC-10	Export a summary or visualisation	Medium	not implemented	not implemented
UC-11	View a relationship graph	Medium	not implemented	not implemented
UC-12	Alert on an anomalous pattern	Low	out of MVP scope	excluded from MVP by design
UC-13	Obtain a forecast of a trend	Low	out of MVP scope	excluded from MVP by design
UC-14	Retrain a model through the UI	Low	out of MVP scope	excluded from MVP by design

Of the 11 use cases placed in MVP scope, 3 are fully implemented and 1 is partial. The 3 deferred by design remain deferred.

This is the principal finding of the artefact: the retrieval pillar of the design was reached, the extraction-and-visualisation pillar and the security pillar were not.

Figure 22 — Use-case coverage: the designed scope of Table 3 set against the implemented artefact. Author's own analysis of the repository.

Two of these absences are visible in the schema itself. The summaries table is defined in the object-relational model but no code path anywhere in the repository writes to it; the summarisation component was designed, given a table, and never built. There is no table for relations or events at all, because there is no extractor to populate one.

The pattern is not random. The artefact implements, completely and coherently, the retrieval pillar of the design: ingestion, chunking, embedding, indexing, retrieval, grounded generation and provenance. It does not implement the structured-extraction pillar (relations, events, timelines) or the security pillar. The significance of that pattern is the subject of Chapter 6.

4.5. Summary

This chapter reported the analysis, design and implementation of the proposed system. The analysis identified four stakeholder groups, fourteen use cases of which eleven are in scope, and a set of non-functional requirements mapped onto ISO/IEC 25010. The design adopted a layered architecture, justified against a microservices alternative, and specified a data model that separates entities, relations and events, together with a security layer built on zero-trust

principles. The implementation of the Minimum Viable Product is reported in Section 4.4. The evaluation of the resulting system is the subject of Chapter 5.

5. Evaluation and Results

5.1. Introduction

This chapter reports the execution of Step 5 of the methodology. Section 5.2 states the evaluation framework and the corpus on which it was applied, Section 5.3 reports the functional validation and the measured performance of the ingestion and retrieval path, Section 5.4 assesses the prototype against ISO/IEC 25010, Section 5.5 states the position taken on the usability assessment, and Section 5.6 summarises the findings. Each result is related back to the research objectives stated in Section 1.2.

5.2. Evaluation framework

The evaluation has two levels. At the task level, the extraction and retrieval components are measured against an annotated ground truth using precision, recall and F1 for entity recognition, and known-item retrieval measures — Recall@k, mean reciprocal rank and normalised discounted cumulative gain — for search. F1-based extraction measures of this kind are those used in the comparable studies reviewed in Chapter 2 (Mo & Li, 2024; Xing & Chen, 2024; Zhang et al., 2025 — the last for relation rather than entity extraction), which allows the results obtained here to be read against them, with the caveat that the corpora differ.

At the system level, the prototype is evaluated against ISO/IEC 25010. The quality-in-use characteristics are instantiated as concrete measures following the method of Barletta et al. (2023), who propose a set of measures for each characteristic and combine them through the Analytic Hierarchy Process, and following Trichkova-Kashamova (2021), whose stated purpose in applying the standard to information systems is to minimise the subjectivity of the assessing expert.

Two constraints on the evaluation follow directly from what was built, and they must be stated before any result is reported. First, three of the components named in the evaluation plan of Sub-step 5.2 do not exist in the artefact: there is no relation or event extractor and no summariser, so relation-extraction F1 and summarisation overlap cannot be measured, now or later, against this artefact. Second, the entity recogniser that does exist is an English model applied to a domain for which no annotated Greek ground truth exists; constructing one is a labelling exercise that lies outside the scope of this dissertation. Entity-level precision and recall are therefore reported as not measurable rather than as pending.

An evaluation harness was written to make what remains measurable reproducible. It is a single script that imports the production classes from the repository without modification, ingests a declared corpus, executes a declared set of known-item queries against the resulting index, and emits a machine-readable record of the environment in which it ran together with the measured values. It is reproduced in Appendix D. The purpose of reporting the harness rather than only its output is that a measurement whose environment is not recorded is not a measurement, and the target hardware — an Apple Silicon machine — is a property of the result, not an incidental detail. The code of the harness was generated by one AI coding assistant (Claude) from the author’s specification, as declared in the front matter, and was reviewed and executed by the author. It was executed on the target host on 19 September 2026; Section 5.3 reports the outcome and Section D.5 records the two runs.

5.2.1. The evaluation corpus

The operational corpus that motivates this work is classified and cannot be used in an academic study. Olsen et al. (2024) survey the open datasets of socio-political events that serve as the legitimate proxy in such cases, and the assembly of a domain proxy corpus from that survey remains the correct next step for the task-level evaluation of extraction quality.

For the validation of the ingestion and retrieval path, a technical validation corpus of twelve open-access documents totalling 283 pages was used. The corpus was chosen for a structural property rather than a topical one: the documents are long, structured, multi-section PDFs, which is the property of the target reports that the design must accommodate; Sharma et al. (2024) and Zhao et al. (2024) identify document length as the source of difficulty in long-document processing and hierarchical section structure as the resource to be exploited. It does not stand in for the domain, and no claim about domain performance is derived from it.

5.3. Results

5.3.1. Functional validation of the ingestion path

The ingestion path was executed over the validation corpus using the production TextExtractor, TextChunker, EmbeddingGenerator and EntityExtractor classes, imported unmodified from the repository. All twelve documents were processed without error. The extraction stage recovered 972 234 characters of text from the 283 pages, and the chunking stage produced 2 147 chunks at the configured window of 512 characters with an overlap of 50 — a mean of 452.8 characters per chunk and 7.59 chunks per page. The entity recogniser returned 9 052 entities, a mean of 32.0 per page. Table 6 reports the per-document result, together with the wall-clock time of each stage. The character, chunk and entity counts are structural properties of the pipeline and the corpus and were identical in both executions of the harness; the timings are properties of the host and are discussed in Section 5.3.2.

One consequence of that measurement bears directly on the architecture, and it is a finding rather than a statistic. At 7.59 chunks per page, the twenty thousand pages per year stated in Section 1.1 correspond to approximately 152 000 chunks per year of operation. The vector index chosen in the implementation is a FAISS IndexFlatIP, which performs an exhaustive scan: it is exact, but its search cost and its memory footprint both grow linearly with the corpus. At 152 000 chunks the index occupies 233 megabytes of 384-dimensional single-precision vectors, which is unproblematic; after five years of operation it approaches three-quarters of a million chunks and more than a gigabyte, at which point an exhaustive index is the wrong structure and an approximate index of the IVF or HNSW family becomes necessary. The implementation is correct for a prototype and will not survive the volume that motivated the prototype. This is discussed in Section 6.3.

Document (validation corpus)	Characters extracted	Chunks produced	Entities recognised	Extraction (s)	Embedding (s)	Entity recognition (s)
dettmers2023_qlora	87,802	194	1,196	0.081	19.186	1.758
devlin2019_bert	64,174	142	831	0.047	19.484	1.301
guri2024_rambo	39,740	88	372	0.031	9.971	0.761
koutsikakis2020_greekbert	47,277	104	766	0.039	15.208	0.883
lewis2020_rag	69,085	153	925	0.046	18.717	1.361
rivera2024_escalation	220,545	485	1,383	0.363	46.797	4.326
rose2020_zerotrust_nist	157,095	345	533	0.093	33.612	2.359
sanh2019_distilbert	18,527	41	296	0.016	3.692	0.334
sarikaya2019_dashboards	78,682	173	575	0.069	14.348	1.351
vaswani2017_transformer	39,600	90	483	0.070	8.909	0.729
xing2024_police_reports	83,098	183	917	0.035	20.405	1.410
zeng2024_rag_privacy	66,609	149	775	0.048	12.587	1.270
TOTAL — 12 documents, 283 pages	972,234	2,147	9,052	0.938	222.916	17.843

Table 6 — Functional validation of the ingestion path on the target host: text extracted, chunks produced, entities recognised and wall-clock time per stage for each document of the technical validation corpus, using the production classes at the configured window of 512 characters with 50 characters of overlap (second run of 19 September 2026, results.json).

5.3.2. Measured performance on the target host

The harness of Appendix D was executed on 19 September 2026 inside the backend container of the Docker Compose stack described in Section 4.4.1 and shown in Figure 15. A native run on the same machine would have been faster, and the choice needs a word of justification: the containerised stack is the artefact, and a latency measured outside it would describe an arrangement that the system does not use. The environment section of the record written by the harness, reproduced as Listing D.3, reports a linuxkit aarch64 kernel with ten visible cores and 7.7 GB of memory allocated to the Docker virtual machine on the Apple Silicon host, Python 3.11.16, torch 2.1.2 with CUDA unavailable, sentence-transformers 2.3.1, FAISS 1.7.4 and spaCy 3.7.2 with the en_core_web_sm 3.7.1 model. The figures that follow are properties of that configuration and should not be transferred to another.

Table 7 gives the latency of each ingestion stage over the twelve documents, and Figure 23 shows the same times per document. Text extraction took a median of 47.8 ms per document and chunking 0.2 ms; together they account for less than half of one per cent of the total. Entity recognition took a median of 1.33 s per document, or 7.4 per cent of the total. Embedding dominates: a median of 17.0 s per document, a 95th percentile of 46.8 s for the longest document, and 92.2 per cent of the 241.7 s that the corpus required in all. The overall rate is 0.854 s per page.

Stage	n	Median	Mean	95th pct.	Min	Max
Text extraction (ms)	12	47.8	78.1	362.9	15.8	362.9
Chunking (ms)	12	0.2	0.2	0.5	0.1	0.5
Embedding (ms)	12	16,962.5	18,576.3	46,797.2	3,691.8	46,797.2
Entity recognition (ms)	12	1,326.2	1,486.9	4,325.8	333.7	4,325.8
Query encoding (ms)	75	113.4	113.8	131.8	85.1	145.6
FAISS search (ms)	75	0.2	0.2	0.2	0.2	0.4
Answer generation (s)	5	40.59	39.55	42.93	34.83	42.93

Table 7 — Latency by stage on the target host, second run of 19 September 2026. Ingestion stages are per document (n = 12); the query stages are per query over 75 executions (15 queries × 5 repetitions) against the 2 147-vector index; generation is per answer over the first five queries (phi3:mini, temperature 0.3, 200-token ceiling).

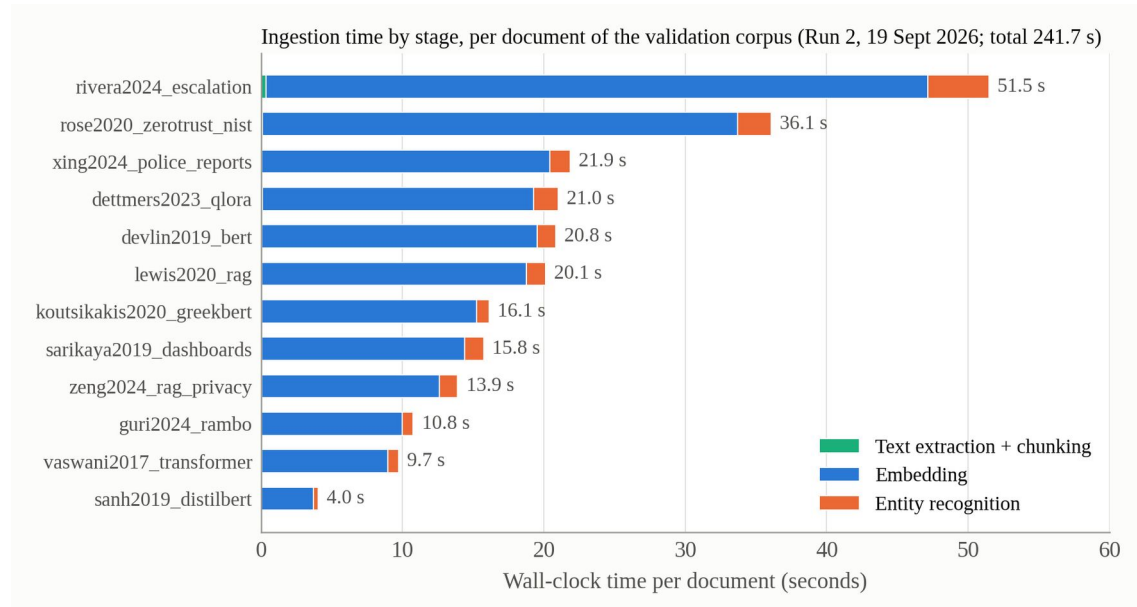


Figure 23 — Wall-clock time of the ingestion stages for each document of the validation corpus, second run of 19 September 2026 (total 241.7 s). Author’s own chart from results.json; text extraction and chunking are below the resolution of the bar for all but the longest document.

That rate answers a question which Section 5.3.1 could only estimate. At 0.854 s per page, the twenty thousand pages per year of Section 1.1 correspond to about four hours and forty-four minutes of processor time on a laptop with no graphics accelerator and no outbound connection, which is one overnight batch. At 7.59 chunks per page the same volume adds about 152 000 chunks a year, for which an IndexFlatIP over 384-dimensional single-precision vectors needs 233 MB; the order of magnitude assumed in Section 5.3.1, and the conclusion drawn from it about the index, are confirmed. At 32.0 entities per page the relational store would receive some 640 000 entity rows a year.

The query path was timed over seventy-five executions, the fifteen declared queries repeated five times each. Encoding the query with the sentence-transformer took a median of 113.4 ms (95th percentile 131.8 ms); the exhaustive FAISS search over the 2 147-vector index took a median of 0.2 ms. At this scale, therefore, the retrieval latency is almost entirely the cost of the embedding model and the index is not a bottleneck. This is consistent with the argument of Section 5.3.1 about index growth rather than against it: the linear scan becomes the limiting factor at the volumes stated there, not at the volume of the validation corpus.

Retrieval quality on the declared query set is reported in Table 8. Recall@5 and the mean reciprocal rank at five are both 1.000: for each of the fifteen queries the document declared relevant was returned at rank one. In fourteen queries all five retrieved chunks came from that document. In the remaining one, Q14 on masked language-model pre-training, four of the five did, and the fifth came from the GREEK-BERT paper, which discusses the same pre-training objective; this is the expected behaviour of a semantic index rather than an error. The cosine similarity of the top-ranked chunk ranged from 0.520 to 0.865, with a mean of 0.691.

Query	Relevant document	Rank of first hit	Chunks from relevant document (of 5)	Top cosine similarity
Q01	vaswani2017_transformer.pdf	1	5	0.610
Q02	vaswani2017_transformer.pdf	1	5	0.720
Q03	sanh2019_distilbert.pdf	1	5	0.520
Q04	lewis2020_rag.pdf	1	5	0.689
Q05	dettmers2023_qlora.pdf	1	5	0.583
Q06	dettmers2023_qlora.pdf	1	5	0.717
Q07	koutsikakis2020_greekbert.pdf	1	5	0.770
Q08	guri2024_rambo.pdf	1	5	0.865
Q09	rose2020_zerotruster_nist.pdf	1	5	0.602
Q10	zeng2024_rag_privacy.pdf	1	5	0.641
Q11	xing2024_police_reports.pdf	1	5	0.673
Q12	rivera2024_escalation.pdf	1	5	0.823
Q13	sarikaya2019_dashboards.pdf	1	5	0.753
Q14	devlin2019_bert.pdf	1	4	0.723
Q15	rose2020_zerotruster_nist.pdf	1	5	0.682

Table 8 — Known-item retrieval on the declared query set of Table D.1, top-k = 5. A retrieved chunk counts as relevant if its source document is the document declared for the query. Recall@5 = 1.000, MRR@5 = 1.000; identical in both runs.

Perfect scores require a clear statement of what they do and do not show. The task is known-item retrieval over twelve topically distinct documents, and each query was written to be answerable from exactly one of them. The result establishes that the chain from ingestion through embedding and indexing to search works end to end, and that the embedding model separates these twelve documents reliably. It establishes nothing about retrieval quality over a large and topically homogeneous corpus, which is the condition an operational deployment would face, and nothing about whether the passage returned is the best passage within the right document, because the ground truth defines relevance at the level of the document.

Normalised discounted cumulative gain is computed by the harness but is not reported, and the reason lies in the ground truth rather than in the system. With relevance defined per document, several retrieved chunks may all come from the one relevant document, so the discounted gain of a result list accumulates over chunks while the ideal list against which it is normalised is constructed over documents; the quotient can then exceed one and is not a normalised measure. The harness emits the value as written and it is disregarded. Recall@k and the reciprocal rank do not depend on that normalisation and are the measures reported. A corrected computation would construct the ideal list over the relevant chunks of the index; it was not applied, because the harness is reproduced verbatim in Listing D.2 and because, under document-level relevance, a corrected value would add nothing to Recall@5.

The entity recogniser produced the descriptive profile of Table 9: 9 052 entities of eighteen spaCy types. CARDINAL accounts for 31.4 per cent of them and PERSON for 26.6, followed by ORG at 16.8 and DATE at 7.1. The distribution says more about the corpus than about the recogniser. In academic papers the cardinal numbers are mostly citation numerals and table values, and the person names are mostly cited authors; the actors, places and times that an analyst would look for in an intelligence report form a small part of this profile. The profile is therefore one more respect, and a concrete one, in which the technical validation corpus does not stand in for the operational domain. No accuracy claim is attached to it. Entity-level precision and recall remain not measurable against this artefact, for want of an annotated ground truth in the target domain and language.

Label	Count	Share (%)	Label	Count	Share (%)
CARDINAL	2,841	31.4	LAW	86	1.0
PERSON	2,408	26.6	FAC	85	0.9
ORG	1,517	16.8	ORDINAL	66	0.7
DATE	642	7.1	LOC	52	0.6
GPE	471	5.2	MONEY	43	0.5
PERCENT	233	2.6	EVENT	39	0.4
PRODUCT	211	2.3	TIME	21	0.2
WORK_OF_ART	199	2.2	QUANTITY	21	0.2
NORP	108	1.2	LANGUAGE	9	0.1

Table 9 — Descriptive profile of the entities recognised by en_core_web_sm over the validation corpus, by spaCy label (9 052 entities, 18 labels, 32.0 per page). Counts only; no accuracy is implied.

Answer generation was timed over the first five queries of the set against phi3:mini, served by the local Ollama container at the temperature of 0.3 and the ceiling of 200 tokens used in production (Section 4.4.6). The median response took 40.6 s, within a range of 34.8 to 42.9 s. The figure is the cost of the offline constraint expressed in seconds. A 3.8-billion-parameter model performing CPU-only inference inside a virtualised guest, sharing ten cores with the embedding and recognition stages and without the unified-memory accelerator of the host, cannot be expected to do much better, and a hosted model was excluded by the constraint that defines this work. Set against the 113.6 ms of the retrieval path the ratio is roughly 350 to one, and it has a design consequence: an analyst who needs a cited passage has it in about a tenth of a second, and the forty seconds pay for a composed sentence rather than for the evidence behind it.

One property of these latencies bounds all of them. The harness was run twice on the same host, once before the entity-recognition model had been installed in the container and once after (Section D.5). The deterministic outputs of the two runs were identical: the same character count for every document, the same chunk counts, the same retrieval measures and the same similarity scores. The latencies were not. Embedding, which did identical work in both runs, was 67 per cent slower in the second; query encoding was 22 per cent slower; generation took 2.3 times as long. The difference is attributable to contention for the ten cores and 7.7 GB of the virtual machine, in which the embedding model, the recogniser and the language model ran without hardware acceleration. The figures reported here are those of the second and more heavily loaded run, which is the conservative choice; they characterise this configuration under load and should not be read as constants of the system.

Table 10 states, for each measure planned in Section 5.2, its status after the execution of the harness.

Component	Measure	Status
Named entity recognition	Precision / recall / F1 per entity type	Not measurable against this artefact — no annotated ground truth exists for the domain and language, and the recogniser is an English model (Section 6.3). A descriptive profile of 9 052 entities over 18 types is reported in Table 9
Relation extraction	Precision / recall / F1	Not applicable — no relation extractor exists in the artefact (Section 4.4.9)
Summarisation	Recall-oriented overlap against reference summaries	Not applicable — no summariser exists in the artefact (Section 4.4.9)
Natural-language search	Recall@5, MRR@5 on the declared query set	Measured — Recall@5 = 1.000, MRR@5 = 1.000 over 15 queries (Table 8). nDCG not reported: the document-level relevance definition makes its normalisation ill-defined (Section 5.3.2)
Processing throughput	Seconds per page, by pipeline stage	Measured — 0.854 s per page overall; embedding 92.2 %, entity recognition 7.4 %, extraction and chunking under 0.5 % (Table 7, Figure 23)
Query latency	Milliseconds per query, by stage	Measured — 113.4 ms query encoding, 0.2 ms exhaustive search (Table 7)
Answer generation	Seconds per answer, phi3:mini	Measured — 40.6 s median, range 34.8–42.9 s (Table 7)
Quality in use (ISO/IEC 25010)	Effectiveness, efficiency, satisfaction, freedom from risk, context coverage	Not assessed — requires participants from the analyst population; protocol stated in Section 5.5 and carried forward to Chapter 7

Table 10 — Status of each planned measure after the execution of the harness of Appendix D on the target host.

5.4. Quality evaluation against ISO/IEC 25010

The prototype is assessed here against those product-quality characteristics of ISO/IEC 25010 for which the execution of the harness supplies evidence. The quality-in-use characteristics are not scored, for the reason given in Section 5.5, so what follows is a product-quality assessment and is labelled as one.

Functional suitability is the characteristic on which the artefact scores lowest, and the score follows from a count rather than from a judgement: of the eleven functions elicited in Appendix A and placed within the scope of the Minimum Viable Product, three are implemented. Functional completeness is accordingly low. Functional correctness, assessed over what was built, is supported by the measurements: every document of the validation corpus passed through every stage without error, and every declared query was answered with the correct document at rank one.

Performance efficiency is satisfied for the interactive path and characterised for the batch path. The query path returns in a median of 113.6 ms end to end, which meets the expectation recorded in Table 4 that inference complete within an operationally acceptable time. Ingestion runs at 0.854 s per page, of which embedding is 92.2 per cent; this places the annual volume of Section 1.1 within a single overnight window and identifies the embedding model, rather than the index, the extractor or the recogniser, as the component to optimise should that window ever bind. Generation, at a median of 40.6 s, is the one figure that would be unacceptable in interactive use if the extractive answer were not available in a tenth of a second.

Reliability is supported by three observations. Twelve of twelve documents completed every stage without error in both executions of the harness; the deterministic outputs were identical across the two executions; and the generator falls back to returning the highest-ranked passage verbatim, with its attribution, whenever the language model cannot be

reached or generation raises, so that the system degrades from abstractive to extractive answering instead of failing.

Security cannot be scored above the lowest level of the scale. The artefact implements no authentication, no authorisation and no encryption at rest (Section 4.4.8). This was a scope decision rather than an oversight, and its consequence is stated in Section 6.4 without mitigation: the system as built should not be deployed against a classified corpus.

Maintainability is supported by the structure rather than by a measurement. The dependency rule of Figure 8 is respected in the code, each component encapsulates one model or one store, and the harness is itself evidence of the property: it imports the production classes unmodified and exercises them outside the application, which is possible only because the retrieval and extraction layers carry no dependency on the web layer.

Portability is supported by the deployment. The system runs as three containers built for linux/arm64 with no host-specific configuration beyond the mounted data directories, and the measurements of Section 5.3 were obtained from that deployment rather than from a development arrangement that differs from it. The one qualification is that the containerisation which makes the system portable also denies it the hardware acceleration of the host; the generation latency is the price of that trade.

Quality in use — effectiveness, efficiency, satisfaction, freedom from risk and context coverage — is not assessed. The instrument was specified in Section 5.2 following Barletta et al. (2023) and Trichkova-Kashamova (2021), and it requires participants. Section 5.5 states the position taken.

5.5. Usability assessment

No usability assessment was conducted within the scope of this dissertation. The protocol that would be applied is stated here rather than omitted, so that the assessment can be carried out by a subsequent study and so that the omission is a bounded decision rather than a gap.

The instrument is the quality-in-use model of ISO/IEC 25010, instantiated as concrete measures following Barletta et al. (2023) and weighted by expert judgement following Trichkova-Kashamova (2021). Each participant would be given a set of tasks defined over the validation corpus — locating the document that answers a stated question, judging whether the passage returned supports the generated answer, and identifying the source of a claim — and the protocol would record, for effectiveness, the proportion of tasks completed with a correct and correctly attributed answer; for efficiency, the time to completion per task; for satisfaction, a standardised post-task questionnaire; for freedom from risk, the frequency with which a participant accepted a generated statement that the retrieved passage did not support; and for context coverage, the proportion of the elicited functions of Appendix A that the participant could exercise at all. The participant profile is the intelligence-analyst population for which the system is designed, and the number required by the weighting procedure is small, of the order of five to ten.

The assessment was not conducted because it requires access to that population, and that access is governed by the same organisational constraints that make the operational corpus unavailable to an academic study (Section 5.2.1). Conducting it instead with participants from outside the analyst population would measure the usability of the interface for a group whose task it is not, and would yield a number that could not be read as evidence about the intended context of use. A measure obtained from the wrong population is not a weaker version of the right measure but a different measure reported under the same name.

The consequence for the conclusions of this dissertation is stated plainly. No claim is made about the usability of the artefact, about its effectiveness in the hands of an analyst, or about the value an analyst would attach to its output. The quality-in-use characteristics of ISO/IEC 25010 remain unscored, and the product-quality assessment of Section 5.4 is not a substitute for them. The assessment is carried forward as the first item of future work in Chapter 7.

5.6. Summary

This chapter reported the execution of the evaluation harness of Appendix D on the target host. The ingestion path processed all twelve documents of the 283-page validation corpus without error, at 0.854 s per page, producing 2 147 chunks and 9 052 entities. The retrieval path returned the correct document at rank one for every one of the fifteen declared queries, giving Recall@5 and a mean reciprocal rank of 1.000, in a median of 113.6 ms of which the exhaustive index accounts for 0.2 ms. Answer generation with phi3:mini took a median of 40.6 s. Two measures were established as not applicable to this artefact and one as not measurable against it, and the quality-in-use assessment was not conducted for want of access to the analyst population. The interpretation of what was and was not built, and of what these figures do and do not license, is the subject of Chapter 6.

6. Discussion

6.1. Introduction

This chapter interprets the results reported in Chapter 5. Section 6.2 states the key findings against the research objectives; Section 6.3 states the limitations of the study and the threats to the validity of its conclusions; Section 6.4 considers the implications for practice; Section 6.5 summarises.

6.2. Key findings

The research question of this dissertation is a design question: whether an information system that analyses and visualises intelligence reports using artificial intelligence can be built and made to work under a complete network-isolation constraint. The artefact answers part of that question and declines to answer the rest, and the shape of the division is itself the principal finding.

The first finding is that the offline constraint is not the binding one. Every component of the retrieval pillar runs locally on commodity hardware without a GPU: a 384-dimensional sentence-transformer for embedding, an exact FAISS index for retrieval, and a 3.8-billion-parameter language model served by a local Ollama container for generation. No component makes an outbound call at run time. The feasibility that Sanh et al. (2019) and Dettmers et al. (2023) establish for individual models is confirmed here at the level of an integrated system: the isolation requirement that renders every commercial platform surveyed in Section 2.2 unusable did not, in the event, prevent the system from being built. The measurements of Chapter 5 give that finding a quantitative form: the whole validation corpus was ingested at 0.854 s per page on ten virtualised processor cores, and every declared query was answered with the correct document at rank one in about a tenth of a second.

The second finding is that the binding constraint was engineering effort, and that it fell on a predictable part of the design. The retrieval pillar — ingestion, chunking, embedding, indexing, grounded generation, provenance — was implemented completely. The structured-extraction pillar — relations, events, timelines — and the visualisation pillar were not implemented at all, and the security pillar was not implemented at all. This is not an accident of scheduling. Retrieval-augmented generation is a solved integration problem with mature libraries; document-level relation extraction, temporal anchoring and the construction of an analyst dashboard are each a research problem in their own right, as Sections 2.5, 2.6 and 2.7 establish. The Minimum Viable Product converged on the part of the design that could be assembled from existing components.

The third finding is a positive one that the design did not anticipate. The implemented provenance mechanism exceeds what was specified. Every query and every retrieved chunk, with its similarity score and its rank, is written to the relational store before an answer is returned, which means the evidence behind any answer the system has ever given can be reconstructed after the fact. Combined with the generator's explicit fallback to returning a retrieved passage verbatim when the language model is unavailable, the artefact is more conservative than its design: it prefers to show the analyst a source than to compose a claim. In a domain where Rivera et al. (2024) demonstrate that language models produce escalatory actions with worrying justifications and can misleadingly appear nuanced, that conservatism is the correct default, and it emerged from an

engineering decision rather than from the requirements. The measured contrast of Section 5.3.2, 114 ms for a cited passage against 41 s for a composed answer, shows what that preference costs and what it buys.

The fourth finding is negative and must be stated with equal clarity. The system as built is not a system for the analysis and visualisation of intelligence reports. It is a system for the semantic retrieval of passages from a corpus of documents, with grounded answers and full provenance. Entities are extracted but never shown; relations and events are not extracted; nothing is visualised. Against the objective stated in Section 1.2, the artefact demonstrates feasibility for the retrieval component and leaves the analysis and visualisation components undemonstrated.

6.3. Limitations

Certain limitations are already determined by the design of the study and can be stated in advance of the results.

- The study evaluates a single artefact built by a single developer. It can support a conclusion about technical feasibility and about the measured quality characteristics of that artefact; it cannot support statistical generalisation, and no claim of superiority over any existing system is made, since no comparison system was evaluated.
- The evaluation corpus consists of representative, unclassified documents. Performance on the classified operational corpus, which cannot be used in an academic study, may differ.
- The abstraction gap identified by Olsen et al. (2024) — between the atomic, span-level annotations on which NLP models are trained and the aggregated, real-world event records that intelligence analysis actually requires — constrains what any system built from off-the-shelf models can achieve, and this system is no exception.
- The system deliberately excludes recommendation, scoring and prediction. This is a design decision grounded in the empirical evidence of escalation behaviour in language-model agents in military and diplomatic settings (Rivera et al., 2024) and in the risk analysis of Probasco et al. (2025). It is a limitation of capability that is accepted in exchange for a reduction in risk, and a reader who does not accept that trade-off will regard it as a weakness.
- The layered-monolith architecture was chosen over a microservices decomposition for the reasons given in Section 4.3. That decision is sound for a single-machine installation but would not survive a requirement for multi-site deployment, for which the scalability and independent-deployment benefits of microservices documented by S. Li et al. (2021) become decisive.

The construction of the artefact adds six further limitations, each of which is a property of what was built rather than of how the study was designed.

- The artefact does not enforce its own isolation. The embedding model is not obtained by either setup script and is therefore fetched from a remote registry the first time it is needed, and the front-end container is started without its usage telemetry disabled. No content leaves the host in either case — no report, chunk, query or embedding is transmitted — but the offline property, which is the defining property of the design, is permitted by the artefact rather than guaranteed by it. On a host with no route out, the embedding model cannot load and the pipeline degrades silently to zero vectors, which is a worse failure than an explicit one. The remedy is small and is stated in Chapter 7.

- The artefact implements three of the eleven use cases placed within the scope of the Minimum Viable Product, and one further use case in part. The extraction of relations and events, summarisation, the event timeline, aggregate visualisation, export and the relationship graph were not built. No claim of feasibility is made for any of them, and none should be read into this dissertation.
- The security layer was not implemented. The artefact has no authentication, no authorisation and no audit logging, and its cross-origin policy is permissive. The zero-trust design of Section 4.3, following Rose et al. (2020), remains a design. What the artefact does provide is the property that motivated the design — the corpus and the retrieval index never leave the host — and it therefore prevents the corpus and the index from leaving the host, but it does not by itself defeat the query-based extraction of the retrieval database that Zeng et al. (2024) demonstrate — which the unimplemented access-control layer was meant to address — and it offers no protection against any actor with access to the machine. This is acceptable in a prototype and unacceptable in a deployment.
- The language models are English. The entity recogniser is spaCy's `en_core_web_sm` and the embedding model is `all-MiniLM-L6-v2`; neither is a Greek model. Section 2.3 identified GREEK-BERT (Koutsikakis et al., 2020) and Meltemi (Voukoutis et al., 2024) as the correct choices for the target corpus, and the artefact does not use them. The retrieval results obtainable from the current artefact therefore say nothing about the performance the system would achieve on the Greek-language corpus it was designed for.
- The vector index does not scale to the stated volume. The measurement of Section 5.3 gives 7.59 chunks per page, which places the corpus at approximately 152 000 chunks after one year at the volume stated in Section 1.1. FAISS IndexFlatIP performs an exhaustive scan, so both search cost and memory grow linearly with the corpus. The structure is exact and appropriate for a prototype and will require replacement by an approximate index before the system meets the volume that motivated it.
- The confidence field of the entity table is written as a constant. spaCy exposes no confidence score for its entity recogniser, so the field carries no information. Any downstream ranking or filtering of entities by confidence would be ranking on a constant, and the field should either be populated by a model that produces one or removed.

Four further limitations attach to the measurements of Chapter 5 and are stated so that its figures are not read beyond their warrant.

- The corpus. The validation corpus is a technical one, chosen for the structural property of length and sectioning rather than for its subject matter, and it stands in for the operational corpus in no respect other than that property. The retrieval figures describe performance over twelve topically distinct open-access papers in English; the operational setting is a large, topically homogeneous, Greek-language corpus, and nothing measured here transfers to it. The entity profile of Table 9 makes the same point concretely: its dominant types are the bibliographic apparatus of academic writing, not the actors and settings of an intelligence report.
- The ground truth. Relevance is declared at the level of the document, which makes the reported measures known-item measures: they establish that the system surfaces the right document, not that it surfaces the best passage within it. This is also why no normalised discounted cumulative gain is reported (Section 5.3.2).
- The environment and its stability. The measurements were obtained inside a virtualised Linux guest on the Apple Silicon host, which is the deployment the system uses but which

imposes a cost that native execution would not. Two executions of the identical workload on the same machine produced identical deterministic outputs but latencies differing by as much as a factor of 2.3, according to what else was competing for the ten cores of the virtual machine. The latency figures are therefore characteristic values for this configuration under load, not constants, and upper bounds rather than the best the hardware could achieve.

- The absence of a usability assessment, stated in Section 5.5, which leaves the quality-in-use half of the ISO/IEC 25010 evaluation unscored and any claim about the artefact in the hands of an analyst unmade.

6.4. Implications

Three implications follow for practice.

- For an organisation in this position, the isolation requirement is not the reason not to proceed. A retrieval system that keeps a classified corpus entirely on an isolated host, answers questions in natural language, and shows the analyst the passage behind every answer, can be assembled from open components on hardware the organisation already owns. That is the practical content of the feasibility result. The ingestion of a year’s volume in a single overnight batch, on such hardware, has now been measured rather than estimated.
- The provenance mechanism should be treated as a requirement rather than a by-product. The artefact records the chunks, scores and ranks behind every answer it has given. In a domain where the responsible-AI literature insists on explainability and human control (Meerveld et al., 2023) and where language-model behaviour has been shown to be unpredictable in military and diplomatic decision-making simulations (Rivera et al., 2024), an answer that cannot be traced to a source should not be shown to an analyst at all.
- The security layer must not be deferred a second time. It was deferred in this prototype for defensible reasons, and the consequence is a system that would be negligent to deploy: any actor with access to the host has unrestricted access to the corpus, and the record of who asked what — which the database already keeps — is not attributable to any user, because there are no users. Access control is the precondition for the audit trail to mean anything.

6.5. Summary

This chapter interpreted the artefact against the objectives of Section 1.2. The offline constraint proved not to be the obstacle; the engineering effort required by structured extraction, visualisation and access control did. The implemented provenance mechanism exceeds the design and is the artefact’s strongest property. Against the stated objective, feasibility is demonstrated for the retrieval component and left undemonstrated for the analysis and visualisation components. The measurements of Chapter 5 give the feasibility claim for the retrieval pillar a quantitative basis and, at the same time, bind it to the configuration on which they were obtained.

7. Conclusions and Future Work

This dissertation set out to design an information system that would apply open-source natural language processing to the analysis and visualisation of intelligence reports, under the constraint that no content may leave the organisation's isolated network, and to demonstrate the technical feasibility of that design by building and evaluating a Minimum Viable Product.

A design was produced. It comprises a layered architecture, a data model that separates documents, entities, relations and events, a retrieval mechanism grounded in the corpus and carrying its provenance, an analyst-facing interface specified against the dashboard design dimensions of Sarikaya et al. (2019), and a security layer built on the zero-trust principles of Rose et al. (2020) and an explicit air-gap threat model. That design is the first contribution and it stands independently of what was subsequently built.

A prototype was built, and it demonstrates the feasibility of one of the three pillars of that design. Retrieval-augmented generation over a corpus of long documents, with locally hosted models, no outbound connectivity, and complete retrieval provenance, is achievable on commodity hardware without a GPU. The prototype ingests, chunks, embeds, indexes, retrieves and answers, and it records the evidence behind every answer it gives. The isolation constraint — the property that makes every surveyed commercial platform unusable in this setting — proved not to be the obstacle. Measured on the target host, it ingested the 283-page validation corpus at 0.854 s per page and returned the document declared relevant at rank one for every one of the fifteen queries put to it.

The structured-extraction and visualisation pillars were not built, and the security pillar was not built. The dissertation therefore does not demonstrate the feasibility of extracting relations and events from intelligence reports, of constructing an event timeline from them, or of operating such a system under an enforced access-control regime. Those remain designs, and the honest statement of what a prototype does not establish is as much a part of the result as the statement of what it does.

The scope of the conclusion is bounded in a further respect. The prototype uses English models on a technical validation corpus. Nothing in it licenses a claim about performance on the Greek-language operational corpus that motivated the work, and no such claim is made.

Future work. Nine directions follow from the design and from what the artefact did and did not establish.

- Conduct the usability assessment. The protocol of Section 5.5 is specified and awaits participants from the analyst population; until it is carried out, the quality-in-use half of the ISO/IEC 25010 evaluation remains unscored and no claim about the artefact in an analyst's hands can be made. It is placed first because it is the only item on this list that requires organisational access rather than engineering effort.
- Enforce the offline property before anything else is added. Warm the embedding-model cache into the `./models` volume during provisioning, so that no remote registry is consulted at run time; disable the front-end usage telemetry; and make the failure to load a model an explicit error rather than a silent fall-back to zero vectors. This is a day's work and it is the precondition for every other claim the system makes about itself.
- Complete the extraction pillar. Implement document-level relation extraction and event extraction with temporal anchoring, following the entity-pair-guided formulation of Zhang et al. (2025) and the temporal reasoning surveyed by Leeuwenberg and Moens (2020), and

add the corresponding tables to the schema. This is the precondition for the event timeline and the relationship graph, and therefore for the visualisation pillar.

- Replace the English models with Greek ones. Substitute GREEK-BERT (Koutsikakis et al., 2020) for spaCy's English entity recogniser and evaluate Meltemi (Voukoutis et al., 2024) against phi3:mini for generation. The modular structure of the artefact makes both substitutions local: each model is encapsulated in a single class behind a stable interface, which is the maintainability property that the layered design was chosen to obtain.
- Implement the security layer as designed. Authentication and authorisation enforced before every resource access, following Rose et al. (2020); role-based authorisation after Sandhu et al. (1996); and an audit record for every action. The database schema already contains the provenance half of the accountability requirement; the access-control half is absent.
- Assemble a domain proxy corpus and run the evaluation. The open socio-political event datasets surveyed by Olsen et al. (2024) are the legitimate substitute for the classified corpus. The evaluation harness of Appendix D executes the task-level and latency measurements against any corpus and query set supplied to it.
- Replace the exhaustive vector index. FAISS IndexFlatIP is exact and linear; at the volume established in Section 5.3 it will require replacement by an approximate index before the system reaches operational scale.
- Automated alerting on anomalous patterns, deferred from the Minimum Viable Product as use case UC-12.
- Predictive analytics over the extracted event stream. The formulation of societal event forecasting as a machine-learning prediction task is surveyed by Deng and Ning (2021), and the extracted entity-event records the system would produce are the natural input to such a model. The warnings of Rivera et al. (2024) and Probasco et al. (2025) apply with full force to any such extension, and it should not be undertaken without an explicit human-in-the-loop design.

References

- Adhikary, S., Sen, P., Roy, D., & Ghosh, K. (2024). A case study for automated attribute extraction from legal documents using large language models. *Artificial Intelligence and Law*. Advance online publication. <https://doi.org/10.1007/s10506-024-09425-7>
- Alneyadi, S., Sithirasanen, E., & Muthukkumarasamy, V. (2016). A survey on data leakage prevention systems. *Journal of Network and Computer Applications*, 62, 137–152. <https://doi.org/10.1016/j.jnca.2016.01.008>
- Alonso, S., Kalinowski, M., Ferreira, B., Barbosa, S. D. J., & Lopes, H. (2023). A systematic mapping study and practitioner insights on the use of software engineering practices to develop MVPs. *Information and Software Technology*, 156, Article 107144. <https://doi.org/10.1016/j.infsof.2022.107144>
- Barletta, V. S., Caivano, D., Colizzi, L., Dimauro, G., & Piattini, M. (2023). Clinical-chatbot AHP evaluation based on “quality in use” of ISO/IEC 25010. *International Journal of Medical Informatics*, 170, Article 104951. <https://doi.org/10.1016/j.ijmedinf.2022.104951>

- Beck, F., Burch, M., Diehl, S., & Weiskopf, D. (2017). A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1), 133–159. <https://doi.org/10.1111/cgf.12791>
- Browne, T. O., Abedin, M., & Chowdhury, M. J. M. (2024). A systematic review on research utilising artificial intelligence for open source intelligence (OSINT) applications. *International Journal of Information Security*, 23(4), 2911–2938. <https://doi.org/10.1007/s10207-024-00868-2>
- Carrara, B., & Adams, C. (2016). Out-of-band covert channels—A survey. *ACM Computing Surveys*, 49(2), Article 23. <https://doi.org/10.1145/2938370>
- Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., Persson, K. A., & Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, 15, Article 1418. <https://doi.org/10.1038/s41467-024-45563-x>
- Das, B. C., Amini, M. H., & Wu, Y. (2025). Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6), Article 152. <https://doi.org/10.1145/3712001>
- de Carvalho, V. D. H., & Costa, A. P. C. S. (2023). Exploring text mining and analytics for applications in public security: An in-depth dive into a systematic literature review. *Socioeconomic Analytics*, 1(1), 5–55. <https://doi.org/10.51359/2965-4661.2023.259008>
- Deng, S., & Ning, Y. (2021). A survey on societal event forecasting with deep learning (arXiv:2112.06345). arXiv. <https://doi.org/10.48550/arXiv.2112.06345>
- Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. Curran Associates. https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- Görg, C., Liu, Z., & Stasko, J. (2014). Reflections on the evolution of the Jigsaw visual analytics system. *Information Visualization*, 13(4), 336–345. <https://doi.org/10.1177/1473871613495674>
- Guri, M. (2024). RAMBO: Leaking secrets from air-gap computers by spelling covert radio signals from computer RAM. In L. Fritsch, I. Hassan, & E. Paintsil (Eds.), *Secure IT systems: 28th Nordic Conference, NordSec 2023 (Lecture Notes in Computer Science, Vol. 14324, pp. 144–161)*. Springer. https://doi.org/10.1007/978-3-031-47748-5_9
- Guri, M., & Elovici, Y. (2018). Bridgeware: The air-gap malware. *Communications of the ACM*, 61(4), 74–82. <https://doi.org/10.1145/3177230>

- Hussain, M. A., Samrouth, K., & Bakir, N. (2025). A survey on malware attacks in industrial air-gap systems. *International Journal of Information Security*, 24(3), Article 146. <https://doi.org/10.1007/s10207-025-01044-w>
- Ignaczak, L., Goldschmidt, G., da Costa, C. A., & da Rosa Righi, R. (2021). Text mining in cybersecurity: A systematic literature review. *ACM Computing Surveys*, 54(7), Article 140. <https://doi.org/10.1145/3462477>
- Koutsikakis, J., Chalkidis, I., Malakasiotis, P., & Androutsopoulos, I. (2020). GREEK-BERT: The Greeks visiting Sesame Street. In *11th Hellenic Conference on Artificial Intelligence (SETN 2020)* (pp. 110–117). Association for Computing Machinery. <https://doi.org/10.1145/3411408.3411440>
- Kushk, A., & Kochut, K. (2021). ESDL: Entity summarization with deep learning. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs (IJCKG '21)* (pp. 186–190). Association for Computing Machinery. <https://doi.org/10.1145/3502223.3502743>
- Leeuwenberg, A., & Moens, M.-F. (2020). A survey on temporal reasoning for temporal information extraction from text (Extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)* (pp. 5085–5089). International Joint Conferences on Artificial Intelligence. <https://doi.org/10.24963/ijcai.2020/712>
- Lenarduzzi, V., & Taibi, D. (2016). MVP explained: A systematic mapping study on the definitions of minimal viable product. In *2016 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 112–119). IEEE. <https://doi.org/10.1109/SEAA.2016.56>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)* (pp. 9459–9474). Curran Associates. <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- Li, Q., Li, J., Sheng, J., Cui, S., Wu, J., Hei, Y., Peng, H., Guo, S., Wang, L., Beheshti, A., & Yu, P. S. (2024). A survey on deep learning event extraction: Approaches and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5), 6301–6321. <https://doi.org/10.1109/TNNLS.2022.3213168>
- Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., & Babar, M. A. (2021). Understanding and addressing quality attributes of microservices architecture: A systematic literature review. *Information and Software Technology*, 131, Article 106449. <https://doi.org/10.1016/j.infsof.2020.106449>
- Li, T., Zhang, X., Wang, Y., Zhou, Q., Wang, Y., & Dong, F. (2024). Machine learning for requirements engineering (ML4RE): A systematic literature review complemented by practitioners’ voices from Stack Overflow. *Information and Software Technology*, 172, Article 107477. <https://doi.org/10.1016/j.infsof.2024.107477>

- Meerveld, H. W., Lindelauf, R. H. A., Postma, E. O., & Postma, M. (2023). The irresponsibility of not using AI in the military. *Ethics and Information Technology*, 25(1), Article 14. <https://doi.org/10.1007/s10676-023-09683-0>
- Mo, Y., & Li, Z. (2024). Incorporating entity type-aware and word–word relation-aware attention in generative named entity recognition. *Electronics*, 13(7), Article 1407. <https://doi.org/10.3390/electronics13071407>
- Mo, Y., Tang, H., Liu, J., Wang, Q., Xu, Z., Wang, J., Wu, W., & Li, Z. (2023). Multi-task transformer with relation-attention and type-attention for named entity recognition. In *ICASSP 2023 – 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1–5). IEEE. <https://doi.org/10.1109/ICASSP49357.2023.10094905>
- Necula, S.-C., Dumitriu, F., & Greavu-Şerban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), Article 2055. <https://doi.org/10.3390/electronics13112055>
- Olsen, H. B., Simon, É., Velldal, E., & Øvrelid, L. (2024). Socio-political events of conflict and unrest: A survey of available datasets. In *Proceedings of the 7th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2024)* (pp. 40–53). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.case-1.5>
- Probasco, E. S., Toner, H., Burtell, M., & Rudner, T. G. J. (2025). *AI for military decision-making: Harnessing the advantages and avoiding the risks* [Issue brief]. Center for Security and Emerging Technology. <https://doi.org/10.51593/20240028>
- Rivera, J.-P., Mukobi, G., Reuel, A., Lamparth, M., Smith, C., & Schneider, J. (2024). Escalation risks from language models in military and diplomatic decision-making. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)* (pp. 836–898). Association for Computing Machinery. <https://doi.org/10.1145/3630106.3658942>
- Rojas, H., Renteria, R., Martinez Duran, V., Tomaylla Gutiérrez, Y., Ibarra Cabrera, M. J., & Aminuddin, A. (2025). Mapping the evolution and future directions of ISO/IEC 25010: A bibliometric and thematic analysis. *Engineering, Technology & Applied Science Research*, 15(5), 27530–27541. <https://doi.org/10.48084/etasr.11772>
- Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture* (NIST Special Publication 800-207). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-207>
- Saeed, A., Bangash, Y. A., Farooq, M., & Rehman, H. (2025). Securing air-gapped systems—Review of covert techniques for data ex-filtration and a new clause proposal for ISO 27001. *International Journal of Information Security*, 24(4), Article 182. <https://doi.org/10.1007/s10207-025-01103-2>
- Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2), 38–47. <https://doi.org/10.1109/2.485845>
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC²), co-located with NeurIPS 2019*. <https://doi.org/10.48550/arXiv.1910.01108>

- Sarikaya, A., Correll, M., Bartram, L., Tory, M., & Fisher, D. (2019). What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, 25(1), 682–692. <https://doi.org/10.1109/TVCG.2018.2864903>
- Sharma, G., Sharma, D., & Sasikumar, M. (2024). Summarizing long scientific documents through hierarchical structure extraction. *Natural Language Processing Journal*, 8, Article 100080. <https://doi.org/10.1016/j.nlp.2024.100080>
- Shen, L., Shen, E., Luo, Y., Yang, X., Hu, X., Zhang, X., Tai, Z., & Wang, J. (2023). Towards natural language interfaces for data visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 29(6), 3121–3144. <https://doi.org/10.1109/TVCG.2022.3148007>
- Stasko, J., Görg, C., & Liu, Z. (2008). Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2), 118–132. <https://doi.org/10.1057/palgrave.ivs.9500180>
- Trichkova-Kashamova, E. (2021). Applying the ISO/IEC 25010 quality models to an assessment approach for information systems. In *2021 12th National Conference with International Participation (ELECTRONICA)* (pp. 1–4). IEEE. <https://doi.org/10.1109/ELECTRONICA52725.2021.9513662>
- van Rossum, G., & the Python development team. (2018). *Python tutorial* (Release 3.7.0). Python Software Foundation. <https://docs.python.org/release/3.7.0/tutorial/>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (pp. 5998–6008). Curran Associates. https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
- Voukoutis, L., Roussis, D., Paraskevopoulos, G., Sofianopoulos, S., Prokopidis, P., Papavasileiou, V., Katsamanis, A., Piperidis, S., & Katsouros, V. (2024). *Meltemi: The first open large language model for Greek* (arXiv:2407.20743). arXiv. <https://doi.org/10.48550/arXiv.2407.20743>
- Xing, X., & Chen, P. (2024). Entity extraction of key elements in 110 police reports based on large language models. *Applied Sciences*, 14(17), Article 7819. <https://doi.org/10.3390/app14177819>
- Zeng, S., Zhang, J., He, P., Xing, Y., Liu, Y., Xu, H., Ren, J., Wang, S., Yin, D., Chang, Y., & Tang, J. (2024). The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG). In *Findings of the Association for Computational Linguistics: ACL 2024* (pp. 4505–4524). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.findings-acl.267>
- Zhang, F., Yu, H., Cheng, J., & Xu, H. (2025). Entity pair-guided relation summarization and retrieval in LLMs for document-level relation extraction. In *Findings of the Association for Computational Linguistics: NAACL 2025* (pp. 4022–4037). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2025.findings-naacl.224>

Zhao, C., Zhou, X., Xie, X., & Zhang, Y. (2024). Hierarchical attention graph for scientific document summarization in global and local level. In *Findings of the Association for Computational Linguistics: NAACL 2024* (pp. 714–726). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.findings-naacl.45>

Appendix A: Requirements elicitation record

This appendix reproduces the problem-statement and requirements-elicitation record produced by the author in the operational setting described in Chapter 1. It is referenced from Section 1.1 and from Section 4.2, and it is presented as primary material gathered during the study, not as a published source. The record was produced in Greek during Sub-steps 2.1 to 2.5 of the methodology and is given here in the author's English translation. The organisation of the material into sub-sections and tables is editorial; no requirement has been added, removed or altered in substance, and the identifiers (FR-n, NFR-n) have been assigned for ease of reference only.

A.1 Problem statement

The directorate receives, on a daily basis, extensive reports — daily intelligence bulletins and intelligence briefs — in a variety of file formats. These reports consist mainly of texts of more than twenty pages that cover a wide range of subject areas: political, military, social and technological. On an annual basis their cumulative volume exceeds twenty thousand pages. At present the documents are stored solely by chronological criteria, without conceptual tagging and without any facility for intelligent search or thematic analysis.

The key elements of the problem, as recorded during elicitation, are the following:

- inability to retrieve information quickly and accurately;
- absence of structured analysis and of trend detection;
- dependence on human analysts for routine tasks;
- inability to exploit artificial intelligence in support of decision-making;
- a high risk of error — chiefly of omission, owing to the volume of information — attributable to the human factor.

Four groups were recorded as affected by the problem: intelligence analysts (the subject-matter experts), who find it difficult to extract knowledge for lack of tools; management and those responsible for operational decision-making, who lack timely and synthesised information; the information-technology and support function, which lacks the tools to meet contemporary processing requirements, none of which exist at the intelligence node; and users with limited technical training, who cannot exploit the information directly.

The proposed solution recorded at this stage was the design of a standalone information system that would (i) accept and store files of various formats (PDF, Word, TXT, Excel); (ii) facilitate the search for information; (iii) automatically extract structured information — names, locations, dates, events, coordinates — by means of natural language processing; (iv) analyse and visualise the resulting knowledge, for example as an event timeline, trends and alerts; and (v) support the generation of reports and warnings and the dynamic exploration of the corpus in natural language. The expected benefits were a reduced dependence on manual procedures, faster data-driven decision-making, better organisation of information with improved retrieval, and the dynamic, automated analysis of trends and events.

A.2 Stakeholders and roles

The stakeholder analysis was elicited through a short series of questions. Documents are entered into the system by the intelligence staff officer or intelligence analyst on duty, who must be authorised to do so. The results of the analysis are used first by the analysts themselves, in support of their own work, and by extension by the director of the sector and

the superior officers, who carry the obligation of timely briefing and early warning for the purpose of decision-making. The strategic decision-maker is the commanding general — and, by extension, the Chief of the Hellenic National Defence General Staff; between the analyst and the general stand the Director of the 2nd Staff Office (intelligence), the Chief of Staff and the Deputy Commander, and the Director is the last officer with direct access to the system. Technical staff will support and administer the system as required. Table A.1 consolidates the stakeholders identified.

Table A.1 — Stakeholders of the proposed system, as identified during elicitation

Stakeholder	Relationship to the system	Category
Intelligence analyst / intelligence staff officer	Enters documents into the system; searches the corpus; produces analyses and reports. Principal user.	Primary
Sector director / Director of the 2nd Staff Office	Uses the analytical output for briefing and assessment; has direct access to the system.	Primary
Deputy Commander / Chief of Staff	Involved in the high-level evaluation of intelligence and in decision support.	Primary
Commanding general / Chief of the Hellenic National Defence General Staff	Relies on the reports and conclusions of the system for critical decisions; indirect user (through briefings) but principal recipient of the information.	Primary
Technical support team / IT staff	Responsible for installation, configuration, maintenance and security of the system.	Secondary
Analysts without full technical training	Must be able to use the system with ease; affected chiefly by the quality of the user experience.	Secondary

A.3 Functional requirements

The functions initially requested were, in the words of the elicitation record: storage of files of every kind (Word, PowerPoint, Excel, text, databases, audio, video); analysis of texts for names and entities, relations between entities, dates, geographical places and geographical coordinates, with multi-criteria classification of all incoming files by entity, topic, date and place; the generation of an event timeline and of a histogram of relations between entities; alerts for “unusual” events, with a capacity for partial, guided prediction; automatic extraction of summaries; and search by keyword or in natural language, with natural language as the primary mode. Table A.2 states the functional requirements derived from that list, grouped by function.

Table A.2 — Functional requirements recorded during elicitation

ID	Group	Requirement
FR-1	File management	The system shall allow the storage of files in the formats Word, PDF, Excel, TXT and PowerPoint, as well as databases, audio files and video.
FR-2	File management	The system shall store metadata for every file (e.g. date, author, source).
FR-3	File management	The user shall be able to upload files in bulk.
FR-4	Text and entity analysis (NLP / NER)	The system shall recognise and extract entities from the texts: persons, locations, organisations, dates, events, numerical data and geographical coordinates.
FR-5	Text and entity analysis (NLP / NER)	The system shall identify relations between the extracted entities (e.g. who did what, where and when).
FR-6	Text and entity analysis (NLP / NER)	The system shall automatically classify and organise the files by topic, date, entities and geographical criteria.
FR-7	Visualisations and reports	The system shall generate dynamic timelines of events from the data.

ID	Group	Requirement
FR-8	Visualisations and reports	The system shall automatically generate histograms or relationship diagrams between the identified entities.
FR-9	Visualisations and reports	The user shall be able to define filters for the above (e.g. by period, region or entity type).
FR-10	Alerts and predictive elements	The system shall automatically notify the user when an “unusual” or critical event is detected on the basis of predefined rules.
FR-11	Alerts and predictive elements	The system shall support guided prediction on the basis of past patterns and user input.
FR-12	Automatic summarisation	The system shall automatically produce summaries of the ingested documents, either as bullet points or as a short narrative text.
FR-13	Information search	The user shall be able to search in natural language (e.g. “Show me all events related to region X or person Y in September 2023”).
FR-14	Information search	The system shall also support keyword search and combinations of filters (e.g. date, event type, name).

A.4 Non-functional requirements

Non-functional requirements describe not what the system does but how it does it — the quality characteristics it must exhibit. They were recorded against the characteristics of ISO/IEC 25010, with two additional headings that the operational setting made necessary: the standalone operation of the artificial-intelligence components and the hardware on which they run. Table A.3 states them.

Table A.3 — Non-functional requirements recorded during elicitation, organised by ISO/IEC 25010 characteristic

Characteristic	Requirement
Security	Access shall be granted only to authorised users through strong authentication (e.g. two-factor authentication). All access and data-modification events shall be recorded in audit logs. User roles with different levels of access shall be supported (e.g. analyst, administrator, director). All data shall be stored and transmitted in encrypted form (end-to-end encryption).
Performance efficiency	The system shall analyse at least one hundred pages of text in less than five minutes. The response to a natural-language query shall not exceed ten seconds.
Maintainability	The system shall have a modular architecture, so that individual functions (e.g. the NLP engine) can easily be replaced or upgraded. The source code shall be documented for maintenance by internal technical staff.
Usability	The user shall be able to perform the basic functions without technical support, through a simple graphical interface. The system shall provide built-in help or guidance (e.g. tips, onboarding).
Functional suitability	The system shall handle documents in both Greek and English. Entity recognition and extraction shall achieve an accuracy of at least 85 per cent.
Compatibility	The system shall operate on an internal network without an internet connection. It shall be compatible with existing Windows and Linux environments.
Reliability	The system shall recover fully after an error or interruption of service (automatic recovery). Backups shall be kept at twenty-four-hour intervals.
AI standalone capability	All artificial-intelligence, machine-learning and natural-language-processing models shall run locally, without connection to external APIs or cloud services. The system shall possess the computing infrastructure (e.g. GPU, processors, storage) needed to execute the models offline. Model updating or training shall take place locally, under approved procedures and without transfer of data to third parties. The architecture shall ensure that no data or file leaks or is transmitted outside the internal network.

Characteristic	Requirement
AI training and extensibility	Administrators shall be able to train or retrain the embedded AI and ML models using the texts and documents ingested into the system. The system shall allow new or improved AI, NLP and ML models to be integrated in a modular manner as they evolve. The integration mechanism shall support rollback and error handling in the event of a failed upgrade. Continual learning was recorded as particularly critical in a dynamic environment in which data change daily; it requires a retraining mechanism at regular intervals or on defined triggers, quality control of new data, and a safe testing environment for new models before they enter service.
Hardware	The system shall be supported by an infrastructure comprising a GPU suitable for inference and training (e.g. NVIDIA A100 or equivalent); 64 to 128 GB of RAM, depending on the size of the data and of the language models; a high-performance processor (e.g. Xeon or Ryzen Threadripper class); 1 to 2 TB of NVMe solid-state storage for fast reading and writing; and expandable storage of at least 10 TB for documents and training data. The software shall run on an on-premise server or workstation without connection to external clusters or cloud services, and shall support GPU-acceleration frameworks such as CUDA, PyTorch, TensorFlow or ONNX.

A.5 Implementation constraints

Three constraints were recorded. The maximum time available for implementation is twelve months. Technical staff is limited, which requires a high degree of automation, good documentation, usability for non-technical users, and possibly phased development with external assistance or outsourcing. No specific budget ceiling had been set at the time of elicitation; the stated objective was optimal performance rather than the cheapest solution, and the assessment of equipment and software costs was deferred to a later stage.

A.6 Prioritisation of functions

In view of the twelve-month horizon and the limited staff, the functional requirements were prioritised on three criteria: operational importance, technical complexity and interdependence with other functions. The stakeholders were asked which of the recorded functions they regarded as essential from the outset and which could be implemented in a second phase. Table A.4 records the outcome; it is the input to the Minimum Viable Product scope definition of Appendix B and to the use-case priorities of Table 3.

Table A.4 — Prioritisation of the elicited functions

Priority	Function (in order of importance)
High — to be implemented first	1. Storage of files of every type 2. Categorisation of documents by content 3. Content-based search of files 4. Entity extraction (names, locations, dates, events, coordinates) 5. Natural-language search 6. Automatic summarisation 7. Event timeline 8. Visualisations (histograms, maps)

Priority	Function (in order of importance)
Medium — second phase or gradual integration	1. Relationship diagrams between entities 2. Alerts for “unusual” events 3. Guided prediction 4. Training of the AI models with new data 5. Import or replacement of AI models

Appendix B: Minimum Viable Product scope definition

This appendix reproduces the Minimum Viable Product scope definition produced during Sub-step 2.5 of the methodology. It is referenced from Section 4.4. The definition was produced in Greek and is given here in the author's English translation; notes on team composition that formed part of the working document but have no bearing on the scope have been omitted. The scope stated here is the designed scope of the Minimum Viable Product; the scope actually implemented is reported in Section 4.4.9 and in Figure 22.

B.1 Purpose and definition

The purpose of the definition is to determine the first version of the system that can be delivered to its users with full operational capability, without including the entirety of the requested functions. A Minimum Viable Product, in the sense adopted here, includes only the critical functions that must work well for the system to have value; it is functional rather than a demonstration; it provides feedback from actual use, so as to guide the continuation of the project; and it reduces the technical and operational risk of the project as a whole.

B.2 Scope

Table B.1 lists the subsystems placed within the Minimum Viable Product and the functionality expected of each.

Table B.1 — Subsystems included in the Minimum Viable Product

Subsystem	Functionality
Document ingestion	Storage of PDF, DOCX and TXT files
AI named-entity-recognition engine	Extraction of entities: names, locations, dates
Search	Natural-language queries (basic understanding)
Summarisation	Automatic summarisation of documents
Timeline	Timeline derived from the extracted events and dates
Dashboard	Presentation of results in a simple interface
Users and roles	Role-based access (analyst, administrator)

Five subsystems were placed outside the Minimum Viable Product and reserved for later phases, beyond the scope of the dissertation: alerts for unusual events; predictive models (a prediction engine); training of the AI models through a graphical interface; the integration of new models; and advanced, graph-based relationship visualisations.

B.3 Estimated effort and phasing

The effort estimate recorded for the Minimum Viable Product is given in Table B.2, and the indicative monthly phasing in Table B.3. The phases of months four to six were expected to overlap.

Table B.2 — Effort estimate for the Minimum Viable Product

Phase	Estimate
Analysis and design	2–3 weeks
Development of the Minimum Viable Product	4–5 months
Testing and corrections	1 month
Total	approximately 6 months

Table B.3 — Indicative monthly phasing of the Minimum Viable Product

Month	Activities and deliverables
Month 1	Finalisation of requirements; usage analysis; selection of technologies; architectural design
Month 2	Set-up of the infrastructure (on-premise server or workstation); creation of the databases and of the document store
Month 3	Implementation of file upload; development of the backend API
Month 4	Integration of the AI models (named entity recognition and summarisation); analysis of files with entity extraction and summary
Month 5	Implementation of basic natural-language search; event timeline and basic visualisation
Month 6	Completion of the user interface; testing and optimisation; delivery of the Minimum Viable Product in working form

B.4 Technology direction

The technological direction recorded for the Minimum Viable Product is summarised in Table B.4, and the models proposed for the AI components in Table B.5. All models were required to be loadable locally, for offline inference. The backend was to be written in Python 3.10 or later with FastAPI, with multipart file upload and database access through SQLAlchemy (PostgreSQL) or PyMongo (MongoDB); metadata were to be held in PostgreSQL, documents in MongoDB GridFS, and Elasticsearch was optional for search. The front end was to be built with React or Vue, or, for a faster prototype, with Dash; visualisation was to use Plotly, Chart.js and Leaflet for maps, with export to PNG or PDF; authentication and security were to rest on JWT authentication, role-based access control and local file encryption.

Table B.4 — Technology direction for the Minimum Viable Product

Function	Proposed technology
Document ingestion	Python (Flask or FastAPI); file system or MongoDB GridFS
Analysis (named entity recognition, summarisation)	Hugging Face Transformers with spaCy or Flair
Storage of metadata	PostgreSQL or MongoDB
Search	Elasticsearch, or a simple vector store (FAISS)
Users and roles	JWT authentication or local authentication
User interface	React.js or Vue.js; Python Dash for a rapid solution

Table B.5 — Models proposed for the AI components of the Minimum Viable Product

Function	Proposed model	Library
Named entity recognition	dslim/bert-base-NER or spaCy en_core_web_trf	Hugging Face, spaCy
Summarisation	facebook/bart-large-cnn or t5-base	Transformers
Query parsing	Custom rules and keyword mapping, with embeddings for basic natural-language queries	sentence-transformers

B.5 Deliverables

At the end of the Minimum Viable Product the following were to be delivered: a web-based interface offering file upload, display of the extracted entities, a timeline, a summary and search; local-only operation; and documentation of installation and basic maintenance.

B.6 User-experience flow and screen descriptions

The intended flow for the analyst was recorded as follows: sign in with a role; upload a file by drag-and-drop or file selection; analysis of the file, producing a summary and the extracted entities; display of the results as a timeline, maps and tags; a natural-language question, for example “What happened in Alexandroupolis in May?”; and the answer, shown as a set of events. The screens were described in text so that they could later be drawn with a design tool or handed to a developer. The dashboard carries a navigation bar (Files, Search, Analyses, Settings, User and role) and an upload section with drag-and-drop or file selection for PDF, DOCX and TXT and a single “Upload and analyse” button. Tables B.6 to B.8 describe the remaining screens.

Table B.6 — Sign-in screen

Element	Description
Logo	Logo of the service
Username	Input field
Password	Input field
Sign-in button	Button
Roles	Determined from the sign-in (e.g. analyst or administrator)

Table B.7 — Document analysis screen

Element	Description
Document title	Shown at the top of the screen
Summary	Automatic summary of two to three paragraphs
Extracted entities	Table or list of names, locations, dates and coordinates
Timeline	Graphical representation of dates and events
Map	Optional; shown when geographical entities are present
Relations	Who is related to whom; only if implemented at a later stage

Table B.8 — Natural-language search screen

Element	Description
Question	E.g. “What happened in Xanthi in July?”
AI answer	Text with links to the relevant files
Visualisation	Timeline, map or table of events
Export	Button: export to PDF or Word

B.7 Developer task breakdown

The indicative breakdown of technical tasks, structured by module, was recorded as follows.

- Module 1 — File upload and storage: an API for the ingestion of PDF, DOCX and TXT files; storage of files on the local file system or in GridFS; extraction and storage of metadata (date, name).
- Module 2 — AI engine (named entity recognition and summarisation): integration of a named-entity-recognition model (e.g. a Hugging Face BERT model); integration of a summarisation model (e.g. BART or T5); pre-processing of the text extracted from the files; conversion of the extracted entities to JSON.

- Module 3 — Natural-language search: a search bar for natural language; keyword matching or embeddings for queries; return of the relevant documents, entities and events.
- Module 4 — Timeline visualisation: decomposition of events by date; display on a timeline (e.g. with Plotly or TimelineJS).
- Module 5 — User management and authentication: sign-in with roles (JWT or local authentication); differentiated access for the analyst and the administrator.
- Module 6 — Front-end interface: dashboard design in React or Vue; document analysis page; search page; sign-in screen.
- Module 7 — Security and logs: recording of actions (who uploaded or viewed what); role-based access to functions; encryption of files (optional).
- Module 8 — Documentation: README for installation; user guide; test scenarios.

Appendix C: Implementation plan

This appendix reproduces the step-by-step implementation plan produced during Sub-step 2.5 of the methodology. It is referenced from Section 4.4. The plan was produced in Greek and is given here in the author's English translation. Self-study material (recommended online courses) and indicative calendar durations, which formed part of the working document but have no bearing on the plan itself, have been omitted. The Minimum Viable Product that was eventually built departs from this plan in adopting a retrieval-augmented architecture (Sections 4.4.3 to 4.4.5) in place of the summarisation and timeline stages; those departures are documented in Section 4.4.9.

C.1 Approach

Because the work is a dissertation and the whole of the implementation is to be carried out by a single student, the approach differs from that of a team project: the work is broken into short, logically ordered and achievable steps, with priority given to understanding, simplicity and progressive learning. The stated priority is a working system with basic artificial-intelligence capabilities, not a complete operational product, and the technical approach is deliberately simplified. The plan is organised in five stages, each of which is stated below as a set of tasks.

C.2 Stage 1 — Preparation of the environment

Table C.1 — Stage 1 tasks

Task	Description
Create the workspace	Create the project folder on the development machine
Install Python 3.10 or later	From python.org
Install VS Code or JupyterLab	Development tools
Install the base packages	pip install transformers, spacy, fastapi, uvicorn, pandas, openpyxl, python-docx, pdfminer.six

C.3 Stage 2 — Document ingestion and analysis

Table C.2 — Stage 2 tasks

Task	Description
Implement a file-upload function	A folder in which PDF, DOCX and similar files are stored
Extract text from the files	Python with pdfminer and python-docx
Run named entity recognition	With spaCy or a Hugging Face model (e.g. bert-base-NER)
Run summarisation	The facebook/bart-large-cnn model
Store the results	As JSON or CSV; no database is needed in the Minimum Viable Product

C.4 Stage 3 — Basic interface (front end and back end)

Table C.3 — Stage 3 tasks

Task	Description
Create the FastAPI back end	A single route for upload that returns the summary and the entities
Create the web interface (React or Dash)	A dashboard with an upload button and display of the entities and the summary

Task	Description
Add natural-language search	E.g. input processed by regular expressions and searched against the JSON records

C.5 Stage 4 — Timeline and visualisation

Table C.4 — Stage 4 tasks

Task	Description
Use the dates from the named-entity-recognition output	Build a list of events with dates
Create the timeline	With Plotly or Matplotlib
Optionally add a map	If coordinates are extracted: Folium or Leaflet.js

C.6 Stage 5 — Testing, documentation and final presentation

Table C.5 — Stage 5 tasks

Task	Description
Test with real documents	Check the results
Write the README, screenshots and architecture description	Documentation of the artefact
Present the reduced Minimum Viable Product to the supervisor	Describe what can be extended subsequently

Appendix D: Evaluation harness

This appendix reproduces the evaluation harness referenced from Section 5.2 and Table 10, together with the ground-truth query set used for the technical validation corpus of Section 5.3 and the record of the harness’s execution on the target host. The harness executes the measurements listed in Table 10. It imports the production classes from the repository without modification, so that what is measured is the system and not a re-implementation of it. It records the environment in which it ran — processor, memory, library versions, thread count — alongside every value it reports, because a latency without a machine attached to it is not a result. It also states, in its own documentation, the things it does not measure and why, so that its output cannot be mistaken for a more complete evaluation than it is. As declared in the front matter, the code of the harness was generated by the AI coding assistant from the author’s specification and was reviewed, executed and interpreted by the author.

D.1 Invocation

The harness is invoked as a single command against a corpus directory and a ground-truth query file:

```
python evaluate.py --corpus ./eval_corpus --queries ./queries.json \
--repeats 5 --top-k 5 --out results.json --with-ollama
```

It writes a single machine-readable record, results.json, with five sections: the environment (A), the functional outcome of every pipeline stage on every document (B), per-stage latency statistics — median, mean and 95th percentile over the requested number of repeats (C), the retrieval measures Recall@k, MRR@k and nDCG@k against the declared ground truth (D), and a descriptive profile of entity counts by spaCy label (E). The optional --with-ollama flag additionally times the generation stage of phi3:mini for the first five queries. The record closes with an explicit list of what was not measured. The normalised discounted cumulative gain that the harness emits is not reported in Chapter 5, for the reason given in Section 5.3.2.

D.2 Ground truth and query set

The ground truth is a list of queries, each declaring the documents that contain its answer. A retrieved chunk counts as relevant if and only if its source document is one of them. This is known-item retrieval: it measures whether the system surfaces the right report, not whether it surfaces the best passage within it, and the harness states that limitation in its own output. The query set used for the technical validation corpus comprises fifteen queries, each answerable from exactly one of the twelve documents of Table 6; it is reproduced in Table D.1 from the file queries.json.

Table D.1 — Ground-truth query set for the technical validation corpus (queries.json)

ID	Query	Relevant document
Q01	Which architecture dispenses entirely with recurrence and convolutions and is based solely on attention mechanisms?	vaswani2017_transformer.pdf
Q02	What BLEU score was achieved on the WMT 2014 English-to-German translation task?	vaswani2017_transformer.pdf
Q03	Which distilled model reduces size by forty per cent while retaining ninety-seven per cent of language understanding?	sanh2019_distilbert.pdf
Q04	How does the model combine a parametric seq2seq memory with a non-parametric dense vector index of Wikipedia?	lewis2020_rag.pdf
Q05	How can a sixty-five billion parameter model be finetuned on a single 48GB GPU?	dettmers2023_qlora.pdf
Q06	What is 4-bit NormalFloat and double quantization?	dettmers2023_qlora.pdf

ID	Query	Relevant document
Q07	Which monolingual BERT model for modern Greek outperforms M-BERT and XLM-R?	koutsikakis2020_greekbert.pdf
Q08	How can secrets be leaked from air-gapped computers using covert radio signals generated from computer RAM?	guri2024_rambo.pdf
Q09	What are the policy engine, the policy administrator and the policy enforcement point?	rose2020_zerotrust_nist.pdf
Q10	Can a retrieval-augmented generation system be induced to leak the contents of its private retrieval database?	zeng2024_rag_privacy.pdf
Q11	How much did LoRA fine-tuning improve the F1 score for entity extraction from police reports?	xing2024_police_reports.pdf
Q12	Do off-the-shelf large language models show escalation patterns in simulated military wargames?	rivera2024_escalation.pdf
Q13	What is the design space of dashboards and how are dashboards used in practice?	sarikaya2019_dashboards.pdf
Q14	What is masked language model pre-training with next sentence prediction?	devlin2019_bert.pdf
Q15	What zero trust principles apply to access control when no implicit trust is granted based on network location?	rose2020_zerotrust_nist.pdf

D.3 Machine-readable record of the functional validation

The per-document figures of Table 6 are those of the functional section of results.json, the record written by the second execution of the harness on the target host on 19 September 2026. Listing D.1 reproduces that section; the timing fields are the per-stage wall-clock times reported in Table 6 and Figure 23. Listing D.3 reproduces the environment section of the same record, which is the configuration to which every figure of Section 5.3.2 is attributed.

Listing D.1 — results.json, functional section: per-document record behind Table 6 (second run, 19 September 2026)

```

1 {
2   "documents": 12,
3   "documents_ok": 12,
4   "total_chunks": 2147,
5   "per_document": [
6     {"file": "dettmers2023_qlora.pdf", "status": "OK", "chars": 87802, "chunks": 194,
7      "entities": 1196, "extract_s": 0.081, "chunk_s": 0.0, "embed_s": 19.186, "ner_s":
8      1.758},
9     {"file": "devlin2019_bert.pdf", "status": "OK", "chars": 64174, "chunks": 142,
10      "entities": 831, "extract_s": 0.047, "chunk_s": 0.0, "embed_s": 19.484, "ner_s": 1.301},
11     {"file": "guri2024_rambo.pdf", "status": "OK", "chars": 39740, "chunks": 88,
12      "entities": 372, "extract_s": 0.031, "chunk_s": 0.0, "embed_s": 9.971, "ner_s": 0.761},
13     {"file": "koutsikakis2020_greekbert.pdf", "status": "OK", "chars": 47277,
14      "chunks": 104, "entities": 766, "extract_s": 0.039, "chunk_s": 0.0, "embed_s": 15.208,
15      "ner_s": 0.883},
16     {"file": "lewis2020_rag.pdf", "status": "OK", "chars": 69085, "chunks": 153,
17      "entities": 925, "extract_s": 0.046, "chunk_s": 0.0, "embed_s": 18.717, "ner_s": 1.361},
18     {"file": "rivera2024_escalation.pdf", "status": "OK", "chars": 220545, "chunks":
19      485, "entities": 1383, "extract_s": 0.363, "chunk_s": 0.001, "embed_s": 46.797, "ner_s":
20      4.326},
21     {"file": "rose2020_zerotrust_nist.pdf", "status": "OK", "chars": 157095,
22      "chunks": 345, "entities": 533, "extract_s": 0.093, "chunk_s": 0.0, "embed_s": 33.612,
23      "ner_s": 2.359},
24     {"file": "sanh2019_distilbert.pdf", "status": "OK", "chars": 18527, "chunks": 41,
25      "entities": 296, "extract_s": 0.016, "chunk_s": 0.0, "embed_s": 3.692, "ner_s": 0.334},
26     {"file": "sarikaya2019_dashboards.pdf", "status": "OK", "chars": 78682, "chunks":
27      173, "entities": 575, "extract_s": 0.069, "chunk_s": 0.0, "embed_s": 14.348, "ner_s":
28      1.351},
29     {"file": "vaswani2017_transformer.pdf", "status": "OK", "chars": 39600, "chunks":
30      90, "entities": 483, "extract_s": 0.07, "chunk_s": 0.0, "embed_s": 8.909, "ner_s":
31      0.729},

```

```
16     {"file": "xing2024_police_reports.pdf", "status": "OK", "chars": 83098, "chunks":  
183, "entities": 917, "extract_s": 0.035, "chunk_s": 0.0, "embed_s": 20.405, "ner_s":  
1.41},  
17     {"file": "zeng2024_rag_privacy.pdf", "status": "OK", "chars": 66609, "chunks":  
149, "entities": 775, "extract_s": 0.048, "chunk_s": 0.0, "embed_s": 12.587, "ner_s":  
1.27}  
18 ]  
19 }
```

Listing D.3 — results.json, environment section: the configuration recorded by the harness at the second run

```
1 {  
2   "timestamp_utc": "2026-09-19T16:23:57Z",  
3   "python": "3.11.16",  
4   "platform": "Linux-6.12.54-linuxkit-aarch64-with-glibc2.41",  
5   "machine": "aarch64",  
6   "processor": "n/a",  
7   "cpu_count": 10,  
8   "ram_gb": 7.7,  
9   "version_torch": "2.1.2",  
10  "version_sentence_transformers": "2.3.1",  
11  "version_faiss": "1.7.4",  
12  "version_spacy": "3.7.2",  
13  "version_numpy": "1.24.3",  
14  "torch_threads": 10,  
15  "cuda_available": false  
16 }
```

D.4 Source of the harness

The complete source of the harness, `evaluate.py`, follows as Listing D.2. It is reproduced verbatim from the evaluation directory that accompanies the repository; the line numbers are editorial.

Listing D.2 — evaluate.py: the evaluation harness

```
1  #!/usr/bin/env python3  
2  """  
3  Kalhas – evaluation harness  
4  =====  
5  Reproducible evaluation of the retrieval pipeline of the Kalhas MVP.  
6  
7  It exercises the *actual* production classes from backend/ (no re-implementation):  
8      nlp.text_extractor.TextExtractor  
9      nlp.chunker.TextChunker  
10     nlp.entity_extractor.EntityExtractor  
11     retrieval.embeddings.EmbeddingGenerator  
12     retrieval.vector_store.VectorStore      (an in-memory twin, to avoid touching  
the live index)  
13  
14     It reports, for a corpus of documents and a set of known-item queries:  
15  
16     A. ENVIRONMENT      – CPU, RAM, OS, Python, library versions  (so the numbers are  
attributable)  
17     B. FUNCTIONAL      – does every stage complete on every document?  
18     C. LATENCY         – per-stage wall-clock, median and p95, over N repeats  
19     D. RETRIEVAL       – Recall@k, MRR@k, nDCG@k against a declared ground truth  
20     E. NER PROFILE     – entity counts by spaCy label (descriptive; NOT an accuracy  
claim)  
21  
22     WHAT THIS SCRIPT DOES NOT DO  
23     -----  
24     * It does not measure NER precision/recall/F1. That needs a human-annotated  
corpus,  
25     which does not exist for this domain and language. Do not invent one.  
26     * It does not measure summarisation quality. The MVP contains no summariser.  
27     * It does not measure the Ollama generation stage (quality or latency) unless  
28     --with-ollama is passed and an Ollama server is reachable.  
29
```

```
30 USAGE
31 -----
32     python evaluate.py --corpus ./eval_corpus --queries ./queries.json --repeats 5
--out results.json
33     python evaluate.py ... --with-ollama          # additionally times phi3:mini
generation
34
35 GROUND TRUTH FORMAT (queries.json)
36 -----
37     [
38         {"query": "which model is used for embeddings?",
39          "relevant_docs": ["architecture.md"]}, # filenames, as in the
corpus dir
40     ...
41     ]
42 A retrieved chunk counts as relevant iff its source document is in relevant_docs.
43 This is *known-item retrieval*: it measures whether the system surfaces the right
44 document, not whether it surfaces the best passage inside it. State that
limitation.
45 """
46 import argparse, json, os, platform, statistics, sys, time
47 from pathlib import Path
48
49 def _env():
50     info = {
51         "timestamp_utc": time.strftime("%Y-%m-%dT%H:%M:%SZ", time.gmtime()),
52         "python": sys.version.split()[0],
53         "platform": platform.platform(),
54         "machine": platform.machine(),
55         "processor": platform.processor() or "n/a",
56         "cpu_count": os.cpu_count(),
57     }
58     try:
59         with open("/proc/cpuinfo") as f:
60             for line in f:
61                 if line.lower().startswith("model name"):
62                     info["cpu_model"] = line.split(":", 1)[1].strip(); break
63     except Exception:
64         pass
65     try:
66         with open("/proc/meminfo") as f:
67             for line in f:
68                 if line.startswith("MemTotal"):
69                     info["ram_gb"] = round(int(line.split()[1]) / 1048576, 1);
break
70     except Exception:
71         pass
72     for mod in ("torch", "sentence_transformers", "faiss", "spacy", "numpy"):
73         try:
74             m = __import__(mod)
75             info[f"version_{mod}"] = getattr(m, "__version__", "?")
76         except Exception:
77             info[f"version_{mod}"] = "not installed"
78     try:
79         import torch
80         info["torch_threads"] = torch.get_num_threads()
81         info["cuda_available"] = torch.cuda.is_available()
82     except Exception:
83         pass
84     return info
85
86 def _stats(xs):
87     if not xs: return {}
88     xs = sorted(xs)
89     return {
90         "n": len(xs),
91         "median_ms": round(statistics.median(xs) * 1000, 1),
92         "mean_ms": round(statistics.fmean(xs) * 1000, 1),
93         "p95_ms": round(xs[min(len(xs) - 1, int(0.95 * len(xs)))] * 1000, 1),
94         "min_ms": round(xs[0] * 1000, 1),
95         "max_ms": round(xs[-1] * 1000, 1),
96     }
```

```

97
98 def main():
99     ap = argparse.ArgumentParser()
100     ap.add_argument("--corpus", required=True, help="directory
of .txt/.md/.pdf/.docx documents")
101     ap.add_argument("--queries", required=True, help="queries.json with ground
truth")
102     ap.add_argument("--repeats", type=int, default=5, help="repeats for latency
statistics")
103     ap.add_argument("--top-k", type=int, default=5)
104     ap.add_argument("--out", default="results.json")
105     ap.add_argument("--backend", default="./backend", help="path to the backend
package")
106     ap.add_argument("--with-ollama", action="store_true")
107     ap.add_argument("--ollama-model", default="phi3:mini")
108     a = ap.parse_args()
109
110     sys.path.insert(0, str(Path(a.backend).resolve()))
111     os.environ.setdefault("MODELS_PATH", "/tmp/kalhas_models")
112     os.environ.setdefault("EMBEDDINGS_PATH", "/tmp/kalhas_eval_index")
113
114     from nlp.text_extractor import TextExtractor
115     from nlp.chunker import TextChunker
116     from nlp.entity_extractor import EntityExtractor
117     from retrieval.embeddings import EmbeddingGenerator
118     import faiss, numpy as np
119
120     R = {"environment": _env(), "config": vars(a)}
121     print("== A. ENVIRONMENT =="); print(json.dumps(R["environment"], indent=2))
122
123     extractor = TextExtractor()
124     chunker = TextChunker() # 512 chars / 50 overlap – the
production default
125     embedder = EmbeddingGenerator() # all-MiniLM-L6-v2
126     ner = EntityExtractor() # en_core_web_sm
127
128     docs = sorted([p for p in Path(a.corpus).iterdir()
129                     if p.suffix.lower() in (".txt", ".md", ".pdf", ".docx")])
130     if not docs:
131         sys.exit(f"no documents in {a.corpus}")
132
133     # ----- B. FUNCTIONAL + C. LATENCY (ingestion) -----
134     t_extract, t_chunk, t_embed, t_ner = [], [], [], []
135     per_doc, all_chunks, chunk_owner, ent_profile = [], [], [], {}
136     dim = embedder.get_embedding_dimension()
137     index = faiss.IndexFlatIP(dim)
138
139     for p in docs:
140         t0 = time.perf_counter(); text = extractor.extract(str(p)); t1 =
time.perf_counter()
141         if not text:
142             per_doc.append({"file": p.name, "status": "EXTRACTION_FAILED"});
continue
143         chunks = chunker.chunk(text); t2 = time.perf_counter()
144         texts = [c["text"] for c in chunks]
145         embs = embedder.generate_embeddings(texts); t3 = time.perf_counter()
146         ents = ner.extract_entities(text); t4 = time.perf_counter()
147
148         index.add(np.asarray(embs, dtype="float32"))
149         chunk_owner += [p.name] * len(chunks)
150         all_chunks += texts
151
152         for e in ents:
153             ent_profile[e["type"]] = ent_profile.get(e["type"], 0) + 1
154
155         t_extract.append(t1 - t0); t_chunk.append(t2 - t1)
156         t_embed.append(t3 - t2); t_ner.append(t4 - t3)
157         per_doc.append({"file": p.name, "status": "OK", "chars": len(text),
158                         "chunks": len(chunks), "entities": len(ents),
159                         "extract_s": round(t1 - t0, 3), "chunk_s": round(t2 - t1,
3),
160                         "embed_s": round(t3 - t2, 3), "ner_s": round(t4 - t3, 3)})

```

```

161         print(f" [{'OK ' if chunks else 'EMPTY'}] {p.name}: {len(text)} chars -> "
162               f"{len(chunks)} chunks, {len(ents)} entities")
163
164     R["functional"] = {"documents": len(docs),
165                      "documents_ok": sum(1 for d in per_doc if d["status"] ==
166                      "OK"),
167                      "total_chunks": len(all_chunks),
168                      "per_document": per_doc}
169     R["latency_ingestion_per_document"] = {
170         "text_extraction": _stats(t_extract), "chunking": _stats(t_chunk),
171         "embedding": _stats(t_embed), "ner": _stats(t_ner)}
172     R["ner_profile_counts_by_label"] = dict(sorted(ent_profile.items(),
173         key=lambda kv: -kv[1]))
174
175     # ----- C. LATENCY (query) + D. RETRIEVAL -----
176     queries = json.loads(Path(a.queries).read_text(encoding="utf-8"))
177     t_qemb, t_search = [], []
178     recalls, rrs, ndcgs, detail = [], [], [], []
179     import math
180
181     for q in queries:
182         hits_rank = None
183         for r in range(a.repeats):
184             t0 = time.perf_counter()
185             qv =
186             embedder.generate_embedding(q["query"]).astype("float32").reshape(1, -1)
187             t1 = time.perf_counter()
188             scores, idx = index.search(qv, a.top_k)
189             t2 = time.perf_counter()
190             t_qemb.append(t1 - t0); t_search.append(t2 - t1)
191             if r == 0:
192                 retrieved = [chunk_owner[i] for i in idx[0] if i >= 0]
193                 rel = set(q["relevant_docs"])
194                 hit = [1 if d in rel else 0 for d in retrieved]
195                 recalls.append(1.0 if any(hit) else 0.0)
196                 rank = hit.index(1) + 1 if any(hit) else 0
197                 rrs.append(1.0 / rank if rank else 0.0)
198                 dcg = sum(h / math.log2(i + 2) for i, h in enumerate(hit))
199                 ideal = sum(1 / math.log2(i + 2) for i in range(min(len(rel),
200                 a.top_k)))
201                 ndcgs.append(dcg / ideal if ideal else 0.0)
202                 detail.append({"query": q["query"], "relevant": sorted(rel),
203                 "retrieved_docs": retrieved, "first_hit_rank": rank,
204                 "top_scores": [round(float(s), 3) for s in
205                 scores[0]]})
206
207     R["latency_query"] = {"query_embedding": _stats(t_qemb),
208                          "faiss_search": _stats(t_search),
209                          "note": f"index size = {index.ntotal} vectors, dim =
210                          {dim}"})
211     R["retrieval"] = {
212         "n_queries": len(queries), "top_k": a.top_k,
213         f"recall_at_{a.top_k}": round(statistics.fmean(recalls), 3) if recalls else
214         None,
215         f"mrr_at_{a.top_k}": round(statistics.fmean(rrs), 3) if rrs else None,
216         f"ndcg_at_{a.top_k}": round(statistics.fmean(ndcgs), 3) if ndcgs else
217         None,
218         "metric_definition": ("known-item retrieval: a retrieved chunk is relevant
219         iff its "
220         "source document is in relevant_docs. Measures
221         document-level "
222         "retrieval, not passage-level relevance."),
223         "per_query": detail}
224
225     # ----- optional: generation stage -----
226     if a.with_ollama:
227         try:
228             import ollama
229             client = ollama.Client(host=os.getenv("OLLAMA_URL",
230             "http://localhost:11434"))
231             t_gen = []
232             for q in queries[:5]:

```

```
223         t0 = time.perf_counter()
224         client.generate(model=a.ollama_model, prompt=q["query"],
225                        options={"temperature": 0.3, "num_predict": 200})
226         t_gen.append(time.perf_counter() - t0)
227         R["latency_generation"] = {"model": a.ollama_model, **_stats(t_gen)}
228     except Exception as e:
229         R["latency_generation"] = {"error": f"{type(e).__name__}: {e}"}
230
231     R["not_measured"] = [
232         "NER precision/recall/F1 – requires a human-annotated ground truth corpus
233         (does not exist).",
234         "Summarisation quality – the MVP implements no summariser.",
235         "Relation and event extraction – not implemented in the MVP.",
236         "Generation faithfulness / hallucination rate – requires human judgement.",
237         "Usability (ISO/IEC 25010 quality in use) – requires participants.",
238     ]
239     Path(a.out).write_text(json.dumps(R, indent=2, ensure_ascii=False),
240                          encoding="utf-8")
241     print("\n== SUMMARY ==")
242     print(json.dumps({k: R[k] for k in
243                      ("latency_ingestion_per_document", "latency_query",
244                      "retrieval")
245                      if k in R}, indent=2, default=str)[:2000])
246     print(f"\nfull results -> {a.out}")
247
248 if __name__ == "__main__":
249     main()
```

D.5 Record of the executions

The harness was executed twice on the target host on 19 September 2026, in both cases from inside the rag-backend container of the deployed Docker Compose stack, with the invocation of Section D.1 (--repeats 5, --top-k 5, --with-ollama) against the twelve-document corpus and the fifteen queries of Table D.1.

The first execution, recorded at 15:55 UTC, ran without the spaCy language model `en_core_web_sm`, which was not present in the container at that time; the EntityExtractor class degraded as designed and returned no entities for any document, so that the entity profile of that run is empty. Its record is retained as `results_run1_no_NER.json` in the evaluation directory. The second execution, recorded at 16:23 UTC after the model had been installed, completed every stage for every document and is the run reported in Chapter 5; its record is `results.json`.

The two records agree exactly on every deterministic quantity — characters and chunks per document, Recall@5, MRR@5 and the per-query similarity scores — and differ in the latencies, as discussed in Section 5.3.2: total embedding time 133.6 s against 222.9 s, median query encoding 93.3 ms against 113.4 ms, median generation 17.4 s against 40.6 s. Both records are kept so that the comparison can be reproduced.

Author’s Statement:

I hereby expressly declare that, according to the article 8 of Law 1559/1986, this dissertation is solely the product of my personal work, does not infringe any intellectual property, personality and personal data rights of third parties, does not contain works/contributions from third parties for which the permission of the authors/beneficiaries is required, is not the product of partial or total plagiarism, and that the sources used are limited to the literature references alone and meet the rules of scientific citations.