Performance Enhancement Techniques for Information Extraction with Small Language Models

Sebastian Deatc

Delft University of Technology



TU Delft

Performance Enhancement Techniques for Information Extraction with Small Language Models

by

Sebastian Deatc

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 31, 2026 at 14:00.

Student number:	5445094	
Project duration:	November 1, 2025 – August 31, 2026	
Thesis committee:	Prof. Dr. A. Katsifodimos,	TU Delft, supervisor
	Dr. ir. J. Yang,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Information extraction from long and complex documents is an important task in domains such as healthcare and law. This study investigates how performance-enhancing techniques can improve information extraction in smaller language models.

We develop a language-model-based extraction pipeline that combines retrieval-augmented generation, structured extraction using schema-driven prompting, and post-extraction filtering. The system is evaluated on a legal dataset (Kleister NDA) and a real-world Dutch medical dataset consisting of general practitioner referral letters. We study the impact of retrieval depth, context size, chunk ordering, and a post-processing large language model (LLM) filter across multiple open and closed-source models.

The results show that retrieval is highly effective, with full coverage achieved using only a small number of retrieved chunks. However, increasing context size does not consistently improve performance, as it introduces a trade-off between precision and recall. Intermediate context sizes typically yield the best balance. Chunk ordering has a smaller but model-dependent effect, while different models show distinct behaviors ranging from conservative extraction to strong sensitivity to context composition.

Finally, a lightweight post-extraction filtering step significantly improves precision while preserving recall, substantially improving overall extraction performance. However, a performance gap remains compared to a state-of-the-art large model benchmark.

Overall, the study shows that smaller language models can achieve strong extraction performance when supported by carefully designed pipeline components, context design, and post-processing filters.

Contents

Abstract i

1 Introduction 1

2 Background & Related Work 3
2.1 Traditional Information Extraction . . . 3
2.2 LLM-based Information Extraction . . . 3
2.3 Retrieval Augmented Generation . . . 4
2.4 Lost in the Middle . . . 4

3 Approach & Design 5
3.1 Preprocessing . . . 5
3.1.1 Text Extraction . . . 6
3.1.2 Chunking . . . 6
3.1.3 Retrieval . . . 7
3.2 Extraction . . . 7
3.3 Post-Processing . . . 8
3.4 Capabilities, Trade-Offs and Constraints . . . 9

4 Experimental Setup 10
4.1 Research Questions . . . 10
4.2 Evaluation . . . 11
4.2.1 LLM-as-a-Judge . . . 11
4.2.2 Metrics . . . 11
4.3 Dataset . . . 11
4.3.1 Kleister NDA Dataset . . . 11
4.3.2 Donor General Practitioner Letters Dataset . . . 12
4.4 Models and Parameters . . . 12
4.4.1 Language Models . . . 12
4.4.2 Embedder . . . 13
4.5 Benchmark . . . 13

5 Results 14
5.1 Retrieval Coverage and Minimum Retrieval Depth . . . 14
5.2 Context Size and Chunk Ordering Effects . . . 14
5.2.1 Kleister NDA . . . 14
5.2.2 Donor GP Letter . . . 16
5.2.3 Overall Performance . . . 20
5.3 Post-Extraction LLM Filtering . . . 22

6 Discussion 24
6.1 Interpretation . . . 24
6.2 Threats to Validity . . . 25
6.3 Responsible Research . . . 26
6.4 Use of AI . . . 26

7 Conclusion 27

References 29

A Benchmark Prompt 32

1

Introduction

Information Extraction (IE) has become an important task in many fields, including the medical domain [23]. Recent advances in Large Language Models (LLMs) have led to increased research on applying these models to information and data extraction tasks [6, 21]. However, limited attention has been given to the use of LLM-based information extraction in environments with restricted computational resources. In practice, many medical companies do not have access to high-performance computing infrastructure. In addition, because they work with sensitive medical data, they are often unable to rely on large state-of-the-art LLMs accessed through external APIs. As a result, they must rely on smaller language models (SLMs) or compact LLMs, which typically suffer more from hallucinations and degraded task performance due to their limited parameter count [35].

There are several techniques that could improve the performance of these SLMs. These include retrieval-augmented generation (RAG) [25, 12], schema-driven scaffolding [36], and multi-pass inference strategies [10, 22].

A similar study was conducted by by Wiest et al., who propose a privacy-preserving framework for medical information extraction using large language models [39]. Their system enables local deployment in healthcare settings and is evaluated on clinical entity extraction from medical reports, demonstrating the feasibility of LLM-based extraction while keeping sensitive data in-house.

The main research question of this study is: ***To what extent can performance-enhancing techniques improve the information extraction capabilities of smaller language models?***. The objective is to determine whether smaller language models can achieve information extraction performance comparable to that of larger state-of-the-art models when supported by techniques such as retrieval-augmented generation, schema-driven scaffolding, and multi-pass reasoning. To investigate this, we develop a language-model-based information extraction pipeline that incorporates these techniques and systematically evaluate their impact on extraction performance. The main research question can be split into the following three research questions:

1. *How well can vector retrieval locate target entities in legal and medical documents, and how many retrieved chunks are needed to achieve full recall?*
2. *How do context size and chunk ordering in a RAG pipeline affect extraction performance?*
3. *To what extent can a post-extraction LLM filtering step improve the quality of extraction?*

We evaluate the proposed pipeline using both the Kleister NDA dataset [37] and a private real-world medical donor dataset containing general practitioner referral letters. The Kleister NDA dataset provides a complex and realistic benchmark that resembles the dense administrative documents often encountered in healthcare settings, while avoiding the privacy concerns associated with real patient data. The donor dataset complements this by enabling evaluation on authentic medical documents with domain-specific language and structure. Furthermore, to overcome the brittleness of standard exact-match metrics on text containing Optical Character Recognition (OCR) noise and variability in

language model outputs, we introduce a semantic LLM-as-a-Judge evaluation framework to accurately measure precision, recall, and F1-scores.

The main contributions of this study are as follows:

- A systematic empirical evaluation of four widely used large language models for information extraction, investigating the effects of context filling, chunk ordering, and post-processing filtering across two high-quality, realistic datasets.
- An open-source framework for information extraction, including experiment orchestration tools and an LLM-as-a-judge evaluation module [1].
- A benchmark for LM-based information extraction in medical and legal domains, designed to reflect state-of-the-art extraction performance.

The main finding of this report is that LM-based information extraction using smaller language models can achieve state-of-the-art large-model recall, while showing a modest decrease in precision.

This report will follow the following structure. In Section 2 we will discuss the related work and the knowledge gap this research will cover. In Section 3 we will outline the approach of the research and display a high-level overview of our pipeline. In Section 4 we will detail the experimental setup and list the sub-questions and the hypotheses that we will examine during the research. Following that, in Section 5 we will present the results. In Section 6 we will discuss the research. Finally, in Section 7 we will conclude the research and discuss possible future work

[1] https://github.com/SebastianDeatc/LLM_information_extraction

2

Background & Related Work

2.1. Traditional Information Extraction

Information extraction has evolved through several paradigms, each addressing limitations of its predecessors. Early information extraction systems were primarily rule-based, relying on manually crafted patterns, regular expressions, and domain-specific rules to identify relevant information. Over time, these systems evolved to include methods for automatically inducing rules from annotated examples, reducing the manual effort required to build extraction systems [34, 7]. Rule-based approaches can achieve high precision within well-defined domains, but they often struggle to generalize. As documents become longer, more diverse, and linguistically complex, maintaining large rule sets becomes increasingly difficult, making these systems brittle when faced with unseen writing styles or document structures [34].

To overcome these limitations, statistical learning approaches became widely adopted. Rather than relying on handcrafted rules, these methods learn extraction patterns directly from annotated data. Prominent examples include generative models based on Hidden Markov Models, discriminative models based on Maximum Entropy, and global conditional models such as Conditional Random Fields (CRFs) [34]. Compared to rule-based systems, statistical approaches generally offer better robustness and adaptability. However, they require substantial amounts of labeled training data and often rely on task-specific feature engineering.

More recently, deep learning approaches have become a popular paradigm for information extraction due to their ability to automatically learn rich semantic representations from text [41]. Neural architectures substantially reduce the need for manual feature engineering and have achieved strong performance across many extraction tasks [21]. Nevertheless, these models require (annotated) training data, and may struggle with long or highly complex documents, particularly when relevant information is distributed across large contexts [26, 38].

The emergence of Large Language Models (LLMs) represents a further shift in information extraction. Rather than training task-specific extraction models, LLMs can perform extraction through natural language instructions and few-shot examples, allowing them to generalize across domains with little or no additional training. Their ability to reason over unstructured text makes them particularly attractive for complex extraction tasks involving long, heterogeneous documents. However, this flexibility introduces new challenges, including hallucinations, context limitations, and sensitivity to prompt and retrieval design. These challenges motivate the work presented in this study, which investigates how retrieval-augmented generation and other pipeline design choices influence the information extraction performance of smaller language models.

2.2. LLM-based Information Extraction

A similar study was conducted by Wiest et al. [39]. The authors present a privacy-preserving framework for medical information extraction using large language models. Their pipeline is designed for

local deployment in healthcare environments, ensuring that sensitive patient data remains within hospital infrastructure rather than being processed through external APIs. The framework is evaluated on clinical entity extraction tasks using medical reports. This work is closely related to our research, as both studies focus on medical information extraction under privacy constraints. However, while Wiest et al. primarily propose a practical extraction framework, our work systematically evaluates how retrieval-augmented generation, context size, chunk ordering, and post-extraction filtering influence the performance of smaller language models across both medical and legal datasets. In addition, their experiments focus on a single large model (Llama-3 70B), whereas we evaluate multiple smaller language models and compare their performance against a state-of-the-art benchmark.

Another similar work explores the use of large language models for structured information extraction in document-heavy workflows [3]. Ateia et al. investigate how LLMs can support scientific literature research and publication processes by automatically extracting structured information from academic papers. Their work demonstrates that LLMs can reduce manual effort in processing scientific documents by identifying and organizing relevant entities and metadata. While this study focuses on general scientific literature and workflow automation, our work instead targets medical and legal documents and provides a more systematic analysis of how retrieval configuration, context size, chunk ordering, and post-extraction filtering affect extraction performance in smaller language models.

2.3. Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) combines external information retrieval with large language model generation to improve factual accuracy and reduce hallucinations. Instead of relying only on parametric knowledge, a retriever first selects relevant documents from an external corpus, which are then provided as additional context to the model during generation [12, 25, 43]. Gao et al. provide a comprehensive overview of RAG systems, describing them as a general framework that integrates retrieval, reranking, and generation components [12]. They highlight that RAG systems can significantly improve performance on knowledge-intensive tasks by grounding model outputs in retrieved evidence, especially when dealing with long or domain-specific documents. However, the survey also emphasizes that RAG performance is highly sensitive to design choices such as chunking strategy, retrieval depth, and how retrieved context is integrated into the prompt. Poor retrieval or noisy context can reduce performance or introduce irrelevant information, making system design a key factor in overall effectiveness. In this work, we build on this insight by systematically analyzing how retrieval depth, chunk ordering, and context scaling affect extraction performance in smaller language models.

Several extensions to the traditional RAG framework have been proposed, each aiming to improve retrieval quality or generation accuracy. Graph-based RAG methods organize documents or entities into graph structures, enabling more effective multi-hop retrieval and reasoning over interconnected information [17, 30, 18]. Active RAG approaches dynamically determine when and what information to retrieve throughout the generation process, rather than relying on a single retrieval step before generation [20]. Similarly, iterative or self-reflective RAG methods repeatedly retrieve additional evidence on demand, using intermediate reasoning or self-evaluation to identify missing information and refine the generated response [2]. These approaches can improve performance on complex, multi-step tasks by allowing retrieval to adapt as generation progresses.

Despite these advances, this work adopts a traditional retrieval-then-generation architecture. Restricting the system to a conventional RAG pipeline allows us to isolate the effects of retrieval depth, chunk ordering, and context scaling without introducing additional variables associated with more sophisticated retrieval strategies.

2.4. Lost in the Middle

A study closely related to our work is that of Liu et al. [26], who show that LLMs are better at using information placed at the beginning of a prompt (”primacy bias”) and at the end of a prompt (”recency bias”) than information located in the middle. This is particularly relevant to our work because different retrieval configurations affect both the position of the retrieved information within the prompt and the amount of context provided. We investigate whether similar ”lost in the middle” patterns appear in the context of information extraction.

3

Approach & Design

The pipeline consists of three main stages: preprocessing, extraction, and post-processing. Figure 3.1 provides an overview of the system architecture and workflow. In the full configuration, text-based input passes through these stages sequentially. However, depending on the experimental setting, not all components are used in every run, and certain steps may be omitted. The exact configurations for each experiment are described in detail in Section 4. This section focuses on each stage of the pipeline, including the design choices made and the rationale behind them.

Rather than maximizing benchmark performance through increasing complexity, the primary design objective of this pipeline is modularity and controlled experimentation. Each stage is designed as an independent component, allowing chunking strategies, embedding models, retrieval mechanisms, prompting strategies, and post-processing techniques to be substituted without affecting the remainder of the pipeline. This modular architecture makes it possible to isolate individual design decisions and systematically evaluate their impact on information extraction performance, which forms the central objective of this thesis.

3.1. Preprocessing

The preprocessing stage is responsible for transforming raw documents into a representation suitable for retrieval and extraction. Although this stage is often viewed as a purely technical preprocessing step, it has a substantial influence on downstream performance. Design choices such as document parsing, chunking strategy, and retrieval configuration determine what information is ultimately presented to the language model.

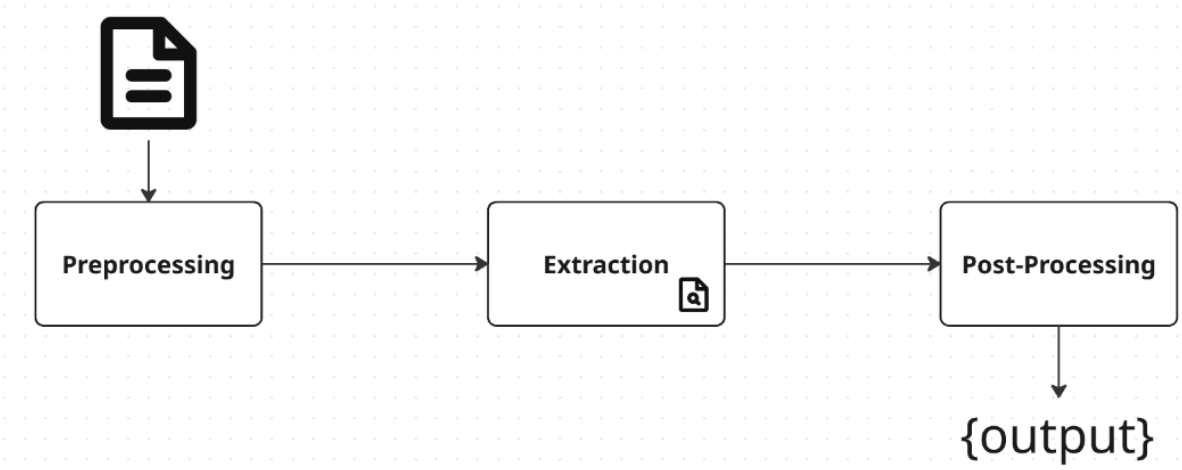


Figure 3.1: Pipeline Design. A high level overview of the stages in our pipeline.

3.1.1. Text Extraction

The pipeline operates exclusively on textual input. For raw PDF documents, text is extracted using Docling[1], a document understanding framework capable of converting PDFs into structured textual representations. This preprocessing step is not required for experiments using the Kleister dataset [37], as these documents are already distributed as extracted text.

Choosing a text-focused pipeline was a deliberate design decision. By converting documents into plain text before retrieval and extraction, the pipeline becomes largely independent of document layout and can operate with any language model capable of processing text. Furthermore, separating document parsing from information extraction allows improvements in either component without requiring changes to the remainder of the system.

However, this choice also introduces limitations. The pipeline does not explicitly reason over visual information such as tables, handwritten annotations, signatures, diagrams, or complex page layouts. While Docling attempts to reconstruct structured elements such as tables during PDF parsing, the quality of the extracted representation depends on the accuracy of the parsing process. Consequently, extraction performance for tabular content is inherently limited by Docling's ability to correctly interpret and serialize these structures into text. Any information contained exclusively within these visual elements may therefore be unavailable to the extraction model. Moreover, PDF parsing is inherently imperfect, and OCR or layout reconstruction errors may introduce missing words, incorrect formatting, or misplaced text before the extraction stage even begins [19]. These preprocessing errors propagate throughout the remainder of the pipeline and may negatively influence both retrieval quality and extraction accuracy.

3.1.2. Chunking

Although modern language models support increasingly large context windows, processing entire documents is not always desirable. Long contexts increase computational cost, may introduce irrelevant information, and can reduce the model's ability to effectively utilize relevant evidence due to attention dilution and positional effects. Chunking therefore represents one of the most important design choices within retrieval-augmented generation systems, as it determines the granularity at which information is retrieved and presented to the language model.

We can use several chunking strategies. Size-based chunking divides documents into equally sized segments, offering predictable chunk distributions and efficient indexing. Sentence-based or recursive approaches attempt to preserve linguistic structure by splitting on natural boundaries, while semantic chunking groups text according to changes in semantic similarity. Each approach represents a trade-off between preserving contextual coherence and maximizing retrieval specificity [31].

The pipeline developed in this thesis has both size-based and semantic chunking strategies. Semantic chunking is implemented by computing embedding representations for consecutive text segments and splitting whenever the cosine similarity between adjacent segments falls below a configurable threshold. Although this strategy was implemented as part of the framework, it is not evaluated experimentally within this thesis.

For all experiments, we instead employ size-based chunking using segments of 512 tokens with a 15% overlap between consecutive chunks. Size-based chunking was selected primarily because it provides deterministic, predictable and reproducible chunk boundaries across all experiments, allowing retrieval depth and context size to be studied without introducing additional unpredictability caused by adaptive chunk boundaries. The overlap further reduces context fragmentation by preserving information that spans chunk boundaries.

An important design objective of the framework is modularity. Since chunking is implemented as an independent preprocessing component, alternative chunking strategies can easily be substituted in future work without requiring modifications to the retrieval or extraction stages.

[1] https://docling-project.github.io/docling/

3.1.3. Retrieval

After chunking, all document segments are embedded and indexed using Qdrant[2], a vector database optimized for efficient similarity search. During retrieval, the embedding of the user query is compared against the indexed chunk embeddings, after which the top-k most similar chunks are returned.

As discussed in Section 2.3, Retrieval-Augmented Generation has evolved considerably in recent years. Besides traditional retrieval-then-generation pipelines, recent work has explored graph-based retrieval, iterative retrieval, and active retrieval strategies that dynamically acquire additional evidence throughout generation. While these methods may improve retrieval quality on complex reasoning tasks, they also introduce additional components whose behavior may influence extraction performance.

In this work, we intentionally adopt a traditional retrieval-augmented generation pipeline. Restricting retrieval to a single similarity search allows the experiments to isolate the effects of retrieval depth, chunk ordering, and context scaling without introducing additional confounding variables associated with more sophisticated retrieval mechanisms.

Finally, retrieved chunks can either be ordered according to their similarity score or according to their original position within the source document. Since transformer-based language models exhibit positional biases when processing long contexts [26], both ordering strategies are evaluated experimentally later in this thesis.

3.2. Extraction

The objective of the extraction stage is to transform unstructured textual input into structured information while minimizing hallucinations and formatting inconsistencies. A straightforward approach would be to prompt a language model directly and subsequently parse its textual output. However when using SLMs, free-form prompting often results in inconsistent formatting, malformed JSON, omitted fields, or hallucinated values, making downstream processing considerably more difficult.

Therefore we use schema-constrained generation, where the language model is explicitly guided toward producing structured outputs that conform to a predefined schema. This approach improves robustness while simultaneously simplifying downstream evaluation and integration into larger information extraction pipelines [4].

For the extraction stage we use LangExtract [13]. While implementing a custom extraction framework was considered, LangExtract provides robust scaffolding, constrained output generation, source grounding, and visualization capabilities (see Figure 3.2). This allows the work presented in this thesis to focus primarily on retrieval strategies and pipeline design rather than sophisticated output parsing. Furthermore, source grounding is particularly valuable in high-risk application domains such as healthcare and law, where extracted information should remain traceable to supporting evidence rather than being accepted without verification [11].

More specifically, LangExtract manages the extraction lifecycle through Schema-Driven Scaffolding. Rather than relying solely on conversational prompting, the framework first serializes the desired extraction schema into a strict JSON Schema. This schema is injected into the system prompt together with task-specific extraction instructions and annotated few-shot examples illustrating the desired output format. During inference, the framework binds this schema to the language model's native structured output interface using the `use_schema_constraints=True` parameter. As a result, output generation is constrained to syntactically valid JSON objects that conform to the predefined schema, substantially reducing formatting failures and simplifying automated evaluation.

Although schema constraints improve structural reliability, prompt design remains an important consideration. Highly restrictive prompts generally improve precision by discouraging uncertain predictions, but may also reduce recall by causing borderline entities to be omitted. Conversely, more permissive prompts often improve recall but increase the likelihood of hallucinated or incorrect extractions. Since false positives can subsequently be removed during post-processing whereas missed entities cannot be recovered, the prompts used throughout this work intentionally favors recall while still including explicit extraction rules intended to reduce obvious hallucinations. This design philosophy aligns with the

[2] https://qdrant.tech/

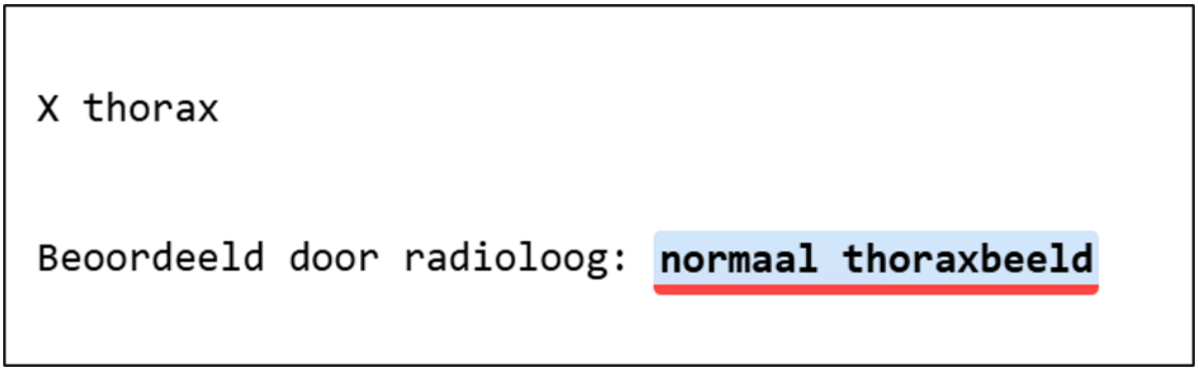


Figure 3.2: LangExtract Visualisation Example.

overall objective of maximizing retrieval coverage.

An additional advantage of this design is model independence. Because LangExtract abstracts much of the structured generation process, the same extraction pipeline can be applied across different language models with only minimal configuration changes, allowing model comparisons to remain focused on extraction capability rather than implementation differences.

3.3. Post-Processing

The post-processing stage serves two primary purposes: improving the reliability of extracted information and increasing the explainability of the overall system. Rather than treating the output of the language model as a final prediction, the pipeline optionally performs additional verification and enriches extraction results with supporting metadata.

At its core, the system produces a structured JSON output in which every extracted entity is linked to its original position within the source document. Optionally, LangExtract's visualization functionality can generate an interactive HTML representation where extracted spans are highlighted directly within the original text [13]. This traceability enables users to quickly verify whether extracted information is supported by the underlying document and facilitates both manual inspection and debugging.

Providing direct evidence for extracted information is particularly important in domains such as healthcare and law, where AI-assisted decisions must remain transparent and explainable [11]. Instead of requiring users to trust model outputs blindly, every prediction can be traced back to the exact text snippet from which it originated, supporting human verification and increasing confidence in the extraction process.

In addition to explainability, the pipeline optionally applies an LLM-based filtering stage. During this second pass, each extracted candidate is reconsidered in the context of the original document, after which the language model determines whether the extraction should be retained or discarded. The objective of this verification step is to improve precision by removing hallucinated or weakly supported extractions. However, this comes at the cost of additional computational overhead, since the document must be processed a second time.

Alternative verification strategies are possible. A human-in-the-loop approach would likely provide the highest level of reliability, particularly in safety-critical applications, but substantially reduces scalability and increases annotation effort. Confidence-based thresholding represents another common strategy, where predictions below a predefined confidence score are discarded [9]. However, designing a reliable confidence estimation mechanism is a challenging research problem in itself and would introduce an additional experimental variable. To keep the evaluation focused, we therefore opted not to implement confidence-based thresholding.

The filtering strategy adopted in this work represents a compromise between these alternatives. It introduces additional computational cost through a second language model inference, but remains fully automated while improving precision without requiring manual intervention. The influence of this design choice is evaluated experimentally in the following chapters.

3.4. Capabilities, Trade-Offs and Constraints

The design of the proposed pipeline involves several trade-offs between flexibility, complexity, and practical applicability. Rather than optimizing for a single specific use case, the primary objective was to develop a modular and general framework that enables systematic investigation of different design choices within LM-based information extraction.

One of the main strengths of the proposed pipeline is its modularity. Each component, including document processing, chunking, retrieval, extraction, and post-processing, is implemented as an independent stage. This allows individual components to be replaced or extended without requiring changes to the remainder of the system. For example, alternative chunking strategies, embedding models, retrieval mechanisms, or extraction models can be integrated into the pipeline with minimal modifications. This modularity is particularly valuable for experimental evaluation, as it enables the effects of individual design choices to be studied independently.

Another important design consideration was balancing system complexity with experimental control. More advanced retrieval architectures, such as graph-based or iterative retrieval approaches, may improve performance for certain tasks but also introduce additional components and dependencies. By adopting a traditional retrieval-augmented generation architecture, this work prioritizes interpretability and controlled experimentation over maximum architectural complexity. This makes it possible to analyze the influence of individual factors such as retrieval depth, context ordering, and context size without introducing additional variables.

A further strength of the proposed approach is its generality. Unlike many information extraction systems that are developed for a specific domain or task, the proposed pipeline is designed to be domain-independent. Although the experiments focus on medical and legal datasets, the architecture itself does not rely on domain-specific rules or manually engineered features. Instead, the extraction process is driven by the provided schema and language model capabilities, allowing the same framework to be adapted to different information extraction tasks with minimal changes. Similarly, the pipeline is not tied to a specific language model or retrieval model, making it suitable for evaluating different models as they become available.

Despite these strengths, the proposed pipeline also has several limitations. First, the system operates exclusively on textual information and does not explicitly process visual elements such as tables, figures, images, or complex document layouts. Consequently, information that is only available through visual representations may be unavailable to the extraction model. Extending the pipeline with multimodal document understanding methods could address this limitation in future work.

Second, the quality of the extraction process depends partly on the quality of the initial document parsing step. Although Docling provides structured document extraction capabilities, PDF parsing and OCR processes can introduce errors such as missing text, incorrect character recognition, or incorrect reconstruction of document structure. Since these errors occur before retrieval and extraction, they may propagate through the entire pipeline and negatively affect downstream performance.

Finally, while the modular design allows more advanced retrieval and verification strategies to be incorporated, this work intentionally focuses on a simpler pipeline configuration. More sophisticated approaches, such as adaptive retrieval, confidence-based verification, or human-in-the-loop validation, may further improve extraction reliability but introduce additional complexity and experimental variables.

4

Experimental Setup

4.1. Research Questions

The goal of this paper is to see to what extent we can use performance-enhancing techniques to improve the information extraction capabilities of smaller language models. To do so, we follow a stepwise experimental approach, beginning with the retrieval coverage (RQ1), context scaling and chunk ordering (RQ2) and post-processing (RQ3). The methodology is reflected in the three research questions presented below:

RQ1 **How well can vector retrieval locate target entities in legal and medical documents, and how many retrieved chunks are needed to achieve full recall?** To assess whether a retrieval-augmented generation (RAG) pipeline is viable, we first need to verify that the correct information can be retrieved from the vector database. For this purpose, we conduct a Recall@K experiment, where we check whether the target entity to be extracted appears in at least one of the top-K retrieved chunks. We gradually increase K until recall reaches 100%, meaning that all target entities are covered within the retrieved chunks. This allows us to determine the minimum retrieval depth required to fully capture the relevant information. We perform this experiment on both datasets and compare the results to see how retrieval performance differs between them.

RQ2 **How do context size and chunk ordering in a RAG pipeline affect extraction performance?** Modern models with large context windows can theoretically process entire documents without needing RAG. We want to investigate how splitting a document into smaller, retrieved chunks impacts the extraction performance, compared to just letting it read the whole document at once. To evaluate this, we gradually increase the amount of retrieved context provided to the model, starting from 20% of the document and scaling up to 100%. In addition, we compare two chunk ordering strategies: ordering retrieved chunks by retrieval relevance and ordering them according to their original position within the document. We perform this experiment on both datasets. For the Donor GP Letter dataset, we first use a sample of 20 documents to compare all configurations. We then evaluate the configuration with the highest recall on a larger sample of 80 documents.

RQ3 **To what extent can a post-extraction LLM filtering step improve the quality of extraction?** When models process large amounts of text, they may extract hallucinated or incorrect information, which lowers precision. We therefore investigate whether a secondary automated filtering step can reduce these errors. To do this, we implement a two-stage pipeline in which the initial extractions are passed through a second LLM call designed to filter out unsupported or incorrect entities based on the original input text. Because recall remains particularly important in our setting, the filter is intentionally kept relatively permissive to avoid removing correct extractions. We then compare the extraction performance of the pipeline both with and without this filtering module. We perform this experiment only for the Kleister dataset.

4.2. Evaluation

A key aspect of this study is the evaluation methodology. Due to the open-ended nature of LLM outputs, the presence of OCR-induced errors, and potential discrepancies between extracted values and the manually created ground truth (see Section 4.3), we chose not to rely on strict exact-match comparisons when calculating performance metrics. Instead, we employ an LLM-based evaluation approach, described in Section 4.2.1, which assesses semantic equivalence between extracted values and ground-truth annotations. This allows for a more accurate estimation of the true performance of the extraction pipeline.

4.2.1. LLM-as-a-Judge

We found that strict exact-match evaluation was too brittle for our setting. Because LLM outputs can vary in formatting, OCR errors are common, and there may be small discrepancies between extracted values and the ground truth, correct extractions were often incorrectly counted as false negatives or false positives. In many cases, the model identified the correct entity, but minor differences in formatting led to a mismatch under exact comparison.

For example, the model predicted *"MAP_Pharmaceuticals,_Inc."*, while the expected value was *"Map_Pharmaceuticals_Inc."*. Although differences in capitalization were already accounted for, the presence of a comma caused the strict match to fail, even though the extracted company name was otherwise correct.

To address this issue, we use an LLM-based evaluation method to determine whether the predicted and reference values are semantically equivalent. LLM-as-a-Judge systems are widely used and have been shown to be aligned with human judgment [27]. This approach provides a more robust assessment of performance and also handles cases where values are expressed in different but equivalent forms.

4.2.2. Metrics

We evaluate the performance of the extraction pipeline using three standard metrics: recall, precision, and the F1-score [24]. Recall measures the ability of the pipeline to identify the relevant entities present in the documents. A high recall indicates that few relevant entities are missed. Precision measures the accuracy of the extracted entities by quantifying how many extracted results are correct. A high precision indicates that the pipeline produces few false positives or hallucinated extractions. The F1-score combines precision and recall into a single metric by calculating their harmonic mean, providing an overall measure of extraction performance that balances completeness and accuracy.

For all reported results, we use micro-averaged metrics. In the micro-averaging approach, true positives, false positives, and false negatives are first aggregated across all documents in the dataset before calculating precision, recall, and F1-score. This differs from macro-averaging, where the metrics are computed separately for each document and then averaged. We choose micro-averaging because it reflects the overall extraction performance across the entire dataset and ensures that documents containing a larger number of entities contribute proportionally to the final evaluation.

4.3. Dataset

4.3.1. Kleister NDA Dataset

We use the Kleister NDA dataset [37], which contains complex and realistic samples (Non-Disclosure Agreements) that resemble dense administrative documents commonly found in healthcare and other privacy-sensitive settings. For our experiments, we use the dev-0 split, which consists of 83 samples.

The orginal task is a multi-entity extraction problem, where the goal is to extract the following attributes:

- **Effective date**: the date in `YYYY-MM-DD` format on which the contract becomes legally binding.
- **Jurisdiction**: the state or country governing the contract.
- **Party**: the party or parties involved in the contract.
- **Term**: the duration of the contract as specified in the document.

In some cases, multiple values may be associated with a single attribute; in such cases, all valid values

are extracted and returned. In addition, the documents may contain misleading or irrelevant values that resemble valid entries, which increases the difficulty of the task.

For this study, we focus exclusively on extracting the **party** attribute. This allows for a more direct comparison with the experiments performed on the donor dataset described in Section 4.3.2, where we also perform single-entity extraction. We selected the party attribute because early experiments indicated that it was the most challenging entity type. Unlike dates, jurisdictions, or contract terms, party names are more open-ended in structure and often appear multiple times throughout a document, sometimes in slightly different forms.

The documents range from 1,790 to 44,006 characters in length, with an average length of 16,920 characters and a median of 15,541 characters. This reflects the size of real-world documents and makes the dataset well-suited for testing the limits of smaller language models under long-context and resource-constrained conditions, effectively pushing them toward their operational boundaries.

4.3.2. Donor General Practitioner Letters Dataset

We also use a private real-world medical donor dataset to evaluate the performance of the proposed pipeline in a realistic healthcare setting. The dataset consists of 185 general practitioner referral letters written in Dutch.

The documents are provided as PDF files and are converted to text using Docling. Each letter contains information about multiple medical examinations, together with clinical observations, comments, conclusions, and other medical remarks. The ground truth labels were manually created by reviewing each document and extracting the conclusion written by the medical professional.

For this study, we focus on a specific extraction task: identifying the conclusion of an X-thorax examination, if present in the document. The goal is to extract the medical interpretation associated with the scan. This may correspond to a normal finding, an abnormality or anomaly, or in some cases there may be no mention of an X-thorax examination.

Compared to the Kleister dataset, these documents contain more domain-specific terminology and less structured language reflecting the complexity of real-world clinical communication. This makes the dataset suitable for evaluating how well the pipeline generalises to practical healthcare scenarios.

For the experiments, we use a subset of 20–30 letters from the full dataset. Within this subset, the documents have an average length of 7,565 characters and a median length of 8,280 characters.

Due to computational constraints and the fact that all experiments were executed locally, this dataset is used as a validation setting rather than a full benchmark. As a result, fewer experimental configurations are evaluated compared to the Kleister NDA dataset.

4.4. Models and Parameters

4.4.1. Language Models

The language models used in the experiments are shown in Table 4.1. Three of the models (Qwen 3.5 9B, Gemma 3n E4B, and Meta LLama 3.1 8B) were run using OpenRouter[1] or locally through Ollama[2]. These represent the smaller models in our setup and were selected based on their popularity and accessibility.

The remaining two language models, Gemini 2.5 Flash Lite and Gemini 2.5 Pro, were accessed via the Google AI Studio API[3]. In general, we use an API-based setup for larger models for efficiency reasons, and in this case the Gemini models are also only available through API access. These models were chosen due to their straightforward integration with LangExtract and availability.

All models were evaluated using their default, out-of-the-box parameters to simulate a realistic baseline deployment scenario without task-specific hyperparameter tuning. Furthermore, we deliberately avoided forcing deterministic generation by setting the temperature to 0. Recent literature indicates

[1] https://openrouter.ai/
[2] https://ollama.com/
[3] https://aistudio.google.com

Running Mode	Model	Developer	Params(B)	Context Window(tokens)
Local (Ollama) / OpenRouter	Qwen 3.5 [32, 33]	Alibaba Cloud	9	262K
	Gemma 3n [16, 15]	Google	8	32K
	Llama 3.1 [29]	Meta AI	8	128K
Google AI Studio	Gemini 2.5 Flash Lite [8, 14]	Google	-	1M
	Gemini 2.5 Pro [8]	Google	-	1M

Table 4.1: The models used for the experiments.

Metric	
Recall	0.957
Precision	0.856
F1	0.903

Table 4.2: Performance Benchmark. The extraction performance of Gemini 2.5 Pro, which serves as the state-of-the-art reference point for this study.

that non-zero temperatures can improve extraction recall by allowing the model more flexibility when resolving ambiguous context, albeit at a potential cost to precision [40]. In the context of this study, recall was prioritized over precision, as medical information extraction systems are expected to operate with human oversight and verification. Therefore, we considered it more important for the model to identify as much relevant information as possible, even if this resulted in a small increase in false positives.

4.4.2. Embedder

For the embedding model, we use Qwen 3 Embedding 0.6B [42]. This model was selected because of its strong performance on multilingual retrieval benchmarks, including Dutch [28], which is particularly relevant for our medical donor dataset. The embedder is used to convert document chunks and retrieval queries into dense vector representations, enabling semantic similarity search within the vector database. During retrieval, the most relevant chunks are selected based on the similarity between the query embedding and the stored chunk embeddings. The retrieval performance on our datasets is evaluated in Section 5.1.

4.5. Benchmark

To establish a reference point for the desired extraction performance, we selected Gemini 2.5 Pro [8] as our state-of-the-art benchmark model. We use the Kleister NDA dataset for our benchmark. All 83 samples were passed to the model with instructions to perform the extraction task. We used the prompt shown in Appendix A, which specified the extraction rules and the expected output format. However, the prompt did not include any few-shot examples or example input-output pairs, allowing us to evaluate the model's baseline extraction performance without much additional task-specific guidance.

The reference point is shown in Table 4.2. Gemini 2.5 Pro is able to extract almost all relevant entities, achieving a recall of 0.957. At the same time, it maintains a relatively high precision of 0.856, indicating that most of the extracted entities are correct. Overall, this results in an F1-score of 0.903, showing strong performance across both completeness and accuracy.

5

Results

We present the results of our experiments, the results are grouped by research question.

5.1. Retrieval Coverage and Minimum Retrieval Depth

Figures 5.1 and 5.2 present the Recall@k results for the Qwen3-Embedding-0.6B retriever across two datasets: the Kleister NDA legal dataset (subset of n = 50) and the Donor GP letter dataset (subset of n = 22).

On the Kleister dataset, the retriever achieves a recall of 82.0% at k = 1, meaning that for 41 out of 50 documents, the correct entity is already present in the single highest-ranked chunk. Recall increases to 96.0% at k = 2 (48/50), and reaches 100% at k = 3 (50/50).

On the Donor GP letter dataset, recall at k = 1 is 54.5% (12/22), rising to 81.8% at k = 2 (18/22), and reaching 100% at k = 3 (22/22).

Across both datasets, full recall is achieved at k = 3, indicating that retrieving three chunks is sufficient to surface the target entity in all evaluated documents.

> **RQ1:** We find that a retrieval depth of k=3 is sufficient to reach full recall on both datasets. At this point, all target entities are included in the retrieved chunks, indicating that this is the minimum effective retrieval depth in our setup.

5.2. Context Size and Chunk Ordering Effects

The results are presented separately for both datasets: Kleister NDA, tested across four models, and Donor GP Letter, tested with Llama 3.1 8B. For each dataset we compare two chunk ordering strategies applied to the retrieved context: relevance ordering, in which chunks are ranked by retrieval score with the most relevant presented first; and document ordering, in which retrieved chunks are re-sorted by their original position within the source document to restore narrative continuity.

5.2.1. Kleister NDA

Figures 5.3 and 5.4 present F1, precision, and recall as a function of the percentage of document context provided, for both ordering strategies across all four models. Figures 5.5, 5.6, , 5.7 and 5.8 isolate the F1, precision and recall for each model for both ordering ordering strategies.

Effect of Context Size

The results reveal that models respond to increasing context in markedly different ways, and no single trend holds across all systems.

Gemini 2.5 Flash Lite (Figure 5.5) displays the clearest precision–recall trade-off as context grows. Under relevance ordering, precision falls monotonically from 0.768 at 20% fill to 0.447 under no-RAG,

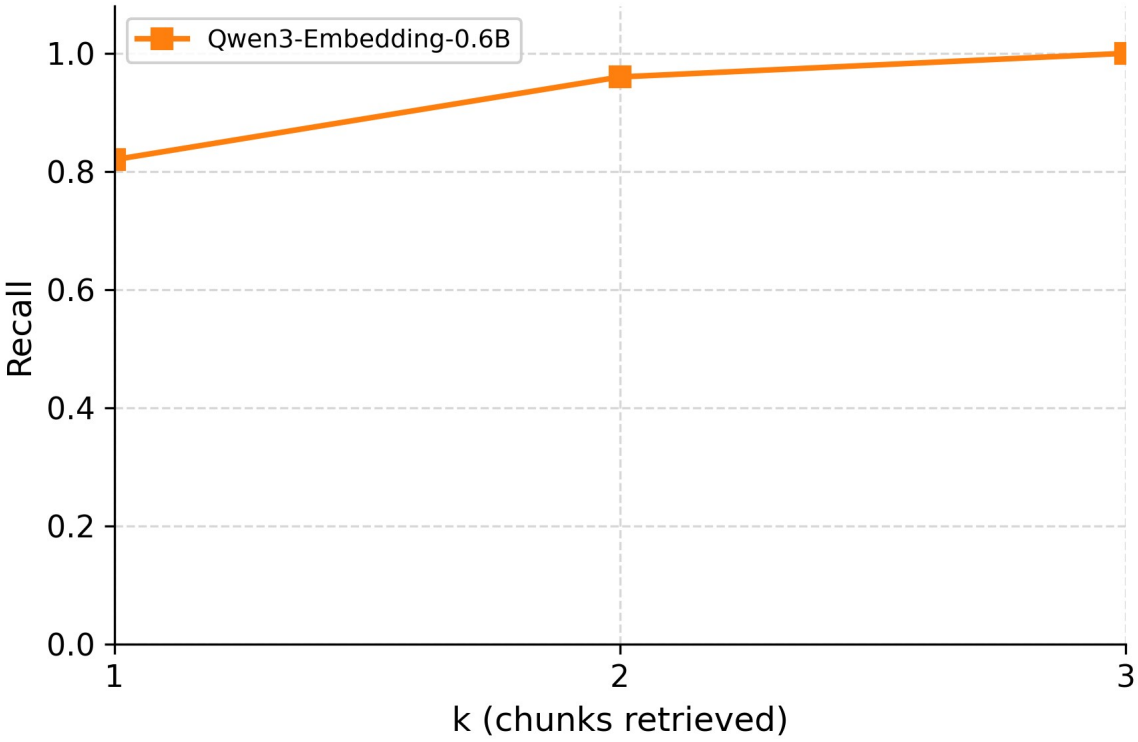


Figure 5.1: Recall@k Kleister

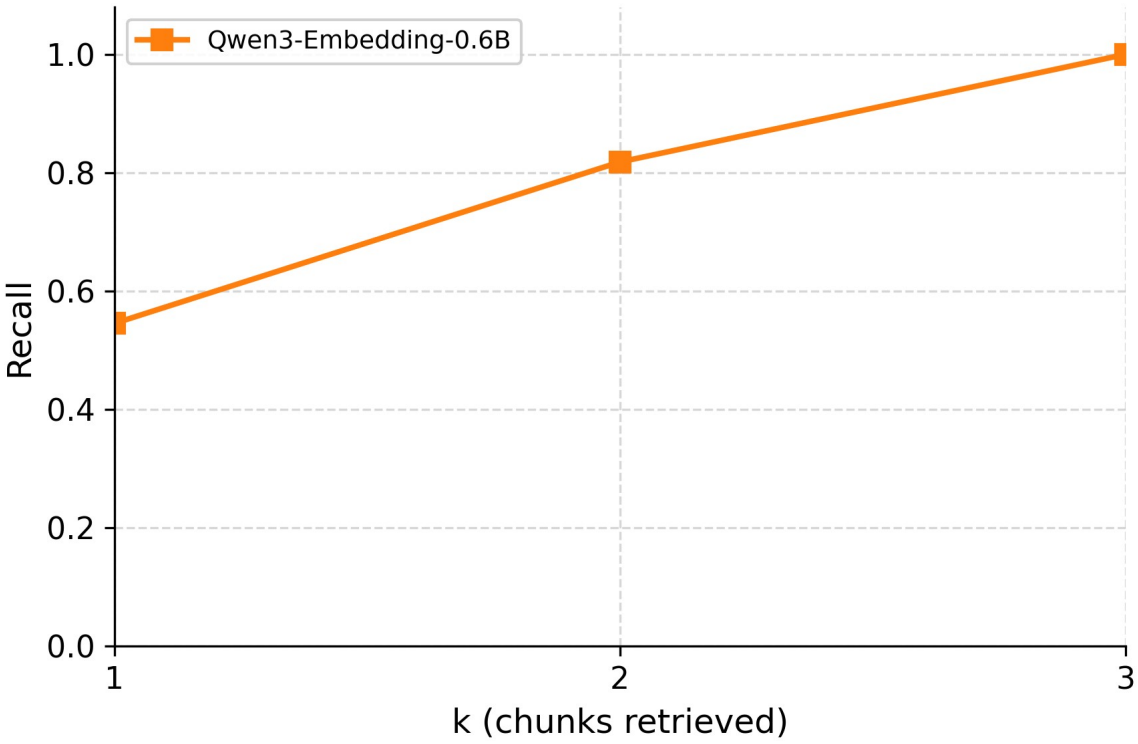


Figure 5.2: Recall@k Donor

while recall climbs from 0.720 to 0.938 over the same range. The F1 score consequently peaks at 40% (0.754) and declines as context volume grows beyond what the model can exploit without becoming over-inclusive. Document ordering produces a similar curve, with F1 values within 0.023 of the relevance-order counterpart at every fill level. This model is therefore largely robust to how chunks are presented, suggesting that context size has a larger effect than chunk ordering for this model.

Qwen3.5 9B (Figure 5.6) shows a consistently conservative extraction pattern across all context settings. Precision remains very high and stable across fill levels, ranging from 0.906 to 0.958 under relevance ordering and from 0.929 to 0.952 under document ordering. In contrast, recall stays persistently low, varying only between 0.308 and 0.437 (relevance ordering) and 0.350 to 0.437 (document ordering), indicating limited sensitivity to additional context.

Overall performance does not improve substantially with increased context size. The best configuration is achieved at 60% context fill using document ordering, where the model reaches an F1-score of 0.597 with a precision of 0.945 and a recall of 0.437. This is only marginally higher than lower-fill configurations, showing that additional context does not meaningfully improve recall or overall extraction quality. This pattern suggests a strongly conservative extraction behaviour, where the model prioritises precision over recall, largely independent of both context size and chunk ordering strategy.

Gemma 3n E4B (Figure 5.8) peaks at 40% fill (F1 = 0.561 under document ordering) and degrades at higher fill levels without recovering under no-RAG (F1 = 0.407). This model appears to benefit from focused, topically dense context but cannot leverage the additional information introduced at higher fill levels. In the no-RAG setting, both precision and recall decrease noticeably compared to the lower fill configurations.

Llama 3.1 8B (Figure 5.7) shows the most irregular performance pattern on the Kleister dataset. In terms of recall, the model performs best at a context fill level of 60% for both chunk ordering strategies. Precision follows a less consistent trend: under relevance-based ordering, performance generally improves as more context is provided, while under document-order ordering precision peaks at 80% context fill. Overall, the strongest F1-score is achieved at 60% context fill using document-order ordering, where the model reaches an F1-score of 0.663.

Effect of Chunk Ordering

Across all models, Gemini 2.5 Flash Lite is the least sensitive to ordering strategy, with absolute F1 differences remaining below 0.023 at every fill level. Qwen3.5 9B shows a small but consistent improvement under document ordering (ΔF1 $\approx$ +0.02–0.05).

Llama 3.1 8B exhibits the strongest and most context-dependent response to ordering. At 60% fill, switching to document ordering raises F1 from 0.554 to 0.663, a gain of +0.109, the largest single positive effect observed across all models and fill levels. This improvement is driven by both a precision gain (+0.161) and a recall gain (+0.021), indicating that when chunks follow reading order, the model extracts more accurately rather than simply becoming more permissive. However, this benefit does not generalise uniformly: at 20% and 80% fill the net F1 change is negligible, and the precision gains at those levels are offset by recall drops, producing a redistribution rather than a net improvement.

At first, document ordering appears beneficial for Gemma 3n E4B, with its strongest performance occurring at 40% context fill where it outperforms relevance ordering. However, this trend reverses as more context is added. At higher fill levels, performance drops more sharply under document ordering, and at 80% fill it falls below the relevance-based configuration. The starkest negative effect occurs at 80% fill, where document ordering reduces F1 from 0.506 to 0.390 (ΔF1 = -0.116), the largest single degradation in the experiment. The drop is consistent across both precision (-0.174) and recall (-0.088), suggesting that at high context levels the model relies on relevance-ranked context to maintain focus, and loses effectiveness when chunks are reordered into narrative sequence.

5.2.2. Donor GP Letter

Figures 5.9 and 5.10 show the F1, precision, and recall for Llama 3.1 8B evaluated on the Donor GP Letter dataset (n=20), comparing relevance ordering (solid lines) with document ordering (dotted lines) across the same five fill levels.

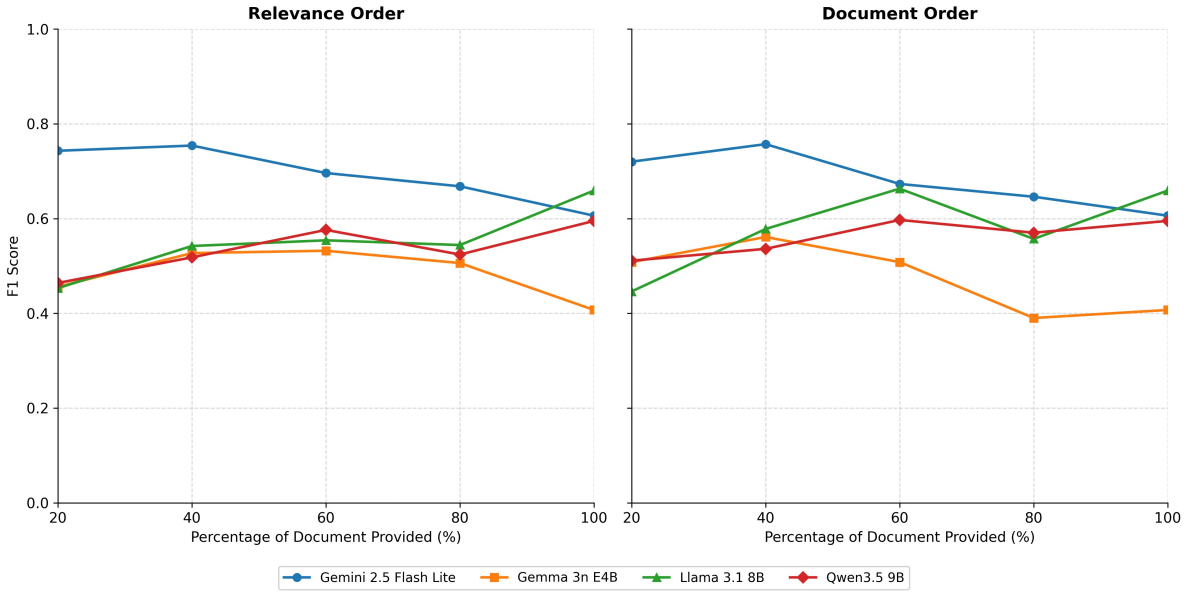


Figure 5.3: F1 Score, Relevance Order vs Document Order. F1 score as a function of the context fill, for both document ordering and relevance ordering. (Kleister NDA)

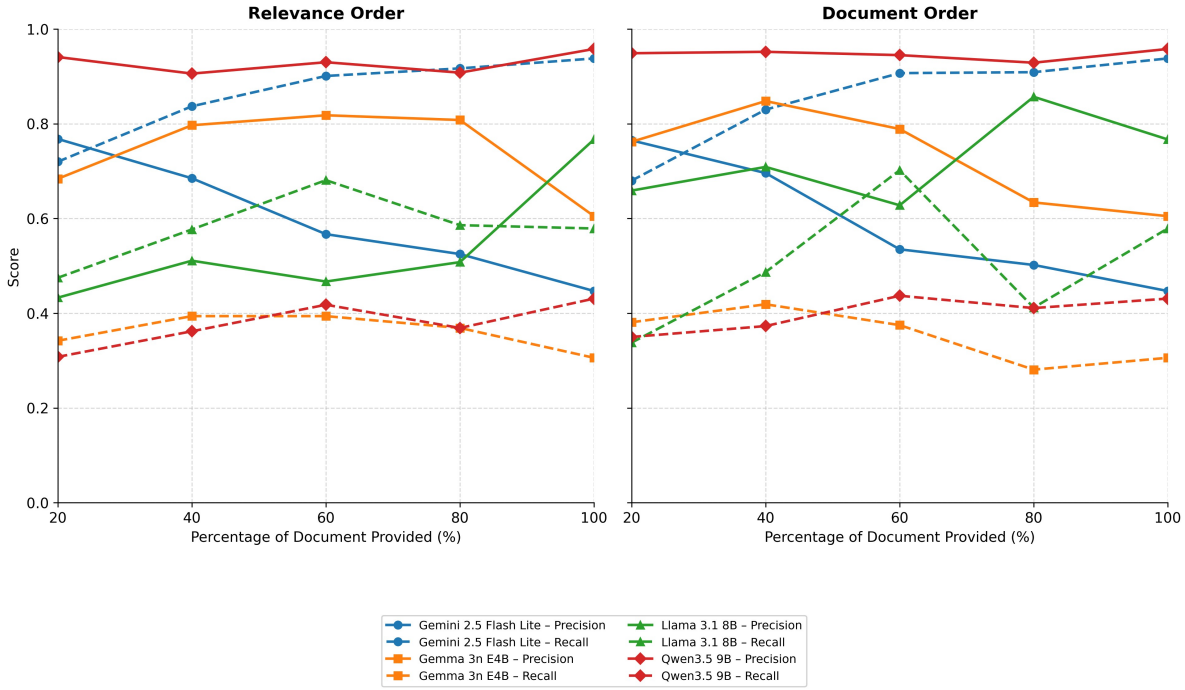


Figure 5.4: Precision and Recall, Relevance Order vs Document Order. Precision and Recall scores as a function of the context fill, for both document ordering and relevance ordering. (Dotted Lines are Recall). (Kleister NDA)

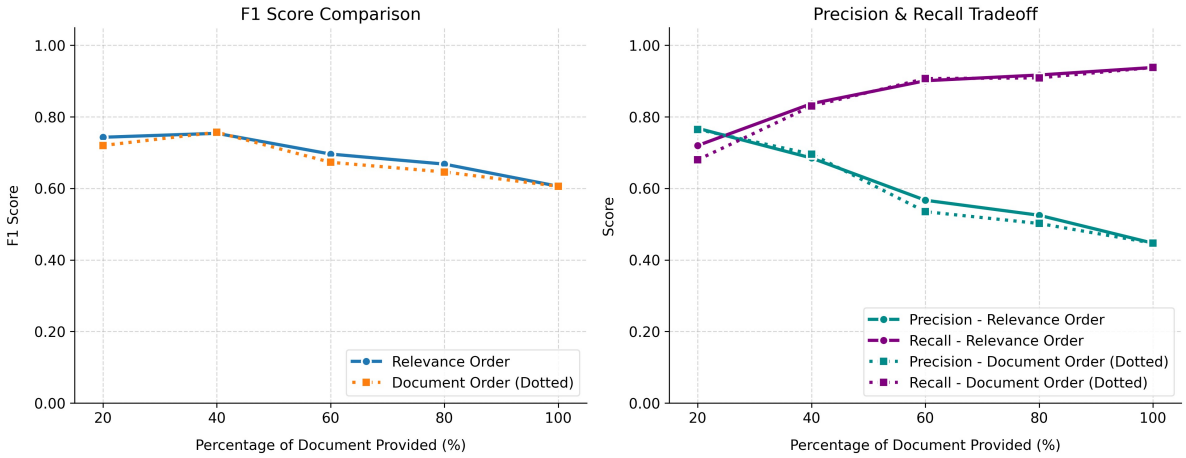


Figure 5.5: Gemini 2.5 Flash Lite, Relevance Order vs Document Order. F1, Precision and Recall scores as a function of the context fill, for both document ordering and relevance ordering. (Kleister NDA)

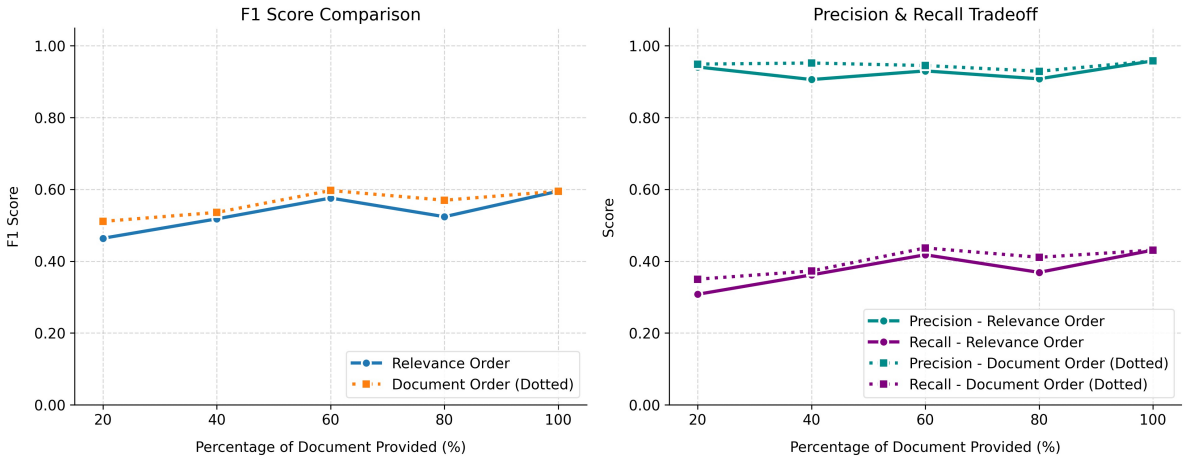


Figure 5.6: Qwen 3.5 9B, Relevance Order vs Document Order. F1, Precision and Recall scores as a function of the context fill, for both document ordering and relevance ordering. (Kleister NDA)

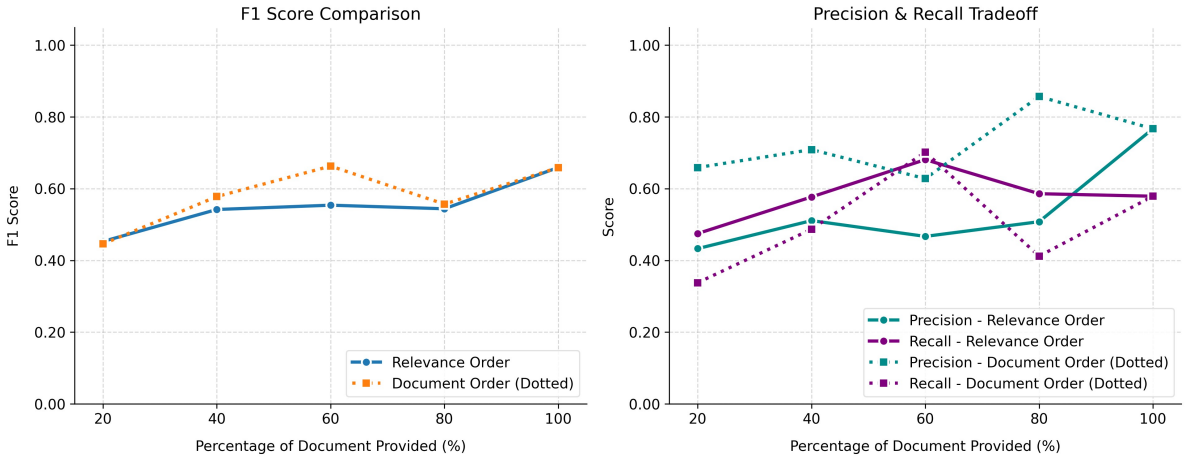


Figure 5.7: Llama 3.1 8B, Relevance Order vs Document Order. F1, Precision and Recall scores as a function of the context fill, for both document ordering and relevance ordering. (Kleister NDA)

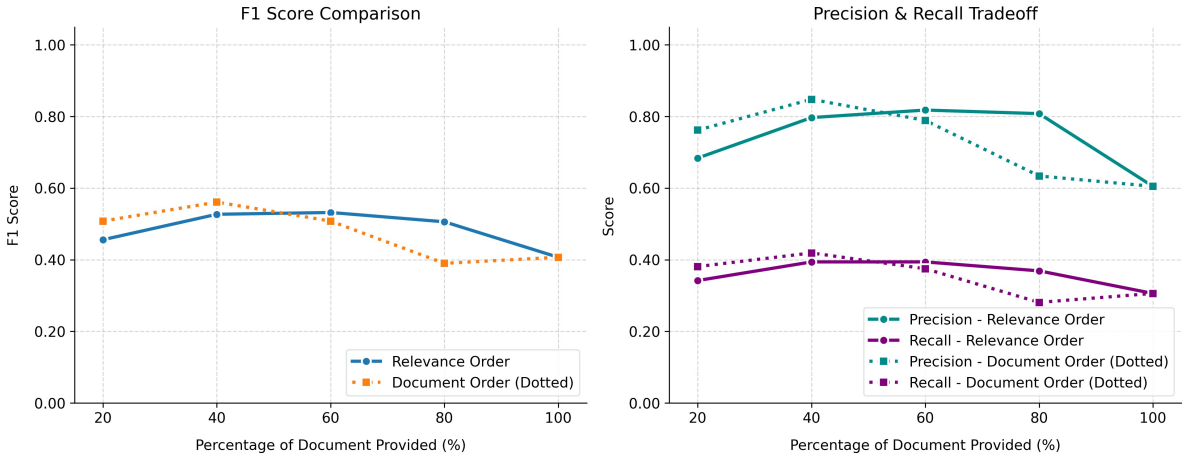


Figure 5.8: Gemma 3n E4B, Relevance Order vs Document Order. F1, Precision and Recall scores as a function of the context fill, for both document ordering and relevance ordering. (Kleister NDA)

Effect of Context Size

Unlike the Kleister results for the same model, both ordering strategies on the Donor GP Letter dataset produce a sharp, consistent F1 improvement from 20% to 40% fill. Under relevance ordering, F1 jumps from 0.417 to 0.765; under document ordering, from 0.417 to 0.727. This steep initial gain is driven almost entirely by recall: at 20% fill recall stands at only 0.333 for both strategies, well below the 0.475 seen on Kleister for the same model and fill level.

Beyond 40%, performance plateaus. Under relevance ordering, F1 stays between 0.698 and 0.765 across the 40–100% range; under document ordering, between 0.667 and 0.727. Recall continues to climb monotonically, reaching 0.938 (relevance) and 0.933 (document order) at 80% fill and remaining at that level under no-RAG. Precision, however, peaks at 40% fill (0.722 and 0.667 respectively) and declines as more context is added, the same trade-off observed on Kleister. The no-RAG condition yields F1 = 0.718 for both strategies, which is competitive with the best RAG result of 0.765 at 40% under relevance ordering.

Effect of Chunk Ordering

On the Donor GP Letter dataset, relevance ordering consistently outperforms document ordering across all RAG fill levels. The gap is most pronounced at 60% fill (ΔF1 = −0.051) and 80% fill (ΔF1 = −0.031). This stands in direct contrast to the Kleister result for the same model, where document ordering produced its largest gain at 60% fill (+0.109 F1). The reversal across datasets points to a dataset-specific interaction.

5.2.3. Overall Performance

Table 5.1 summarises the best-performing configuration for each model and dataset based on F1-score. For the Kleister NDA dataset, each model reaches its best performance at intermediate context levels rather than at full context, confirming that more retrieved text does not necessarily improve extraction quality. The optimal settings vary across models, but most perform best with document ordering and 40–60% context fill.

Gemini 2.5 Flash Lite achieves the strongest overall result among the smaller models, reaching an F1-score of 0.757 at 40% context with document ordering. Qwen3.5 9B and Gemma 3n E4B both prioritise precision over recall, resulting in more conservative extraction behaviour and lower overall F1-scores. Llama 3.1 8B performs best at 60% fill with document ordering, balancing recall and precision more evenly.

On the Donor GP Letter dataset, Llama 3.1 8B reaches its best performance at 40% context fill using relevance ordering, with an F1-score of 0.765. Because recall was prioritized over precision for this task, we additionally evaluated the configuration that achieved the highest recall (80% context fill with relevance ordering) using a larger sample of 80 documents. The results are reported in Table 5.1. This evaluation achieved a precision of 0.600, a recall of 0.958, and an F1-score of 0.738, representing a slight increase in all three metrics compared to the corresponding evaluation on the 20-document sample. Compared to Kleister, the optimal configuration differs, suggesting that dataset characteristics influence both the best context level and ordering strategy.

Finally, the Gemini 2.5 Pro benchmark (evaluated on Kleister) remains the strongest overall system, with an F1-score of 0.903, indicating a clear performance gap between the best RAG-based configurations and a state-of-the-art large model.

RQ2:
Kleister NDA: Models respond differently to increasing context size, no single trend applies across all systems. Gemini 2.5 Flash Lite shows a clear precision–recall trade-off: recall improves with more context, while precision steadily decreases. Qwen3.5 9B maintains very high precision but consistently low recall, largely independent of context size or ordering strategy. Gemma 3n E4B performs best at intermediate context levels (40–60%) and degrades at higher fill levels. Llama 3.1 8B benefits the most from document-order chunking, achieving the largest observed improvement at 60% fill (+0.109 F1).
Donor GP Letter: On the Donor dataset, increasing context from 20% to 40% produces a large performance improvement, mainly driven by higher recall. Beyond 40% fill, F1 performance

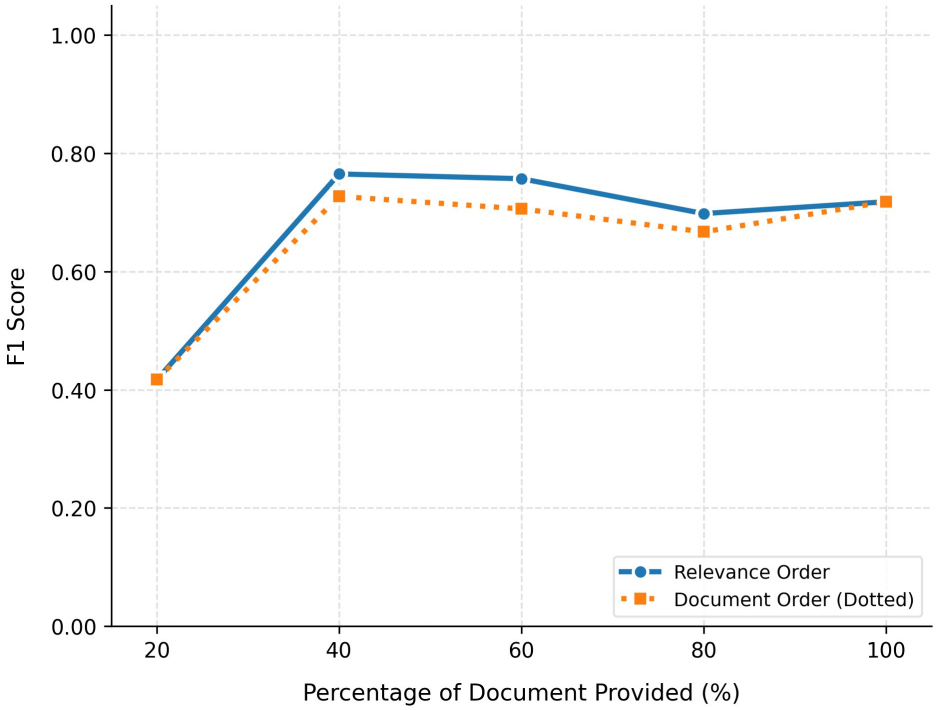


Figure 5.9: F1 Score, Document Order vs. Relevance Order. (Donor GP Letter)

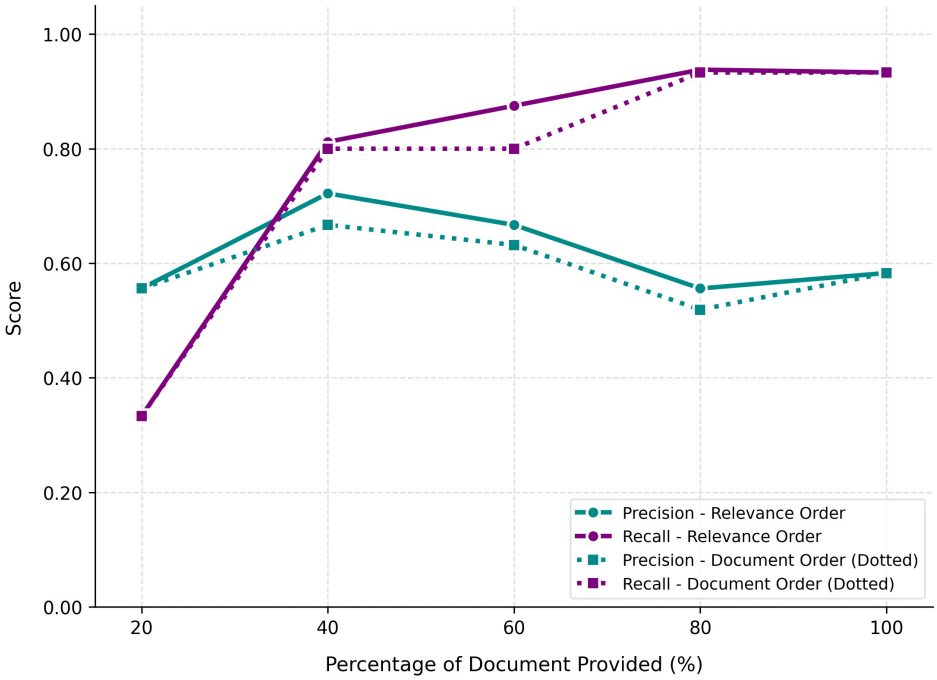


Figure 5.10: Precision and Recall, Document Order vs. Relevance Order. (Donor GP Letter)

Dataset	Model	Best Configuration	Recall	Precision	F1
Kleister NDA	Gemini 2.5 Flash Lite	40% fill, document order	0.830	0.696	0.757
	Qwen3.5 9B	60% fill, document order	0.437	0.945	0.597
	Gemma 3n E4B	40% fill, document order	0.419	0.848	0.561
	Llama 3.1 8B	60% fill, document order	0.702	0.628	0.663
Donor GP Letter (N=20)	Llama 3.1 8B	40% fill, relevance	0.812	0.722	0.765
Donor GP Letter (N=80)	Llama 3.1 8B	80% fill, relevance	0.958	0.600	0.738
Kleister NDA	**Gemini 2.5 Pro**	**Benchmark**	0.957	0.856	0.903

Table 5.1: Best-performing configurations for each model and dataset based on F1-score. Additionally, an extra evaluation on the Donor GP Letter dataset is included using the recall-optimized configuration and a larger sample size (80 documents).

Configuration	Recall	Precision	F1
Base Extraction (Gemini 2.5 Flash Lite)	0.938	0.447	0.606
Extraction + LLM Filter (Gemini 2.5 Flash Lite)	0.938	0.689	0.797
Benchmark (Gemini 2.5 Pro)	0.957	0.856	0.903

Table 5.2: Pipeline Configurations vs Benchmark.

plateaus while precision gradually decreases as more context is added, and recall slowly climbs . In contrast to the Kleister dataset, relevance ordering consistently outperforms document ordering across all RAG configurations. The effect of chunk ordering appears to depend strongly on the dataset and document characteristics.
Overall Performance: Gemini 2.5 Flash Lite represents the strongest RAG-based configuration overall, achieving the highest performance among the retrieval-augmented setups, although it still remains clearly below the Gemini 2.5 Pro benchmark.

5.3. Post-Extraction LLM Filtering

Figure 5.11 and Table 5.2 compare the extraction performance before and after applying the post-extraction LLM filtering step.

The base extraction pipeline using Gemini 2.5 Flash Lite achieves a recall of 0.938, a precision of 0.447, and an F1-score of 0.606. After applying the LLM filter, recall remains unchanged at 0.938, while precision increases substantially from 0.447 to 0.689. This improvement in precision results in a corresponding increase in the F1-score from 0.606 to 0.797.

The results show that the filtering step successfully removes a large portion of incorrect or hallucinated extractions without reducing recall. This is consistent with the intentionally permissive filter configuration, which was designed to preserve as many correct extractions as possible while still reducing false positives.

When compared to the Gemini 2.5 Pro state-of-the-art benchmark, the filtered pipeline remains slightly lower across all metrics. The benchmark achieves a recall of 0.957, a precision of 0.856, and an F1-score of 0.903. Nevertheless, the filtered Gemini 2.5 Flash Lite pipeline substantially narrows the gap relative to the base extraction configuration, particularly in terms of precision and overall F1-score.

RQ3: The post-extraction LLM filtering step substantially improves precision while preserving recall. Applying the filter increases precision from 0.447 to 0.689 and raises the F1-score from 0.606 to 0.797, while recall remains unchanged at 0.938. Although the filtered pipeline still performs below the Gemini 2.5 Pro benchmark, it significantly reduces the performance gap compared to the base extraction pipeline.

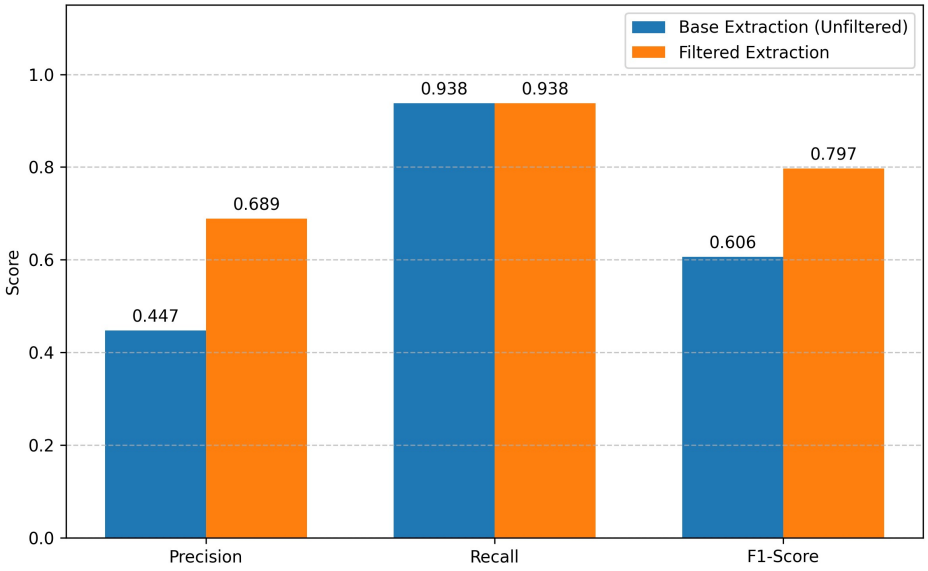


Figure 5.11: Base Extraction vs. Filtered Extraction. The difference in the metrics when applying an LLM Filter after the Base Extraction.

6

Discussion

6.1. Interpretation

Retrieval Coverage and Minimum Retrieval Depth

We observed that full retrieval coverage is achieved at k=3 for both datasets, meaning that all target entities are present within the top three retrieved chunks. This indicates that the retrieval component of the pipeline is generally effective, and that only a small number of chunks is sufficient to surface the relevant information, particularly in the Kleister dataset.

We also observed a clear difference in performance at lower values of k. At k=1 and k=2, recall is substantially higher on Kleister than on the Donor GP Letter dataset. This difference may be partly explained by structural properties of the documents, but we also consider language effects as a contributing factor. The Donor dataset consists of Dutch medical letters, and although Qwen3 Embedding 0.6B performs strongly on multilingual benchmarks [28], embedding models remain largely biased toward English and high-resource languages [1], with Dutch being comparatively underrepresented [5].

Overall, while the retrieval performance is strong and sufficient for full coverage at small k, the lower early-retrieval performance on the Dutch dataset suggests that there is still room for improvement. We see potential gains from using domain-specific or language-adapted embedding models.

Context Size and Chunk Ordering

We find that increasing the amount of retrieved context does not consistently improve extraction performance. Instead, performance is mainly governed by a trade-off between precision and recall. Intermediate context sizes (typically between 40% and 60%) tend to provide the best balance: at lower levels, the model often lacks sufficient evidence for reliable extraction, while at higher levels precision tends to drop as additional retrieved content introduces noise and distracts from relevant information. This suggests that more context is not inherently better, but instead changes the balance between coverage and accuracy.

The effect of chunk ordering is less consistent and depends on both the model and the dataset. In some cases, ordering chunks by their original document position improves performance, while in others relevance-based ordering performs better. Overall, chunk ordering has a smaller impact than context size, but it can still meaningfully shift the precision–recall balance for certain models.

Looking across models, we observe three main behavioural patterns.

First, there are models that show a clear precision–recall trade-off as context increases and are relatively robust to chunk ordering. Gemini 2.5 Flash Lite fits this pattern. For this model, recall improves steadily with more context while precision declines, suggesting that it makes use of additional evidence but becomes less selective as noise increases. Its stable performance across ordering strategies suggests that it can integrate information across long contexts without relying heavily on input structure. We attribute its strong recall performance and stability to its large context window (1M tokens), its reasoning capabilities, and its file-search functionality [14].

Second, there is a more conservative extraction pattern where precision remains high but recall stays low across all settings. Qwen 3.5 9B follows this behaviour. It tends to avoid extracting uncertain entities even when more context is available, which makes its performance largely insensitive to both context size and chunk ordering. We hypothesize that this conservative behavior is due to Qwen 3.5's ”thinking mode”, which generates ”thinking content”, before its response, potentially encouraging more precise and selective extraction behavior [32].

Third, there are models that are more sensitive to retrieval quality and context composition, where performance peaks at intermediate context sizes and degrades when too much or poorly structured context is added. Gemma 3n E4B and Llama 3.1 8B both fall into this category. Gemma 3n E4B benefits from intermediate context sizes but degrades under higher context levels, suggesting sensitivity to noise. This is consistent with its technical documentation, which notes limitations on complex open-ended tasks and handling of language ambiguity [15]. Llama 3.1 8B shows stronger dependence on how context is structured: it benefits from document ordering on Kleister but performs better with relevance ordering on the Donor dataset. This indicates that its performance is closely tied to the coherence of the retrieved context rather than simply its size.

Finally, the differences between datasets are important. On the Donor GP Letter dataset, performance improves sharply even with small increases in context, driven mainly by gains in recall. On Kleister, improvements are more gradual and more sensitive to ordering. This indicates that document structure and task complexity strongly influence how effective retrieval is, and that optimal RAG configurations do not transfer cleanly across datasets. A plausible explanation for these differences is the “lost in the middle” effect, which describes how language models tend to make better use of information at the beginning and end of long contexts than information in the middle [26]. In Kleister, relevant information is more often located earlier in the document, which may explain why document ordering performs well in that setting. In contrast, in the Donor GP Letter dataset, relevant information is more dispersed throughout the document, making relevance-based ordering more effective for bringing key information into more accessible positions in the context.

Post-Extraction Filtering

We find that applying a post-extraction LLM filtering step substantially improves extraction quality, by increasing precision while preserving recall. The filtered pipeline removes a large proportion of incorrect or hallucinated entities without discarding correct ones, resulting in a strong overall gain in F1-score.

This improvement suggests that many of the errors in the base extraction pipeline are not due to missing information, but rather due to over-generation, where the model extracts plausible but unsupported entities. The filtering step is effective because it explicitly re-evaluates extracted entities against the original context, allowing it to reject these unsupported outputs while keeping valid ones.

Importantly, recall remains unchanged after filtering. This indicates that the filter behaves as intended: it is permissive enough not to remove correct extractions, and its main effect is to improve precision rather than trade off coverage. This makes it particularly suitable for settings where missing entities is costly.

Despite these gains, the filtered pipeline still does not reach the performance of the Gemini 2.5 Pro benchmark. This suggests that while post-processing can significantly improve the quality of a weaker model's outputs, it cannot fully compensate for limitations in the base extraction model itself.

Overall, these results show that a simple two-stage architecture, initial extraction followed by lightweight filtering, can be an effective way to reduce hallucinations in RAG-style extraction tasks without sacrificing recall.

6.2. Threats to Validity

Internal Validity

For the Donor GP Letter experiments, a single annotator was responsible for creating the ground truth. However, the annotations were derived directly from the medical conclusions in the source documents, which reduces ambiguity and increases confidence in their correctness.

External Validity

This study evaluates 4 LLMs, which represents only a small subset of the current model landscape. However, the selected models were chosen to reflect a range of architectures and capabilities, which supports diversity in observed behavior. While exact performance values may vary for other models, the general trends observed in terms of context size effects and chunk ordering sensitivity are expected to remain relevant.

Additionally, experiments were conducted on 2 structured document datasets. Although this limits coverage of more diverse domains, the datasets are representative of real-world structured text settings such as legal and medical documents. Therefore, the findings are likely to generalize best to data of similar matter and structure.

Construct Validity

For the Kleister experiments, an LLM-based evaluation method was used to assess extraction correctness. While this enables large-scale evaluation, it may occasionally misclassify correct extractions, particularly in cases where the model refers to semantically correct entities that are not lexically identical to the reference. As a result, the reported scores may not perfectly reflect true performance. However, we consider this trade-off acceptable, as it allowed for a significantly larger experimental scope. Prior work has also shown that LLM-as-a-judge methods can approximate human evaluation reasonably well [27].

6.3. Responsible Research

An important ethical consideration in this study is the use of sensitive data. For this reason, only details related to the Kleister NDA dataset can be publicly released, while the Donor GP Letter dataset cannot be shared due to its confidential nature. All experiments involving the Donor GP Letter dataset were conducted on an isolated virtual machine, and a risk assessment was carried out prior to experimentation to ensure appropriate handling of the data.

In terms of reproducibility, we make all code and experimental configurations publicly available for the Kleister NDA dataset. A key challenge to reproducibility in this area is the rapid pace of development in large language models, which may lead to model deprecation or changes in behavior over time. However, these advancements also enable new capabilities, particularly in agentic and automated information extraction systems, and are likely to drive further research in this direction.

6.4. Use of AI

AI tools were used to help improve the writing by rephrasing and polishing the text, to assist with programming tasks such as fixing bugs and creating experiment orchestrators, and to help identify relevant literature. All ideas, research decisions, and contributions in this thesis are our own. AI was used only as a supporting tool after the ideas had already been developed.

7

Conclusion

This study explored how smaller language models can be improved for information extraction in long, real-world documents, especially in medical and legal settings. The main question was whether techniques like retrieval-augmented generation, better context handling, and post-processing can bring smaller models closer to the performance of larger, state-of-the-art systems.

Overall, the results show that more context is not always better. Instead, there is a clear balance to be found. When too little context is given, models miss important information. When too much context is added, performance often drops because the model is distracted by irrelevant text. The best results are usually found at intermediate context sizes, where the model has enough information without being overloaded.

Retrieval works well in general. In most cases, only a small number of retrieved chunks is needed to include all relevant information. This shows that the retrieval step is effective and not the main limitation of the system. However, performance is slightly weaker in the Dutch medical dataset, which suggests that language and domain differences still matter.

Chunk ordering has a smaller but still noticeable effect. Some models perform better when chunks are kept in their original document order, while others do better when the most relevant chunks are placed first. This means there is no single best strategy, and the best choice depends on both the model and the type of data.

The models themselves behave differently as context increases. Some models gradually improve recall but lose precision. Others stay very precise but fail to extract more information even when more context is available. A third group performs best at medium context sizes and becomes worse when too much or poorly structured context is added. This shows that models react very differently to the same changes in input.

The two datasets also behave differently. One shows strong improvements even with small increases in context, mostly because recall improves quickly. The other improves more slowly and is more sensitive to how the context is ordered. This suggests that document structure and task difficulty have a strong impact on what works best, and that there is no single configuration that works well everywhere.

Finally, the post-extraction filtering step proves to be very effective. It removes many incorrect or hallucinated extractions while keeping almost all correct ones. This leads to a strong improvement in precision and overall F1-score. However, it still does not reach the performance of the strongest benchmark model, which suggests that filtering helps but cannot fully make up for a weaker base model.

In summary, the main takeaway is that better performance does not come from simply adding more context or using more complex pipelines. Instead, it comes from finding the right balance between how much information is given, how it is structured, and how outputs are checked.

This work provides a structured look at how smaller language models can be used for information extraction in complex, real-world documents. The main contribution is not a single best-performing

system, but a clearer understanding of how different design choices affect performance. In that sense, it can serve as a practical starting point for building more reliable extraction pipelines with smaller models. The goal is to offer a baseline that can be extended and improved in future work, rather than a final optimized solution.

At the same time, the experiments show that performance is influenced by many interconnected factors across the entire pipeline. This ranges from early decisions like chunking strategy and embedding quality, to retrieval design and prompt formulation, and finally to post-processing and filtering of model outputs. Each of these steps has a measurable impact on final performance, and improvements in one part often depend on choices made in another. Because of this, future work should not focus on individual components in isolation, but rather on jointly optimizing the full pipeline. For example, better chunking strategies or domain-specific embeddings may improve retrieval quality, while more targeted prompting could reduce noise at the extraction stage. Similarly, more advanced filtering methods or iterative refinement steps could further reduce hallucinations without harming recall.

References

[1] Antonios Anastasopoulos and Graham Neubig. “Should all cross-lingual embeddings speak English?” In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 8658–8679.

[2] Akari Asai et al. “Self-rag: Learning to retrieve, generate, and critique through self-reflection”. In: *International conference on learning representations*. Vol. 2024. 2024, pp. 9112–9141.

[3] Samy Ateia et al. “LLM-Based Information Extraction to Support Scientific Literature Research and Publication Workflows”. In: *International Conference on Theory and Practice of Digital Libraries*. Springer. 2025, pp. 90–99.

[4] Fan Bai et al. “Schema-driven information extraction from heterogeneous tables”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 10252–10273.

[5] Nikolay Banar et al. “Mteb-nl and e5-nl: Embedding benchmark and models for dutch”. In: *Findings of the Association for Computational Linguistics: ACL 2026*. 2026, pp. 24684–24709.

[6] Anjanava Biswas and Wrick Talukdar. “Robustness of structured data extraction from in-plane rotated documents using multi-modal large language models (LLM)”. In: *Journal of Artificial Intelligence Research* (2024).

[7] Laura Chiticariu, Yunyao Li, and Frederick Reiss. “Rule-based information extraction is dead! long live rule-based information extraction systems!” In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 827–832.

[8] Gheorghe Comanici et al. “Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities”. In: *arXiv preprint arXiv:2507.06261* (2025).

[9] Aron Culotta and Andrew McCallum. “Confidence estimation for information extraction”. In: *Proceedings of HLT-NAACL 2004: Short Papers*. 2004, pp. 109–112.

[10] Yufan Dang et al. “Multi-agent collaboration via evolving orchestration”. In: *Advances in neural information processing systems* 38 (2026), pp. 165025–165059.

[11] European Parliament and Council of the European Union. *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)*. Official Journal of the European Union. Accessed: 2026-05-30. 2024. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

[12] Yunfan Gao et al. “Retrieval-augmented generation for large language models: A survey”. In: *arXiv preprint arXiv:2312.10997* (2023).

[13] Akshay Goel. *LangExtract*. Version 1.5.0. May 2026. DOI: 10.5281/zenodo.17015089. URL: https://github.com/google/langextract.

[14] Google AI. *Gemini 2.5 Flash-Lite*. https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-lite. Google AI for Developers documentation. Accessed: 2026-06-29. 2026.

[15] Google DeepMind. *Gemma 3n Model Card*. https://ai.google.dev/gemma/docs/gemma-3n/model_card. Google AI for Developers documentation. Accessed: 2026-06-29. 2025.

[16] Google DeepMind. *Gemma 3n Model Overview*. https://ai.google.dev/gemma/docs/gemma-3n. Google AI for Developers documentation. Accessed: 2026-06-26. 2025.

[17] Zirui Guo et al. “Lightrag: Simple and fast retrieval-augmented generation”. In: *arXiv preprint arXiv:2410.05779* 2.3 (2024).

[18] Haoyu Han et al. “Retrieval-augmented generation with graphs (graphrag)”. In: *arXiv preprint arXiv:2501.00309* (2024).

[19] Adam Jatowt et al. “Deep statistical analysis of OCR errors for effective post-OCR processing”. In: *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE. 2019, pp. 29–38.

[20] Zhengbao Jiang et al. “Active retrieval augmented generation”. In: *Proceedings of the 2023 conference on empirical methods in natural language processing*. 2023, pp. 7969–7992.

[21] Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. “Recent advances in named entity recognition: A comprehensive survey and comparative study”. In: *arXiv preprint arXiv:2401.10825* (2024).

[22] Songsoo Kim et al. “A Multi-Pass Large Language Model Framework for Precise and Efficient Radiology Report Error Detection”. In: *arXiv preprint arXiv:2506.20112* (2025).

[23] Kory Kreimeyer et al. “Natural language processing systems for capturing and standardizing unstructured clinical information: a systematic review”. In: *Journal of biomedical informatics* 73 (2017), pp. 14–29.

[24] David D Lewis. “Evaluating and optimizing autonomous text classification systems”. In: *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. 1995, pp. 246–254.

[25] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[26] Nelson F Liu et al. “Lost in the middle: How language models use long contexts”. In: *Transactions of the association for computational linguistics* 12 (2024), pp. 157–173.

[27] Yang Liu et al. “G-eval: NLG evaluation using gpt-4 with better human alignment”. In: *Proceedings of the 2023 conference on empirical methods in natural language processing*. 2023, pp. 2511–2522.

[28] Massive Text Embedding Benchmark (MTEB) Team. *MTEB Leaderboard*. `https://huggingface.co/spaces/mteb/leaderboard`. Hugging Face Space. Accessed: 2026-06-26. 2026.

[29] Meta AI. *Introducing Llama 3.1: Our Most Capable Models to Date*. Accessed: 2026-06-26. July 2024. URL: `https://ai.meta.com/blog/meta-llama-3-1/`.

[30] Boci Peng et al. “Graph retrieval-augmented generation: A survey”. In: *ACM Transactions on Information Systems* 44.2 (2025), pp. 1–52.

[31] Renyi Qu, Ruixuan Tu, and Forrest Bao. “Is semantic chunking worth the computational cost?” In: *Findings of the Association for Computational Linguistics: NAACL 2025*. 2025, pp. 2155–2177.

[32] Qwen Team. *Qwen3.5-9B*. `https://huggingface.co/Qwen/Qwen3.5-9B`. Hugging Face model card. Accessed: 2026-06-29. 2026.

[33] Qwen Team. *Qwen3.5: Towards Native Multimodal Agents*. Accessed: 2026-06-26. Feb. 2026. URL: `https://qwen.ai/blog?id=qwen3.5`.

[34] Sunita Sarawagi. “Information extraction”. In: *Foundations and trends in databases* 1.3 (2008), pp. 261–377.

[35] FA Senel and Hayri Baytan Ozmen. “A comparative Review of Hallucination Mitigation and Performance Improvement Techniques in Small Language Models”. In: *Journal of Research and Design* 2.1 (2025).

[36] Anubhav Shrimal et al. “PARSE: LLM driven schema optimization for reliable entity extraction”. In: *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*. 2025, pp. 2749–2763.

[37] Tomasz Stanisławek et al. “Kleister: key information extraction datasets involving long documents with complex layouts”. In: *International Conference on Document Analysis and Recognition*. Springer. 2021, pp. 564–579.

[38] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[39] Isabella Catharina Wiest et al. “LLM-AIx: An open source pipeline for Information Extraction from unstructured medical text based on privacy preserving Large Language Models”. In: *MedRxiv* (2024).

[40] Paul Windisch et al. “The impact of temperature on extracting information from clinical trial publications using large language models”. In: *Cureus* 16.12 (2024).

[41] Yang Yang et al. “A survey of information extraction based on deep learning”. In: *Applied Sciences* 12.19 (2022), p. 9691.

[42] Yanzhao Zhang et al. “Qwen3 embedding: Advancing text embedding and reranking through foundation models”. In: *arXiv preprint arXiv:2506.05176* (2025).

[43] Penghao Zhao et al. “Retrieval-augmented generation for ai-generated content: A survey”. In: *Data Science and Engineering* (2026), pp. 1–29.

A

Benchmark Prompt

```
You are a legal document analysis assistant. Read the following Non-Disclosure Agreement (NDA)
    and extract the following field if present in the document:

    - party: The formal legal names of the primary corporate entities or individuals entering into the agreement. Do NOT extract contractual roles (e.g., "Provider", "Disclosing_Party") or the names of the specific people signing on behalf of the companies.

    Rules:
    - Only extract information that is EXPLICITLY stated in the document.
    - Do NOT guess or infer missing fields. If a field is not found, omit it entirely.
    - Replace spaces in values with underscores.
    - For party names, extract the full formal legal name (e.g., "Acme_Corporation" not "Acme").
    - Do NOT extract contractual roles (e.g., "Provider", "Disclosing_Party").

    Return your answer as a JSON object with field names as keys and values as either a single
    string or an array of strings (for party, which can have multiple values).

    Example output format:
    {
        "party": ["Acme_Corporation", "Widget_Industries_LLC"]
    }

    IMPORTANT: Return ONLY the JSON object, no other text or explanation.

    --- DOCUMENT START ---
    %s
    --- DOCUMENT END ---
```