LLM-Orchestrated Iterative Retrieval for Contract Clause Extraction

A Controlled Study on CUAD

Alexandru Preda

TU Delft

LLM-Orchestrated

Iterative Retrieval for Contract Clause Extraction

A Controlled Study on CUAD

by

Alexandru Preda

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, July 1, 2026.

Thesis committee:	Dr. Luciano Cavalcante Siebert,	TU Delft, thesis advisor
	Koray Bingöl,	TU Delft, daily co-supervisor
	Dr.ir. Jie Yang,	TU Delft, examiner

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

TU Delft

Abstract

Legal contract review requires practitioners to find specific provisions in documents that can span hundreds of pages, and errors in this task carry direct legal consequences. Standard retrieve-and-extract pipelines reduce hallucination by grounding language model outputs in retrieved document text, but their single retrieval step cannot follow cross-references, check whether the gathered evidence is complete, or reliably confirm that a clause is absent. Whether having a language model actively direct each retrieval step, rather than passively reading a fixed set of retrieved chunks, improves clause extraction accuracy has not, to the best of our knowledge, been tested. This thesis provides a controlled comparison of retrieve-and-extract and LLM-orchestrated iterative retrieval on the Contract Understanding Atticus Dataset (CUAD), keeping all components except the retrieval architecture constant and evaluating both systems across all 38 clause categories under Exact Match, token-level F1, and ROUGE-L. Iterative retrieval consistently outperforms single-pass retrieval. The main experiment, a 5427-pair run on Gemma4 e4b, a 4-billion-parameter model run locally via Ollama, improves Exact Match from 0.687 to 0.744, Token F1 from 0.341 to 0.505, and ROUGE-L from 0.308 to 0.467, with every gain statistically significant under paired tests. A smaller balanced run on both models confirms this and shows the gain again on the larger closed model GPT-5-mini, which improves from Exact Match 0.711 to 0.800, Token F1 0.314 to 0.538, and ROUGE-L 0.283 to 0.490. The F1 gains substantially exceed the Exact Match gains in every case, indicating that iterative retrieval improves the completeness of extracted clause text, not only presence-or-absence decisions. The comparison isolates the contribution of LLM involvement at the retrieval step in a domain and evaluation setting where, to the best of our knowledge, it has not previously been measured.

Keywords: contract clause extraction, legal natural language processing, iterative retrieval, LLM orchestration, large language models, information extraction, CUAD

Contents

1

Introduction

Legal contract review is a routine but demanding task in legal practice. Lawyers regularly examine large collections of contracts to find specific provisions such as liability clauses, renewal terms, governing law, or termination conditions across documents that can span hundreds of pages and thousands of contracts (Hendrycks et al., 2021; S. H. Wang et al., 2023). Hendrycks et al. (2021) estimate that law firms spend roughly half their working time on contract review, at billing rates that can reach $900 per hour. Recent advances in large language models (LLMs) have made AI-assisted legal document processing increasingly feasible (Siino et al., 2025a; Braun et al., 2025; Ariai et al., 2024).

The main challenge is accuracy. An incorrect or fabricated extraction in legal practice is not only unhelpful, but it can lead a lawyer to miss a material restriction, misstate a party's obligations,and have direct consequences for individual rights and professional decision-making (Ariai et al., 2024; Z. Wang and Yuan, 2025). Dahl et al. (2024) tested four major large language models on verifiable legal questions and found hallucination rates between 58% and 88%. For contract review, this means a model may answer based on what is statistically common in its training data rather than on what the specific document under review actually contains.

Retrieve-and-extract pipelines address this problem by grounding language model outputs in retrieved document text. At query time, a retriever finds the most relevant passages from the indexed contract, and the language model reads those passages to extract the factual claim and supporting text in a single pass. This architecture makes answers verifiable against the source document and has become the dominant approach to improving factual reliability in the legal domain (Siino et al., 2025a; Lai et al., 2024; Pipitone and Alami, 2024). However, the standard retrieve-and-extract design treats retrieval and extraction as independent steps: the retriever returns its top-k results, and the language model reads them passively. For long legal contracts, this is a practical limitation. A termination clause may specify a notice period defined in a separate definitions section. An assignment restriction may depend on change-of-control conditions described under a different article heading. A governing law clause may list exceptions only in an appendix (S. Wang et al., 2025). A retriever returning the most semantically similar passages cannot follow these cross-references or recognise that its initial evidence is incomplete.

A potential solution is to involve the language model at the retrieval step itself, having it direct and evaluate each retrieval call rather than passively reading its output. In this architecture, the language model writes queries, inspects retrieved passages, issues follow-up queries when evidence is insufficient, and decides when the accumulated evidence supports a definitive answer or warrants a null response. Systems of this form have consistently outperformed single-pass retrieval by 10 to 21 percentage points on retrieval recall in general-domain multi-hop question answering (Trivedi et al., 2023; Jiang et al., 2023; Shao et al., 2023; Asai et al., 2024). Agentic systems applied to legal tasks (Watson et al., 2025; Z. Wang and Yuan, 2025) demonstrate that LLM-orchestrated retrieval can work in legal contexts, but evaluate on tasks where the output is a synthesized answer rather than a verbatim span from the document. However, no controlled study has yet tested whether LLM-orchestrated retrieval improves accuracy on extractive contract clause retrieval. Chapter 3 develops the full argument for why this gap has not yet been filled.

This thesis addresses the gap through a single research question.

Research Question: What is the change in clause extraction accuracy when the language model directs the retrieval process, through LLM-orchestrated iterative retrieval, compared to single-pass retrieve-and-

extract on legal contracts?

Clause extraction accuracy here refers to how closely the extracted clause text matches the ground-truth span, measured by Exact Match, token-level F1, and ROUGE-L as defined in Section 2.6.

To answer the question, a controlled experiment is conducted with two contributions. First, it provides a controlled comparison of retrieve-and-extract and LLM-orchestrated iterative retrieval on the Contract Understanding Atticus Dataset (CUAD), with all components except the retrieval architecture held constant so that any difference in accuracy is directly attributable to the retrieval strategy. Second, it measures the contribution of each system under an extractive formulation suited to the legal domain, where the system must return the exact clause text from the document or report that no such clause exists. This setting is important because the EU AI Act (European Parliament and Council of the European Union, 2024) classifies AI systems used for legal interpretation as high-risk and requires them to support human judgment rather than replace it.

The thesis is structured as follows. Chapter 2 provides background on the clause extraction task, key terminology, evaluation metrics, and the limitations of single-pass retrieval. Chapter 3 reviews related work on iterative retrieval and legal question answering, and identifies the precise research gap. Chapter 5 describes the two retrieval conditions under comparison and the dataset. Chapter 6 describes the experimental setup. Chapters 7 and 8 present and discuss the results. Chapter 10 outlines directions for future work.

2

Background

This chapter provides the background needed to follow the rest of the thesis. It defines the contract clause extraction task and explains why it is hard, reviews how large language models are used in legal text and where they fail, introduces retrieval and the move from single-pass to iterative retrieval, and sets out the evaluation metrics used in the later chapters.

2.1. Contract Clause Extraction

Contract clause extraction is a legal information extraction task that asks a system to identify whether a specific type of provision appears in a contract and, when it does, to locate the relevant text within the document. The task arises in practice because practitioners must regularly check contracts for the presence of specific terms such as liability caps, governing law provisions, notice requirements, and renewal conditions in documents that can span hundreds of pages (Hendrycks et al., 2021; Lai et al., 2024).

Hendrycks et al. (2021) introduced the task as a reading comprehension problem: given a contract and a clause category, a large language model (LLM) must return the exact supporting text span from the document, or indicate that the clause is absent. Braun et al. (2025) assess LLM performance in terms of identifying and citing relevant contractual clauses, treating detection and extraction as a single unit. Topollai et al. (2025) refer to the same capability as semantic clause retrieval, where a retriever locates candidate passages and a model determines which constitute the target clause. Across these formulations, clause extraction requires two things: determining whether a clause of the specified type exists in the contract, and if so, returning the verbatim text that constitutes it.

The task is factual rather than interpretive. The system receives a question about a specific clause type and responds with the text spans from the contract that are relevant to answering it, rather than interpreting those spans or drawing legal conclusions from them. Ariai et al. (2024) distinguish this from contract analysis, the downstream step of advising on the legal and business implications of clauses for a specific client, which requires legal judgment and knowledge of applicable law. Clause extraction is the first step: reading the contract to find what it literally says about a given provision. The goal is to help a lawyer locate the relevant text faster, not to substitute for the professional judgment that follows.

2.2. Why Clause Extraction Is Hard

Several properties of legal contracts make clause extraction substantially harder than comparable information retrieval tasks on general documents.

Document length. Commercial contracts regularly span fifty to over one hundred pages (Hendrycks et al., 2021). The clause relevant to a given category may appear anywhere in the document, and a retrieval system must find it without prior knowledge of its location.

Cross-referencing. Legal contracts organize obligations across named articles, schedules, and appendices, and frequently define terms in one place that govern clauses in another. Retrieving the termination article without the relevant definition, or the definition without the clause it applies to, returns incomplete evidence for either (Siino et al., 2025b). A termination clause may specify a notice period only by reference to a definitions section at the start of the document. An assignment restriction may depend on change-of-

control conditions described under a different article heading. A governing law clause may list jurisdictional exceptions only in an appendix.

Absence as the expected outcome. In the CUAD benchmark, approximately 70–80% of clause category and contract pairs have no matching clause (Hendrycks et al., 2021). A system that always retrieves and extracts something would be wrong on the majority of instances. Correctly determining that a clause is absent requires the system to have searched thoroughly enough to be confident in the null conclusion, not merely to have failed to find anything.

Domain-specific vocabulary and formatting. Legal contracts use precise terminology that differs from general text. Defined terms carry exact meanings established earlier in the document, and similar-sounding provisions may have substantively different legal effects (Ariai et al., 2024; Siino et al., 2025b). How the contract text is structured also matters: Braun et al. (2025) find a drop of approximately thirty percentage points in clause extraction accuracy when input text is poorly structured compared to well-formatted input.

2.3. Large Language Models in Legal NLP

Large language models are neural networks trained on large text corpora to predict and generate text. They acquire broad knowledge of vocabulary, syntax, and domain-specific patterns from this training, and can be prompted to perform a wide range of tasks without task-specific fine-tuning.

2.3.1. Capabilities

In legal NLP, LLMs have demonstrated strong performance on tasks where the relevant text is provided as context. LegalBench, a benchmark of 162 legal reasoning tasks, shows that models can classify contract clauses, identify relevant provisions, and answer legal questions with reasonable accuracy when the input text is given (Guha et al., 2023). Surveys of LLM applications in law confirm that these capabilities extend across contract review, statutory interpretation, and legal question answering (Lai et al., 2024; Siino et al., 2025a). General-purpose models without legal fine-tuning are competitive with fine-tuned alternatives on many tasks, particularly when the task requires reading and reasoning over a provided document rather than recalling legal rules from memory (Guha et al., 2023; Lai et al., 2024).

2.3.2. Hallucination and Its Limits

The same models that reason well over provided text also produce factually incorrect outputs when they lack the right context. Dahl et al. (2024) tested four large language models on verifiable legal questions and found hallucination rates between 58% for GPT-4 and 88% for Llama 2. For contract extraction, this means a model may answer based on what is statistically common in contracts of that type rather than on what the specific document under review actually contains. Siino et al. (2025b) document the same failure patterns in practice: overlooked details, factual inconsistencies, and misinterpretation of legal language.

Xu et al. (2025) show formally that hallucination cannot be fixed by better training alone: no model trained on a finite corpus can represent all computable functions, so there will always be inputs for which the model's stored weights produce wrong answers. Hallucination is an inherent property of the architecture, not a correctable training artefact.

The inevitability of hallucination carries regulatory implications. The EU AI Act (European Parliament and Council of the European Union, 2024) classifies AI systems used in legal contexts by their risk level. AI systems that assist judicial authorities are listed in Annex III as high risk. Contract review tools used in private legal practice sit in the limited risk tier, as they do not make binding decisions and their outputs are reviewed by a qualified professional before any action is taken. Regardless of tier, the Act requires that AI systems in legal contexts be transparent enough for human professionals to meaningfully verify their outputs (Article 14 of the EU AI Act (European Parliament and Council of the European Union, 2024), which requires that users can understand, oversee, and if needed correct AI system outputs). A system that answers from parametric memory, without grounding its response in the specific document under review, gives the professional no text to check against. Grounding model outputs in retrieved document text is therefore a prerequisite for satisfying this requirement, not an optional improvement.

2.4. Retrieval-Augmented Generation

Retrieval-augmented generation (RAG), introduced by Lewis et al. (2020), is an architecture in which a language model conditions its output on documents retrieved from an external source rather than on knowledge

encoded in its weights. A retriever selects the most relevant passages from an indexed corpus at query time, and the language model uses those passages as context for its response. The key characteristic of RAG is that the output is generated text produced by the model based on retrieved evidence.

For contract clause extraction, the pipeline used is more precisely called **retrieve-and-extract**. It differs from RAG in one key respect: the output is not generated text, but a factual claim and a verbatim text span extracted from the retrieved passages. The model reads the retrieved passages in a single forward pass and identifies what the contract says about the target clause category, or returns null if the evidence does not support a positive claim. The language model plays no role in directing which passages are retrieved. Dense passage retrieval enables the retriever to match queries to passages by semantic similarity rather than keyword overlap, which is important for legal text where the same concept may be expressed in varied terminology (Karpukhin et al., 2020).

In the legal domain, retrieval-grounded approaches have become the dominant paradigm for improving factual reliability (Siino et al., 2025a; Lai et al., 2024). LegalBench-RAG confirms that retrieval quality, not generation quality, is the main bottleneck: low retrieval precision and recall scores across all evaluated systems indicate that the retrieval step is the primary source of failure in legal RAG pipelines (Pipitone and Alami, 2024).

A retrieval-first design also enables the auditing that the EU AI Act requires. Article 12 of the Act requires that high-risk systems log sufficient information about their operation to allow post-hoc auditing. A system that records which passages were retrieved and presents them alongside its answer provides an auditable trail. Any legal professional can read the retrieved text without specialist knowledge of how the model works internally, which is a practical form of transparency that the Act is designed to protect.

A retrieve-and-extract pipeline with a single retrieval step has no way to correct an incomplete or wrong first pass. If the relevant clause is not among the top-k returned passages, the language model will either produce a false positive from unrelated text or fail to locate the correct supporting span. For long legal contracts, this happens regularly in the CUAD dataset (Hendrycks et al., 2021).

For example, a termination clause may specify a notice period that is defined in a separate definitions section at the start of the document. Querying for “termination notice period” may retrieve the termination article without the definitions section, or vice versa. An assignment restriction may depend on change-of-control conditions described under a different heading, and a governing law clause may list jurisdictional exceptions only in an appendix. Siino et al. (2025b) observe that for long legal documents “performance might suffer greatly when essential information is moved within long context,” and CUAD contracts can exceed one hundred pages (Hendrycks et al., 2021).

Using vector-based semantic search (Karpukhin et al., 2020) narrows the gap between a query and the relevant text by matching on meaning rather than exact keywords, but it does not solve this problem. A single query still produces a single ranked list with no way to check whether the retrieved passages are complete or to ask follow-up questions. Considering these aspects, a language model that inspects what has been retrieved, judges whether the evidence is enough to give a confident answer, and issues additional queries when it is not has the potential to cope with this limitation.

2.5. Iterative Retrieval

Iterative retrieval extends single-pass retrieval by allowing the system to issue multiple retrieval calls, each informed by what has been found so far. The core motivation is that a single query is often insufficient when the information needed spans different parts of a document or depends on intermediate findings (Trivedi et al., 2023; Shao et al., 2023).

Early iterative approaches follow a predetermined pattern. IRCoT (Trivedi et al., 2023) builds on chain-of-thought reasoning, in which a language model generates a sequence of intermediate reasoning sentences before producing a final answer. After each reasoning sentence, IRCoT uses that sentence as a retrieval query and adds the returned passages to the model’s context before the next reasoning step. The process continues until the model generates a sentence that constitutes a final answer, or until a maximum number of steps is reached. Iter-RetGen (Shao et al., 2023) takes a different approach: the model first generates a draft response to the question, then uses that draft as the query for a new retrieval round, and produces an improved response from the newly retrieved passages. The number of rounds is set in advance as a hyperparameter. Both approaches outperform single-pass retrieval on multi-step question answering, but in neither case does the model itself decide whether the retrieved evidence is sufficient to stop: in IRCoT the process ends when the reasoning chain reaches an answer, and in Iter-RetGen it ends after a pre-set number of rounds.

The ReAct framework (Yao et al., 2023) gives the language model full control over this process. Rather than following a fixed schedule, the model interleaves reasoning traces with retrieval actions in a continuous loop: it generates a thought that interprets current evidence, issues a search, observes the result, and decides at each step whether to continue searching or to stop. The model's freedom to decide when to stop searching is what distinguishes ReAct from fixed-pipeline approaches such as IRCoT or Iter-RetGen.

Legal contracts are particularly suited to this kind of retrieval. Defined terms are often placed far from the clauses that invoke them, obligations are spread across articles, schedules, and appendices, and a single question may require evidence from several non-adjacent sections. How much retrieval a question requires therefore varies by contract and clause type and cannot be determined in advance, which is precisely the variability that a model-driven stopping criterion is designed to handle.

This thesis adopts the ReAct paradigm under the label LLM-orchestrated iterative retrieval, in which the language model drives the search process, evaluates returned passages, and determines when the accumulated evidence is sufficient to produce a final answer.

2.6. Evaluation Metrics

Choosing the right metrics for this task is not straightforward. Siino et al. (2025b) survey evaluation approaches across legal AI systems and identify three families: human evaluation, automated metrics (precision, recall, F1, BLEU, cosine similarity), and LLM-based validation. Each family involves trade-offs that depend on what the task requires, and several common choices are unsuitable for verbatim clause extraction.

Human evaluation is flexible and can capture errors that automated metrics miss, but it is slow, expensive, and difficult to reproduce across studies (Siino et al., 2025b). Evaluating 1000 pairs across two systems manually is not practical, and the comparison would not be independently verifiable.

Retrieval metrics such as Recall@k, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) measure whether the correct passage appears in the system's ranked list of retrieved chunks. They evaluate the retrieval step in isolation and say nothing about whether the system drew the correct factual conclusion or extracted the right span from the returned passage. Pipitone and Alami (2024) confirm this gap directly: retrieval-only metrics such as Recall@k evaluate the retrieval step in isolation and say nothing about whether the system drew the correct factual conclusion from the retrieved passages. Since the evaluation in this thesis covers the full pipeline from question to structured answer, a retrieval-only metric would miss half the problem.

BLEU (Papineni et al., 2002) was designed to evaluate machine translation by measuring n-gram overlap between a generated output and one or more human reference translations. For verbatim clause extraction it has two problems. First, BLEU does not distinguish between a span copied directly from the contract and a paraphrase of similar content; a system that paraphrases the clause rather than quoting it can receive a high BLEU score even though the output is not grounded in the document text. Second, BLEU degrades unpredictably when there is only a single reference per instance (Papineni et al., 2002), which is the case in CUAD.

Embedding-based similarity metrics such as cosine similarity and BERTScore measure how semantically close the predicted text is to the reference, rather than whether the words match. Embedding-based similarity is too lenient for legal extraction (Siino et al., 2025b). Consider a contract that specifies Delaware as its governing law and a system that returns New York instead. The prediction is factually wrong and would mislead any lawyer relying on it, yet an embedding-based metric would score it as nearly correct because New York and Delaware are both US jurisdictions and their vector representations are close. For legal clause extraction, what matters is whether the extracted text matches what the contract actually says, not how close the prediction is in meaning to the correct answer.

LLM-based evaluation uses a language model to judge whether a system's output is correct. It is flexible but introduces a dependency on the judging model's own biases and limitations, produces results that vary across model versions, and is expensive at scale (Siino et al., 2025b). For a controlled comparison designed to isolate a single architectural variable, a deterministic metric that any researcher can compute identically is preferable.

Exact Match (EM) is computed on the factual claim field across all instances. A prediction is counted as jointly correct: for present instances, the model must return True with at least one supporting span; for absent instances, it must return False with no span. A model that identifies a clause as present but provides no supporting text, or one that says a clause is absent but still returns spans, is scored as wrong. Over a set of

N instances, EM is defined as:

$$\text{EM} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\hat{a}_i = a_i\right] \tag{2.1}$$

where a_i is the ground-truth answer for instance i, $\hat{a}_i$ is the predicted answer, and $\mathbf{1}[\cdot]$ is the indicator function.

Token-level F1 is computed on the span field, restricted to positive instances where the ground-truth annotation contains a clause text span (Rajpurkar et al., 2016). F1 does not apply to absent-clause instances because comparing two null values gives no signal about retrieval quality. When the ground truth contains a clause but the system returns null, F1 is set to zero so the retrieval failure is counted as a penalty. All spans are normalised (lowercased, punctuation removed, whitespace collapsed) before comparison. Multiple spans on each side are concatenated into one token sequence. Let TP denote the number of tokens appearing in both, $|\hat{s}|$ the total predicted tokens, and $|s|$ the total ground-truth tokens:

$$P = \frac{\text{TP}}{|\hat{s}|}, \qquad R = \frac{\text{TP}}{|s|}, \qquad \text{F1} = \frac{2 \cdot P \cdot R}{P + R} \tag{2.2}$$

Token-level F1 rewards vocabulary overlap and gives partial credit for spans that are close but not exact. It does not penalise predictions that contain the right words in the wrong order, which is a known weakness for verbatim extraction tasks.

ROUGE-L addresses the insensitivity to word order by measuring the Longest Common Subsequence (LCS) between the predicted and ground-truth token sequences, rather than unordered token overlap (C.-Y. Lin, 2004). A prediction that contains the correct words in the wrong order will score lower on ROUGE-L than on token-level F1, because the LCS captures only the longest ordered subsequence shared by both sides. Multiple spans on each side are concatenated before computing the LCS. Let $\text{LCS}(X, Y)$ denote the length of the longest common subsequence between token sequences X and Y:

$$P_{\text{RL}} = \frac{\text{LCS}(\hat{s}, s)}{|\hat{s}|}, \qquad R_{\text{RL}} = \frac{\text{LCS}(\hat{s}, s)}{|s|}, \qquad \text{ROUGE-L} = \frac{2 \cdot P_{\text{RL}} \cdot R_{\text{RL}}}{P_{\text{RL}} + R_{\text{RL}}} \tag{2.3}$$

Token-level F1 and ROUGE-L are complementary. If a system scores high on F1 but noticeably lower on ROUGE-L, it suggests the retrieved text contains the right vocabulary but not in the correct order, which for verbatim legal extraction is a signal of fragmented or paraphrased spans rather than clean verbatim copies. When both metrics move together, it confirms that differences between conditions are not an artefact of the bag-of-words assumption in token-level F1.

2.7. Statistical Significance Testing

A score measured on a sample of questions is only an estimate of how a method would behave on the whole population of questions. A difference between two methods on a sample might be real, or it might be an accident of which questions happened to be drawn. A significance test asks how likely a difference of the observed size would be if the two methods were in truth equally good. A small probability, the p-value, means that chance alone is an unlikely explanation. A confidence interval reports the same evidence as a range, giving the plausible sizes of the true difference rather than only whether one exists.

Independent and paired designs. The right test depends on how the two sets of measurements relate to each other. When the two groups are made of different items, the measurements are independent. When both methods are measured on the same items, each item yields a matched pair of scores and the design is paired. Paired designs are more sensitive, because they remove the variation between items and look only at the within-item difference. Applying a test built for independent groups to paired data is a mistake, because it ignores the pairing and overstates the uncertainty.

Tests for independent groups: z-test and t-test. Two standard tests assume independent groups. The **two-proportion z-test** compares two proportions, such as two accuracy rates, by referring the standardised difference to the normal distribution, written z. It suits yes-or-no outcomes that are summarised as a rate. The **independent two-sample t-test** compares the means of two groups of continuous measurements and refers the result to Student's t distribution, which is the normal distribution widened to allow for the standard deviation being estimated from the sample rather than known in advance. It suits continuous scores. The difference between them is the type of outcome: the z-test is for proportions, the t-test is for means of

continuous values. Both assume that the two groups are independent, so neither is appropriate when the same questions are answered by both methods.

McNemar's test. For paired yes-or-no outcomes, McNemar's test (McNemar, 1947) is the standard choice and the paired counterpart of the two-proportion z-test. It looks only at the items where the two methods disagree, since items that both get right, or both get wrong, carry no information about which method is better. It then tests whether the two kinds of disagreement, those where the first method wins and those where the second wins, occur equally often. The exact binomial form of the test can be used instead of the chi-square approximation, which keeps it valid even when the number of disagreements is small.

Wilcoxon signed-rank test. For paired continuous scores, the Wilcoxon signed-rank test (Wilcoxon, 1945) is the paired counterpart of the t-test for cases where the differences are not normally distributed. It works on the ranks of the per-item differences rather than on their raw values, so a few extreme differences do not dominate, and it makes no assumption that the differences follow a normal distribution.

Bootstrap confidence intervals. A confidence interval can be built without assuming any particular distribution by resampling the observed data with replacement many times and recomputing the statistic on each resample (Efron, 1979). The spread of the recomputed values gives the interval. Resampling whole contracts rather than individual pairs keeps clauses from the same contract together, which respects the fact that they are not independent of one another.

3

Related Work

This chapter places the thesis between two bodies of work that have grown apart. It first reviews iterative retrieval in general domains, then retrieval for legal clause extraction, and then the few legal systems that let the model direct retrieval. It closes by stating the research gap that the thesis addresses.

3.1. Iterative Retrieval: Evidence from General Domains

A single query is often not enough when the answer needs evidence from more than one place. Several peer-reviewed systems show that letting the language model run more than one retrieval round beats single-pass retrieval on multi-hop tasks. They were all tested on general-domain Wikipedia benchmarks. What separates them, and separates them from the system studied here, is how the next query is formed and who decides to make it.

The clearest pattern is that the query is rarely a free choice. IRCoT (Trivedi et al., 2023) turns each new reasoning sentence into the next search query, so retrieval is automatic after every step. On four multi-hop benchmarks (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Ferguson et al., 2020) it raised retrieval recall by up to 21 points and answer accuracy by up to 15 points over single-pass retrieval. FLARE (Jiang et al., 2023) instead triggers retrieval by a confidence threshold: when the model is unsure of the next sentence, that sentence becomes the query (Geva et al., 2021; Stelmakh et al., 2022; Hayashi et al., 2021). Iter-RetGen (Shao et al., 2023) reuses the model's whole previous answer as the next query, for a fixed number of rounds set in advance (Press et al., 2023). In all three the model cannot decide on its own whether to retrieve, how to word the query, or when the evidence is enough.

Self-RAG (Asai et al., 2024) comes closest to a real decision. It trains a model to emit special markers that say whether to retrieve and whether the answer is supported, which cut hallucination on several QA sets (Joshi et al., 2017; Mallen et al., 2023; Clark et al., 2018; Stelmakh et al., 2022). Even so, the query itself is not rewritten freely, and the method needs a model trained for this exact signal rather than working through prompting alone.

Two recent works push toward more model control. Interact-RAG, in a preprint (Hui et al., 2025), lets the model filter, expand, and re-rank passages instead of only issuing queries, reaching a 22.5 percent relative Exact Match gain over the next-best method. Y. Lin et al. (2025), also in a preprint, carry the comparison to regulatory documents and report that 48 percent of single-pass failures happen because the right chunk is never retrieved at all, which is direct support for iterative search in document-heavy legal corpora.

The takeaway for this thesis is that more model control over retrieval consistently beats fixed single-pass pipelines, by roughly 10 to 21 points on recall and 8 to 15 points on accuracy when the answer spans more than one place. The system here is built on the ReAct framework (Yao et al., 2023) from Section 2.5. ReAct is chosen over fixed-pipeline approaches because it gives the language model three capabilities absent from all four systems above: it can decompose retrieval goals and track progress across steps, reformulate queries when initial searches return uninformative results, and decide when accumulated evidence is sufficient to produce a final answer (Yao et al., 2023). It can also conclude that a clause is absent and return null, a decision none of the four systems was designed to make. These properties directly address the cross-referencing structure of legal contracts described in Section 2.2, where a single question may require evidence from a definitions section, a separate article, and an appendix that no fixed sequence of queries can anticipate in

advance. None of the systems above was originally tested in the legal domain, and none was built for a task where the correct answer is often that no clause exists, yet the capabilities they introduce, iterative query reformulation and model-controlled stopping, are well suited to both and motivate adapting them to this setting.

3.2. Clause Extraction Systems and Benchmarks

Legal NLP has built its own retrieval and question-answering work, mostly apart from the general-domain literature above. The recurring pattern is that retrieval is the bottleneck, yet the retrieval step itself stays single-pass and free of any model direction.

LegalBench (Guha et al., 2023) shows that models reason well over legal text once the right passage is placed in the prompt, but it skips retrieval entirely. LegalBench-RAG, a preprint (Pipitone and Alami, 2024), adds the retrieval step and confirms that retrieval quality is the main bottleneck and that retrieval scores alone do not predict answer quality, though it covers only single-pass retrieval. Work at the clause level repeats this shape. Topollai et al. (2025), in a preprint, run a single-pass retrieve-and-extract pipeline for clause acceptability on proprietary data. ACORD (S. Wang et al., 2025) is a clause retrieval benchmark that finds standard vector retrieval leaves clear room for improvement and directly limits extraction accuracy. Braun et al. (2025), in a preprint, show input format dominates results: GPT-4.1 reaches about 80 percent Exact Match on clean Markdown but drops to about 48 percent on poorly structured plain text. Together these studies fix single-pass retrieve-and-extract as the current standard for contract clause work and confirm chunking and retrieval as the bottleneck.

Top-k single-pass retrieval is the shared baseline on both sides. The legal systems above all pass a fixed set of top-ranked chunks to the model with no further search, and IRCoT, FLARE, Iter-RetGen, and Self-RAG all measure their gains against the same top-k baseline (Trivedi et al., 2023; Jiang et al., 2023; Shao et al., 2023; Asai et al., 2024). That common baseline makes top-k single-pass retrieval the natural comparison for this evaluation.

3.3. LLM-Directed Retrieval for Legal Question Answering

Two recent legal systems do let the language model direct retrieval, LAW and L-MARS, but each combines it with additional components, which makes the effect of retrieval alone hard to read. LAW (Watson et al., 2025) has a code-generation LLM orchestrate domain-specific tools such as section retrieval with BM25, a classical keyword-based ranking function that scores passages by query-term overlap, together with date extraction and party identification, while correcting its own execution errors through a feedback loop. LAW improves over a direct-prompting baseline by up to 92.9 points of accuracy on contract retrieval and multi-hop question answering, which shows that orchestrated tool use scales to specialized legal corpora. The difficulty for isolating the contribution of retrieval is that orchestration, code generation, and tool use are entangled in the reported numbers and cannot be measured separately.

L-MARS, released as a preprint (Z. Wang and Yuan, 2025), splits statutory question answering across several cooperating models over web search and a case-law API, with a dedicated judge step that checks whether evidence is sufficient and current. The benchmarked configuration is single-pass. The authors also release LegalSearchQA (Z. Wang and Yuan, 2025), 50 multiple-choice questions whose answers post-date model training. There L-MARS reaches 96 percent accuracy against a 58 percent zero-shot baseline, while chain-of-thought alone falls to 30 percent by reasoning from stale premises. On 594 bar-exam questions, though, it gains less than one point over zero-shot (55.9 versus 55.2 percent). This split confirms the same pattern as Section 3.1: retrieval helps most when the answer cannot be recalled from the model's own parametric knowledge.

3.4. The Research Gap

The two bodies of work have grown in parallel with little contact. Iterative retrieval shows steady gains on multi-hop QA in general domains but has not been tested on contract clause extraction. Contract clause extraction names retrieval as the main bottleneck but applies single-pass top-k retrieval throughout, without asking what changes if the model directs the search. No published study compares single-pass and iterative LLM-directed retrieval for extractive legal question answering with all other components held fixed.

The legal systems in Section 3.3 do not close this gap. LAW (Watson et al., 2025) fuses retrieval with code generation and tools in a way its results cannot tease apart. L-MARS (Z. Wang and Yuan, 2025) targets statu-

tory questions, where the output is a synthesized multiple-choice answer rather than a verbatim span from a private document. Neither was tested where the correct response is often that no clause exists.

Contract clause extraction is the right task to isolate this variable because it is factual rather than interpretive. Judging what a clause means or whether it favours one party needs legal expertise beyond locating text, and is a professional judgment a system should support, not replace. The EU AI Act (European Parliament and Council of the European Union, 2024) reinforces this through its human oversight requirement (Article 14): legal AI must support meaningful human verification, which means presenting retrievable evidence rather than interpretive conclusions. Verbatim extraction, returning the actual clause text or null, gives the lawyer something to read and verify. Retrieval quality therefore decides whether the system does its job, which makes it the right quantity to measure.

Prior work confirms retrieval as the primary bottleneck for contract clause extraction (Topollai et al., 2025; Pipitone and Alami, 2024; S. Wang et al., 2025; Braun et al., 2025). This thesis addresses the gap by comparing single-pass and iterative retrieval under a controlled design that holds the embedding model, vector store, chunking parameters, language model, and output format constant. It uses CUAD because, as established in Section 4.1, it is the only expert-annotated contract dataset that combines full-document retrieval, verbatim span annotation, and a substantial share of absent-clause instances.

4

Dataset

This chapter describes the data used in the study. It introduces the Contract Understanding Atticus Dataset (CUAD), explains why CUAD is the right choice for an open-corpus clause extraction study and why other contract datasets were set aside, and reports the statistics of the evaluation pool that the experiments draw from. The procedure that turns the raw annotations into question-answer pairs is described later, in Section 5.2, because it is part of the method rather than the data itself.

4.1. The Contract Understanding Atticus Dataset

CUAD (Hendrycks et al., 2021) is a corpus of 510 commercial legal contracts drawn from the public filings of the U.S. Securities and Exchange Commission through its EDGAR system. The contracts cover a range of agreement types, including licensing, distribution, services, and joint-venture agreements, and many run to tens or even hundreds of pages. The dataset was built by the Atticus Project, where trained law students annotated the contracts under the supervision of qualified lawyers, producing more than thirteen thousand clause-level annotations.

Each contract is annotated for 41 categories of legal clauses, such as Governing Law, Change of Control, Anti-Assignment, and Exclusivity. For every contract and category pair, the annotators highlighted the exact verbatim text span that supports the clause, and left the entry empty when the contract contains no clause of that category. The present-or-absent label used in this study is derived from these spans by the question-generation step (Section 5.2), which treats a pair with a valid span as present and an empty one as absent. Two properties follow from how the data was built and matter for this study. First, the supporting span is the exact contract text rather than a paraphrase, which makes verbatim extraction measurable against an expert reference. Second, the dataset is heavily skewed toward absent clauses, since most contracts contain only a subset of the 41 categories. Across the corpus, roughly 70 to 80 percent of the contract and category pairs are absent (Hendrycks et al., 2021), which makes correctly reporting that a clause does not exist as important as locating one that does.

4.2. Why CUAD

CUAD is used because it is the only large, expert-annotated contract dataset that fits all three needs of this study at once. It requires retrieving from the full raw document rather than from a passage handed to the model, it gives both a present-or-absent label and the exact supporting span for each clause, and it contains a large share of absent clauses, which makes the null decision part of the task.

Other contract and legal datasets were considered and set aside because they do not match the task. ContractNLI (Koreeda and Manning, 2021) classifies whether a statement is supported or contradicted by a contract, with no retrieval step. MAUD (S. H. Wang et al., 2023) supplies the relevant text with each question and tests legal interpretation rather than search. LLeQA (Louis et al., 2024) is in Dutch and French under Belgian law, which is hard to compare with the English CUAD contracts. LegalSearchQA (Z. Wang and Yuan, 2025) needs web retrieval for current statutes rather than a clause in a fixed private document. CUAD is therefore the only large-scale, expert-annotated contract dataset that requires retrieving from the full raw document, gives both a label and a verbatim span, and includes many absent-clause cases, which are the three properties this evaluation needs.

4.3. From CUAD to the Evaluation Pool

CUAD itself contains no natural-language questions. For each contract and clause category it provides only a present-or-absent label and, when the clause is present, the supporting span. The question-answer pairs used in this study are therefore not part of the dataset. They are constructed from these annotations by the question-generation procedure described in Section 5.2, which is a step of the method rather than a property of CUAD. Applied to all 510 contracts and the 38 categories treated as extractable, that procedure yields a pool of approximately 18000 question-answer pairs. Because each pair simply carries over the present-or-absent label of its annotation, the class balance of the pool is inherited from CUAD and is approximately 78 percent absent (Hendrycks et al., 2021).

Two evaluation samples are drawn from this pool, and the reasoning behind their sizes is given in Section 6.2. The main experiment uses 5427 pairs evaluated on the locally hosted model, and a smaller balanced set of 500 present and 500 absent instances is used for the cross-model comparison, because the second model is a paid API. The balanced set uses equal classes so that present and absent performance can be read separately, since on the natural 78 percent absent mix a system that almost always says absent would already look accurate.

5

Methodology

This chapter walks through the methodology of the study. It begins by stating the task and showing how the evaluation questions are made, then explains how the corpus is indexed, introduces the baseline and the proposed method, and lays out the metrics and statistical tests used to judge them. The goal throughout is a clean comparison, where the retrieval strategy is the only thing that changes between the two. Figure 5.1 gives a high-level view of the whole pipeline, from the contract corpus through indexing and retrieval to the structured output that is scored.

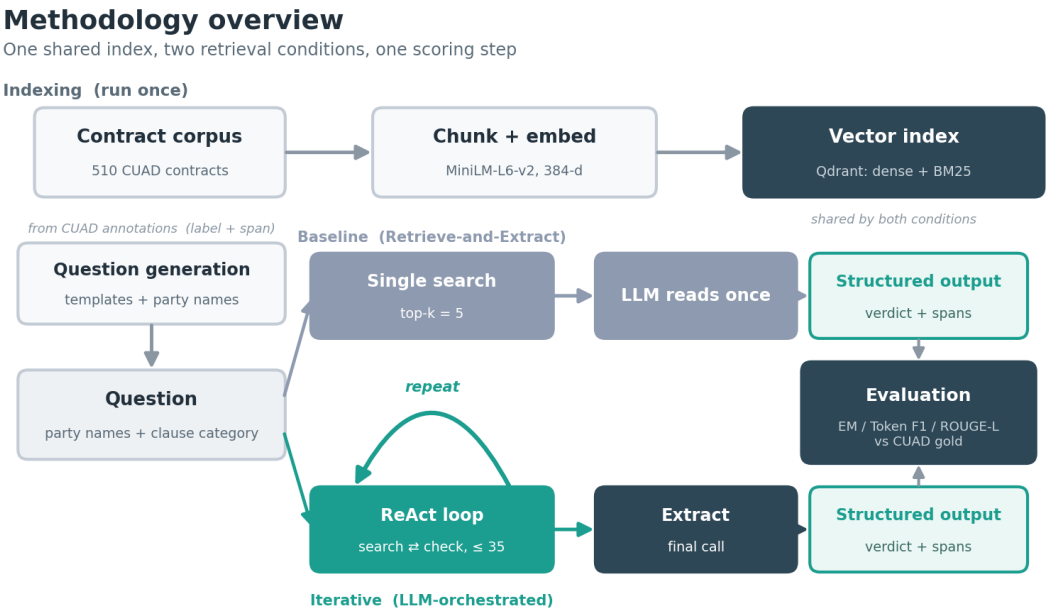

Figure 5.1: High-level overview of the methodology. The contract corpus is split into chunks, embedded, and stored once in a shared vector index. At evaluation time each question is answered under two conditions that share every component except retrieval: the single-pass baseline runs one search and reads the result passively, while the iterative method lets the model run a loop of searches before a final extraction step. Both produce the same structured output, which is scored against the CUAD gold annotation.

5.1. Task Formulation

The task is open-corpus contract clause extraction. The data it operates on, the CUAD corpus, is described in Chapter 4.

The setup can be built up in a few steps. The study starts from a corpus of contracts D, with $|D| = 510$. Every contract is split into smaller passages, called chunks, and the chunks of all contracts together form the searchable index X. A chunk $x \in X$ is simply a continuous piece of one contract's text.

CUAD groups clauses into 38 categories, each one a type of clause such as Governing Law or Change of Control. Let C be this set of 38 clause categories, and let c stand for a single category taken from it. A category and the clause it labels are treated as the same thing: c is the label, and the clause is the contract text that carries it.

Two things describe each contract $d \in D$. It is signed by a set of parties $P(d)$, the companies named in the agreement, and it contains a number of clauses. The clauses present in d cover only some of the categories, so they form a subset $\mathrm{Cat}(d) \subseteq C$. No contract carries a clause of every category, which is why most contract-category pairs turn out to be absent.

An evaluation entry is built from one contract $d \in D$ and one clause category $c \in C$, and it has two parts: a question q and a set of gold spans S^*. The question q asks whether the agreement between the parties $P(d)$ contains a clause of category c, and it names those parties but gives no file name or document identifier. The gold spans are the verbatim text the annotators highlighted for that clause. If a clause of that category is present, meaning $c \in \mathrm{Cat}(d)$, then S^* holds its spans, and if it is absent then $S^* = \emptyset$.

The system answers each entry with a response of the same shape: a present-or-absent verdict $\hat{a} \in \{0, 1\}$ together with a set of predicted spans $\hat{S}$, where $\hat{S} = \emptyset$ when $\hat{a} = 0$. Each predicted span is a piece of text taken from the retrieved chunks, so a span can be part of a chunk or a whole chunk. Scoring compares the response $(\hat{a}, \hat{S})$ against the gold (a^*, S^*), where the gold verdict a^* is one exactly when $c \in \mathrm{Cat}(d)$.

To produce that response the system must first locate the correct contract within the corpus, then find the relevant clause inside it. Because the question carries only party names, the contract has to be recovered from the content of the chunks rather than from a document identifier.

5.2. Question Generation

Evaluation questions are generated from the CUAD master annotation file, which records one row per contract and one column pair per clause category: one column for the ground-truth answer and one for the verbatim supporting clause text.

Category selection. CUAD defines 41 annotation categories (Hendrycks et al., 2021). Three are treated as contract metadata rather than extractable clauses (Document Name, Parties, and Agreement Date) and are excluded from question generation. The remaining 38 categories form the evaluation set.

Question phrasing. Each question identifies the contract by the names of the contracting parties rather than by document title or file name. Party names are extracted from the Parties column of the annotation file and cleaned by removing alias blocks in parentheses. A question template is applied for each category. For example, the boolean category *Anti-Assignment* produces the question: "Does the agreement between [Party A] and [Party B] require consent or notice before the contract can be assigned to a third party?" The Governing Law category produces: "Which state or country's law governs the agreement between [Party A] and [Party B]?"

This phrasing has a direct consequence for retrieval. Let $e(\cdot)$ be the embedding function and $\mathrm{sim}(q, x)$ the similarity between a question q and a chunk x. Single-pass retrieval returns the top-k chunks

$$R(q) = \underset{x \in X}{\arg\text{top-k}}\ \mathrm{sim}\big(e(q), e(x)\big).$$

A system that pre-filtered the index by document name would need a document identifier inside q, but q holds only party names, so no such filter can be applied and the search ranges over the whole index X. The correct contract therefore has to be recovered from the content similarity $\mathrm{sim}(e(q), e(x))$ between the question and the contract text, not from metadata.

Answer and supporting clause. For each contract–category pair the ground-truth label is taken directly from the annotation file: a pair is labelled *present* if the clause text column has content, and *absent* otherwise. The verbatim supporting span is the clause text from that column. Pairs where the answer column is empty or contains redacted text (marked with square brackets) are filtered out.

The resulting pool of question-answer pairs, its class balance, and the two evaluation samples drawn from it are described in Chapter 4.

5.3. Corpus Indexing

All 510 CUAD contracts are indexed before evaluation begins. Each contract is split into chunks using a recursive character text splitter with a chunk size of 4000 characters and an overlap of 200 characters, with sentence boundaries preserved where possible to avoid splitting clause text mid-sentence. Each chunk is encoded using the `all-MiniLM-L6-v2` sentence-transformer model, producing 384-dimensional dense vectors, and stored in a Qdrant collection alongside the full chunk text and contract metadata (document name, party names, and source file name). This model is chosen because it is a small, widely used sentence embedder that runs quickly on local hardware, which keeps the full corpus index and every retrieval call cheap to compute. Its absolute quality is not central to the study, because the controlled design holds the embedder fixed across both conditions, so any difference in results comes from the retrieval strategy rather than from the choice of embedding model. The metadata is stored in the chunk payload alongside the text. Questions are phrased using only party names and contain no document identifier, so the initial retrieval query is guided solely by the question content. Once chunks are retrieved, the document name and file name are included in the context provided to the language model. The iterative condition uses this information to scope follow-up searches to the identified contract.

Retrieval is hybrid. Each query is scored against the index by both a dense vector from the embedding model and a sparse BM25 vector, and the two ranked lists are merged with Reciprocal Rank Fusion (Cormack et al., 2009). Hybrid retrieval suits contract clause extraction because the two signals catch different things. The dense vectors capture meaning, so a question phrased in plain language can match a clause that uses different wording. BM25 captures exact word matches, which matter here because the questions carry party names and the contracts use precise defined terms and section labels that a small dense model can miss. Combining the two recovers clauses that either signal alone would rank too low. Both the baseline and the proposed method use this same hybrid configuration, so the comparison stays controlled and any difference in results comes from the retrieval strategy rather than from the index.

5.4. Conditions Under Comparison

The proposed iterative method is evaluated against a single-pass baseline. The two conditions differ only in whether the language model is passive or active at the retrieval step, and all other components are identical. Figure 5.2 contrasts the two pipelines side by side, Figure 5.3 shows a concrete run of the iterative method on a single question, and Figure 5.4 shows how its context window grows across iterations.

Both retrieval conditions share the same embedding model, vector store, chunking parameters, retrieval budget ($k = 5$ chunks per retrieval call), and language model. The specific models used are described in Chapter 6.

Retrieve-and-Extract (R&E) is the baseline. It is the standard single-pass pipeline that prior contract clause extraction work relies on (Section 3.2), and it serves here as the reference point the proposed method is measured against rather than as a contribution of this study. The natural-language question is embedded and used to run a single similarity search over the full 510-contract index, returning the top-$k = 5$ most relevant chunks across all contracts. These chunks are concatenated with the original question and passed to the language model, which produces a structured factual claim and supporting spans in one step (Figure 5.2, top). The language model has no role in deciding what to retrieve or in evaluating whether the returned chunks are sufficient.

LLM-Orchestrated Iterative Retrieval is the proposed method, referred to as *Iterative* throughout the results. It places retrieval inside an iterative ReAct loop (Yao et al., 2023). An orchestrator language model receives the question and decides at each step whether to issue a retrieval call, what query to use, and whether the evidence gathered so far is sufficient to produce a final answer. The retrieval tool is implemented as a LangChain tool wrapping the hybrid search function over the Qdrant collection. Retrieved chunks are added to the model's context one step at a time, so the context window grows as the loop proceeds (Figure 5.4). The model can reformulate its search query based on what it has found, follow cross-references by issuing targeted follow-up queries, and scope subsequent searches to a specific section or contract once it has identified which document is relevant (Figure 5.3). The loop terminates when the model decides the evidence supports a definitive answer, when it determines that no relevant clause exists and returns null, or when a per-question limit of 35 tool calls is reached. Once the loop ends, a separate structured-output LLM call over all accumulated evidence produces the final factual claim and supporting spans. This two-stage design separates the retrieval orchestration from the extraction step.

The proposed method differs from the baseline in exactly one respect: whether the language model di-

rects the retrieval process or reads a fixed retrieval output passively. Both produce the same structured response format, and neither generates free-text answers from parametric memory.

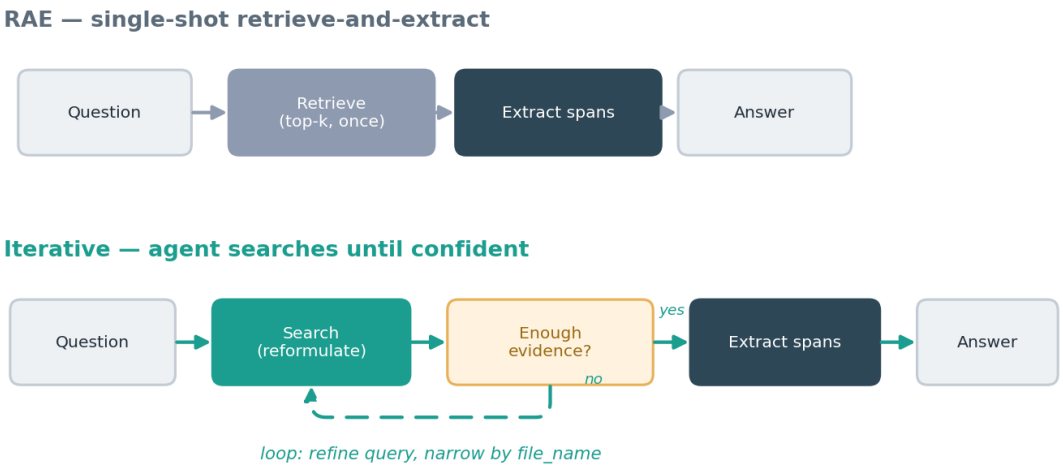

Figure 5.2: The two retrieval architectures. In Retrieve-and-Extract (top), the query is embedded once, the top-k chunks are returned in a single call, and the language model reads them passively to produce the structured output. In the iterative method (bottom), the language model runs a search, checks whether the evidence is enough, and loops back to reformulate the query or narrow the search when it is not, before extracting the final spans. The only difference between the two is whether the language model directs retrieval or reads a fixed output.

5.5. Evaluation Methodology

This study evaluates both conditions using Exact Match, token-level F1, and ROUGE-L, as defined in Section 2.6. Exact Match is reported at three levels: aggregate over all instances, restricted to present instances, and restricted to absent instances. Token-level F1 and ROUGE-L are reported on present instances, where a gold supporting span exists to compare against. The reasoning for this particular set of metrics, and how it connects to the demands of the legal setting, is part of the experimental design and is given in Section 6.4.

Statistical testing. The tests themselves are defined in Section 2.7; this section states how they are applied. Because both conditions answer the same questions, every measurement is paired rather than independent, so paired tests are used throughout. Exact Match is a yes-or-no outcome per question, so the two conditions are compared with McNemar's exact test (McNemar, 1947). Token-level F1 and ROUGE-L are continuous scores whose paired differences are not normally distributed, since many spans score zero or near one, so they are compared with the Wilcoxon signed-rank test (Wilcoxon, 1945). Alongside the tests, 95 percent confidence intervals are computed by bootstrap resampling over contracts (Efron, 1979), so the intervals account for the fact that clauses from the same contract are not independent. The results of these tests are presented in Chapter 7.

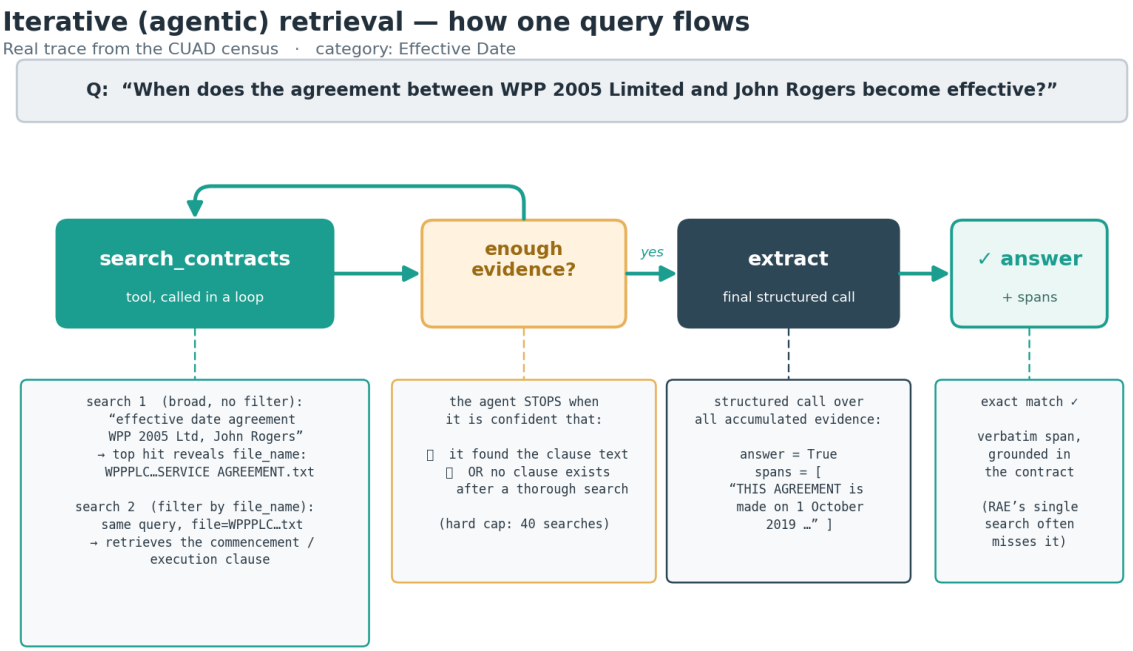

Figure 5.3: A concrete run of the iterative method (Yao et al., 2023) on one CUAD question. The model calls the `search_contracts` tool, first broadly and then narrowed to the identified contract by file name, checks at the gate whether the evidence is sufficient, and only then makes the final structured extraction call. The answer is a verbatim span grounded in the retrieved text. This loop is what a single retrieval pass cannot do.

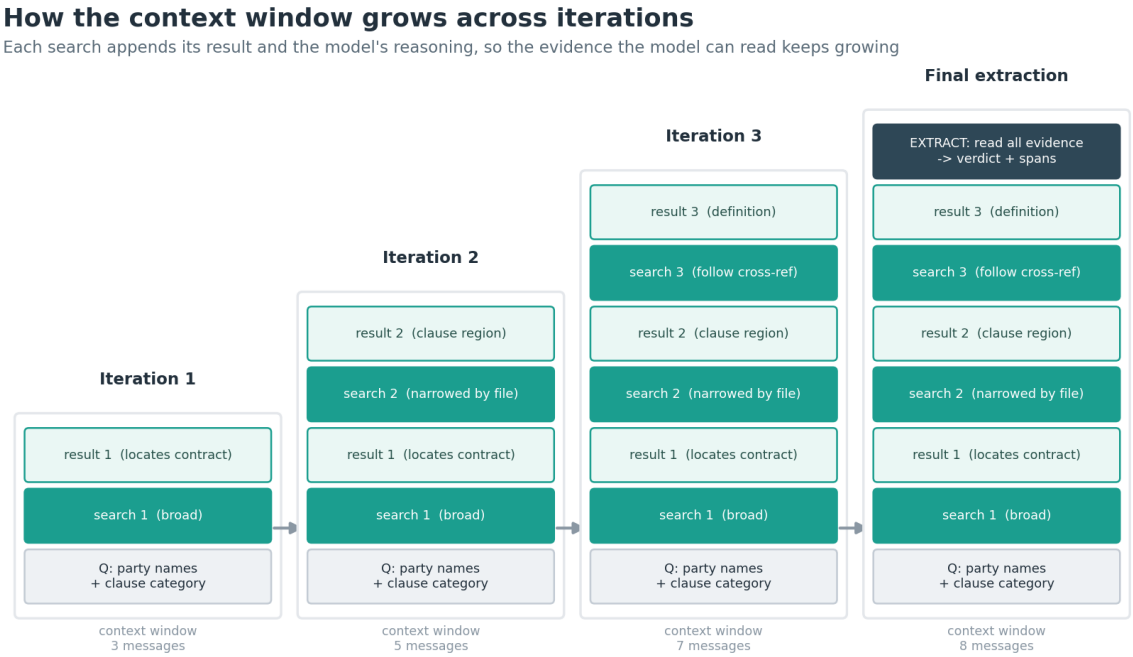

Figure 5.4: How the context window of the iterative method changes across iterations on a single question. Each search call appends its retrieved chunks and the model’s short reasoning to the running message history, so the evidence the model can read grows from one step to the next. The first query is broad and locates the contract, later queries are narrowed to that contract and follow cross-references, and once the accumulated context is judged sufficient the loop stops and a final extraction call reads the whole window to produce the answer.

6

Experiments

This chapter describes how the experiments are carried out in practice. It gives the hardware and the two language models used, the two evaluation samples drawn from the question pool of Chapter 4, the software implementation of each condition, and the pipeline that turns model outputs into scores. It also explains why the chosen metrics suit the legal setting.

6.1. Experimental Setup

All experiments were run on a MacBook Pro with an Apple M3 Max chip and 48 GB of unified memory. All local components, including Qdrant and Ollama, ran on this machine.

Two language models were evaluated. GPT-5-mini was accessed via the OpenAI API. Gemma4 e4b is a 4-billion-parameter model run locally via Ollama on the same machine. Running Gemma4 e4b locally avoids API costs and external latency, and makes those runs fully reproducible without requiring access to an external service. Both models were run with temperature set to 0 to reduce sampling variance across runs.

Qdrant ran locally as a Docker container. The collection uses hybrid indexing: dense vectors from the `all-MiniLM-L6-v2` sentence-transformer model and sparse BM25 vectors from `fastembed`, combined via Reciprocal Rank Fusion at query time.

6.2. Evaluation Sample

CUAD (Hendrycks et al., 2021) is used for evaluation. A full description of the dataset is given in Section 4.1.

Main experiment. The main experiment evaluated both conditions using Gemma4 e4b on a dataset of 5427 paired questions, sampled from a total pool of approximately 18000 pairs. The evaluation was limited to this subset due to the substantial computational requirements: processing the 5427 pairs alone required slightly more than two weeks of local computation time. This estimate does not include the time spent developing the evaluation pipeline or conducting the separate cross-model experiments described below. Given these constraints, extending the evaluation to the full dataset was not feasible within the available timeframe. The questions are generated by iterating through the contracts in order, and the run processed them in that same order, so the completed 5427 pairs form a continuous slice of the dataset rather than a hand-picked subset. This slice already spans a large number of contracts and all 38 clause categories, with a substantial number of instances in every category, and it keeps CUAD's natural skew toward absent clauses (1975 present and 3452 absent) (Hendrycks et al., 2021). Covering every category in dataset order, with the natural class skew preserved, is why this run is treated as representative of the full pool for the comparison studied here.

Cross-model set. The second model, GPT-5-mini, is a paid API model, so running it at the same scale would have been expensive. The cross-model comparison therefore uses a smaller, class-balanced sample of 500 present and 500 absent instances drawn from the pool (seed 42), giving 1000 pairs across the 38 categories, run on both models. Balancing the classes removes the 78% absent-clause skew so that present and absent performance can be read separately. The reliance on this smaller set for the API model is a cost limitation, discussed in Section 8.2.

The evaluation is open-corpus: for each question, the full 510-contract index is searched. The system has no information about which contract the question refers to and must find it through retrieval. Ground-truth labels and supporting clause texts are drawn from the CUAD annotations.

6.3. Implementation

The implementation is a set of Python scripts organised as a sequential pipeline. Each condition runs as an independent script: `03_run_rae.py` for the Retrieve-and-Extract condition and `04_run_iterative.py` for the iterative condition. Shared configuration (Qdrant host, collection name, model provider, and output directory) is supplied via environment variables and a `common.py` module imported by both scripts.

LangChain provides the retrieval tool interface and the LLM wrappers. LangGraph provides the retrieval loop for the iterative condition.

The R&E condition is implemented as a two-node LangGraph graph. A retrieve node calls the vector store and collects the top-k chunks. A generate node calls the language model with those chunks and the original question, producing a structured response via `with_structured_output`.

The iterative condition uses LangGraph's `create_react_agent` with a single tool, `search_contracts`, that wraps the hybrid search function over the Qdrant collection. The iterative loop accumulates retrieved chunks in the message history. After the loop finishes, a separate structured-output LLM call over all accumulated evidence produces the final answer. This two-stage design separates the retrieval orchestration from the extraction step.

Both conditions use LangChain's `with_structured_output` to enforce a fixed response schema: a boolean answer field and an optional list of verbatim span strings.

Results are written to JSON files named by condition and model, for example `rae_openai_gpt_5_mini.json`. The pipeline is resumable: completed question IDs are tracked in the output file and skipped on restart, so a run can be interrupted and continued without reprocessing questions that already have results.

Figure 6.1 shows the internal data flow of the two conditions as implemented, from the shared vector store through the LangGraph graphs to the structured output that is written to disk.

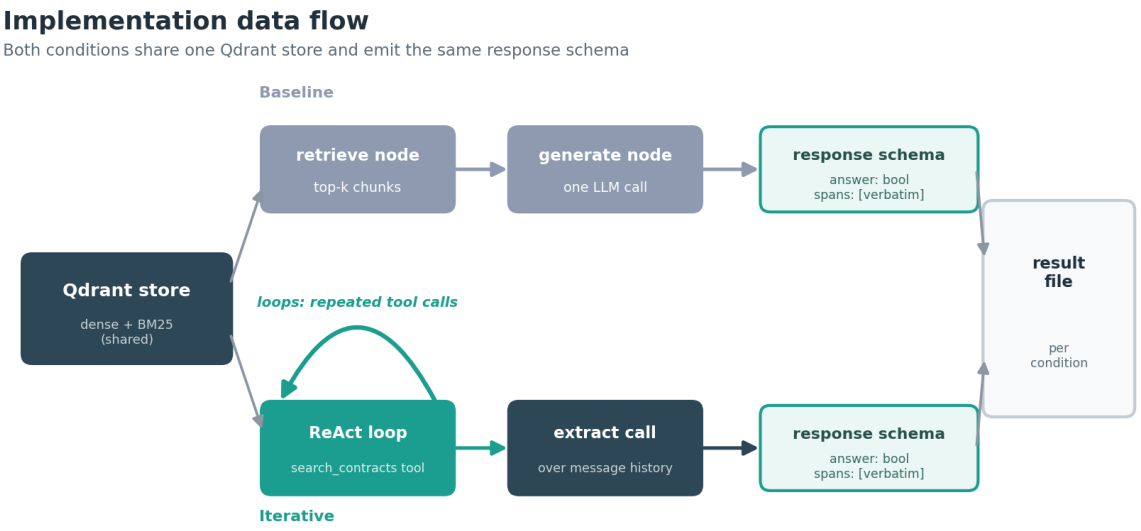

Figure 6.1: Internal data flow of the two conditions. The single-pass baseline is a two-node LangGraph graph: a retrieve node fetches the top-k chunks from the shared Qdrant store and a generate node calls the language model once to produce the structured output. The iterative condition uses a ReAct loop whose single `search_contracts` tool wraps the same store, accumulating retrieved chunks and reasoning in the message history, after which a separate structured-output call reads the accumulated evidence and produces the final answer. Both paths emit the same fixed response schema, a boolean verdict and an optional list of verbatim spans, which is written to a per-condition result file.

6.4. Evaluation Pipeline

Each condition processes every question in the evaluation sample independently. For each question, the condition runs its retrieval pipeline and produces a structured response: a boolean factual claim and an optional list of verbatim supporting text spans. This response is compared against the ground-truth annotation

to compute EM, token-level F1, and ROUGE-L as defined in Section 2.6.

EM is reported at three levels: aggregate across all instances, restricted to positive instances (ground-truth clause exists), and restricted to absent instances (ground-truth clause is missing). This split separates each condition's ability to locate real clauses from its ability to correctly abstain when no clause is present.

This particular set of metrics is chosen for the legal setting for three reasons. They cover the full pipeline output rather than only the retrieval step, they are deterministic and reproducible against the CUAD expert annotations (Hendrycks et al., 2021), and they measure the factual claim and the supporting span as two independent dimensions. Accurate clause extraction can fail in two distinct ways: a system may return a correct verdict with incorrect supporting text (high EM, low F1), or retrieve the correct text but draw the wrong factual conclusion (low EM, high F1). The EU AI Act classifies AI systems used in legal interpretation as high-risk and requires under Article 14 that users can oversee and correct system outputs (European Parliament and Council of the European Union, 2024). Reporting EM and span-level metrics separately makes both failure modes visible to a reviewing professional, in line with that requirement.

Each model-condition pair produces a separate result file. The evaluation script scans for all result files and computes metrics for each.

7

Results

This chapter reports the evaluation results. The main experiment is the large run on Gemma4 e4b, the open-weight model hosted locally through Ollama, over 5427 paired questions. This run is treated as the primary result because it is the largest, and because it could be carried out at no API cost on local hardware, as explained in Section 6.2. A second, smaller run on 1000 balanced questions, executed on both Gemma4 e4b and GPT-5-mini, is then used to test whether the finding holds across two very different models. That run is limited to 1000 questions mainly because GPT-5-mini is a paid API model and a larger run on it would have been expensive, which is itself a limitation discussed in Section 8.2. R&E (Retrieve-and-Extract) is the single-pass baseline and Iterative is the LLM-orchestrated method, both defined in Chapter 5. EM Overall is Exact Match across all instances, where a prediction counts as correct only when the presence verdict and the supporting span are jointly right, as defined in Section 2.6. Each Exact Match figure is the proportion of instances scored correct, that is the mean of the per-instance one-or-zero outcomes, not a median. EM Pos and EM Abs restrict Exact Match to present and absent instances. Token F1 measures unordered token overlap between predicted and gold spans, and ROUGE-L measures the longest common subsequence, so it also captures word order. Both span metrics are defined in Section 2.6.

Sections 7.1 to 7.4 answer the research question on the main experiment, namely what change in clause extraction accuracy follows when the language model directs retrieval through LLM-orchestrated iterative retrieval compared to single-pass retrieve-and-extract, and where any change comes from. Section 7.5 then checks that the gain is not specific to one model by repeating the comparison on a second model.

7.1. Main Experiment: Accuracy Gain from Iterative Retrieval

Table 7.1 shows the aggregate scores for the main experiment, the 5427-pair run on Gemma4 e4b.

Table 7.1: Main experiment results, Gemma4 e4b on the 5427-pair run. EM Overall: Exact Match over all instances. EM Pos and EM Abs: Exact Match on present and absent instances. Token F1: token-overlap F1 between predicted and gold spans (Rajpurkar et al., 2016). ROUGE-L: longest-common-subsequence F1 (C.-Y. Lin, 2004). Each Exact Match value is a proportion (the mean of the per-instance one-or-zero outcomes). The Gain row is the iterative score minus the baseline score. The best score in each column is shown in bold. Values rounded to three decimals.

Condition	EM Overall	EM Pos	EM Abs	Token F1	ROUGE-L
R&E (Gemma4 e4b)	0.687	0.668	0.698	0.341	0.308
Iterative (Gemma4 e4b)	**0.744**	**0.734**	**0.750**	**0.505**	**0.467**
Gain (Δ)	+0.057	+0.066	+0.052	+0.164	+0.159

The iterative method beats the single-pass baseline on every metric. Overall Exact Match rises from 0.687 to 0.744, a gain of 0.057. The span metrics rise much more, with Token F1 climbing from 0.341 to 0.505 and ROUGE-L from 0.308 to 0.467. Figure 7.1 shows all three gains together, a relative rise of about 8 percent on Exact Match but 48 percent on Token F1 and 52 percent on ROUGE-L.

The size of the span gain relative to the Exact Match gain is the key reading. Exact Match only checks the presence verdict, while Token F1 also measures how much of the correct clause text was found. Because F1

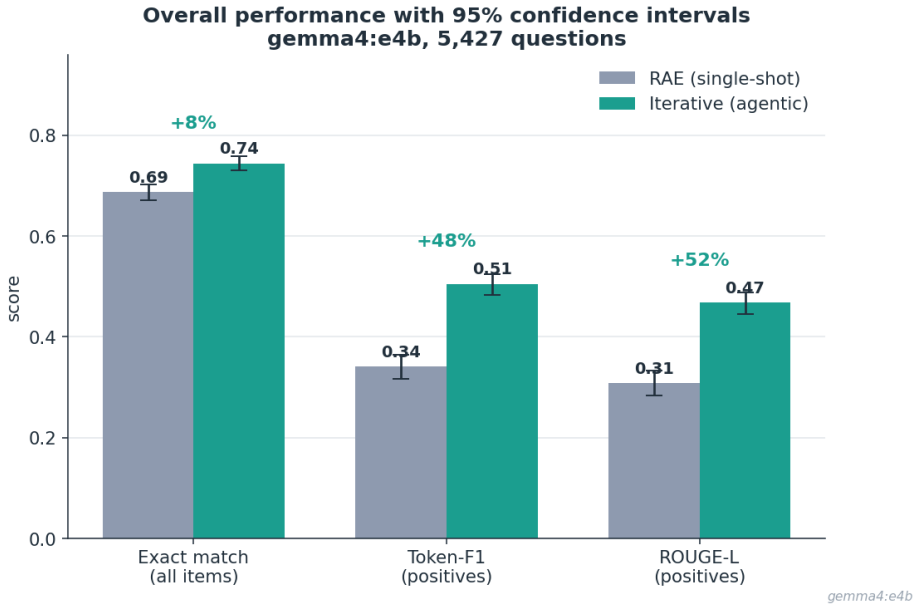

Figure 7.1: Overall scores for the main experiment, Gemma4 e4b on the 5427-pair run, for the single-pass baseline and the iterative method, with 95% confidence intervals. The Exact Match bar is EM Overall, scored over all instances including absent clauses, while Token F1 and ROUGE-L are scored on present instances only, where a gold span exists. The iterative method improves all three metrics, and the relative gain on the span metrics, Token F1 and ROUGE-L, is far larger than on Exact Match.

climbs far more than EM, the iterative method is not just flipping more verdicts from absent to present, it is returning more complete clause text. That fits how legal contracts are written, where a clause often points to a definition or condition stated in another section, as explained in Section 2.2. A single retrieval pass cannot follow that pointer, but the iterative loop can issue a follow-up search once it sees one. Across all conditions the two span metrics move together, with ROUGE-L staying a little below Token F1, by 0.03 to 0.05 in every case, so the span gains are not an artefact of one metric's formula. What this small, steady gap implies for the quality of the extracted text is taken up in Section 8.

7.2. Statistical Significance of the Results

The gains in Table 7.1 are measured on a sample, so the next question is whether they are larger than what sampling noise alone could produce. The tests used are the paired tests introduced in Section 5.5: McNemar's exact test for the binary Exact-Match outcomes, the Wilcoxon signed-rank test for the continuous span metrics, and 95 percent confidence intervals from bootstrap resampling over contracts for the effect sizes.

Figure 7.2 shows the paired Exact-Match outcomes. The iterative method is correct on 934 questions where the baseline fails, while the baseline is correct on only 623 where the iterative method fails. McNemar's test over these 1557 disagreements gives $p = 3.2 \times 10^{-15}$, and the overall EM gain of 0.057 has a 95% interval of 0.043 to 0.072, so the interval stays well above zero. The span gains are even clearer. Wilcoxon's test gives $p = 1.9 \times 10^{-65}$ for Token F1 and $p = 6.0 \times 10^{-62}$ for ROUGE-L. Every gain is significant far beyond the 0.001 level, so sampling noise is not a plausible explanation for the difference.

7.3. Source of the Gain

Figure 7.3 breaks the Exact-Match change down by clause category, iterative minus baseline.

The gain is not spread evenly. It is concentrated in a group of categories where single-pass retrieval misses clauses that are actually present. The largest improvements are in Change Of Control (0.43 to 0.79), Governing Law (0.68 to 0.92), Insurance (0.62 to 0.91), and No-Solicit Of Employees (0.72 to 0.92), all significant by McNemar's test (McNemar, 1947). These are categories where the answer often depends on text that is spread across the contract or defined away from the clause itself, which is exactly the case the iterative loop is built for.

The same per-category breakdown for the span metrics, Figure 7.4, tells a stronger version of the same story. The Token F1 and ROUGE-L gains are positive almost everywhere and larger than the Exact-Match

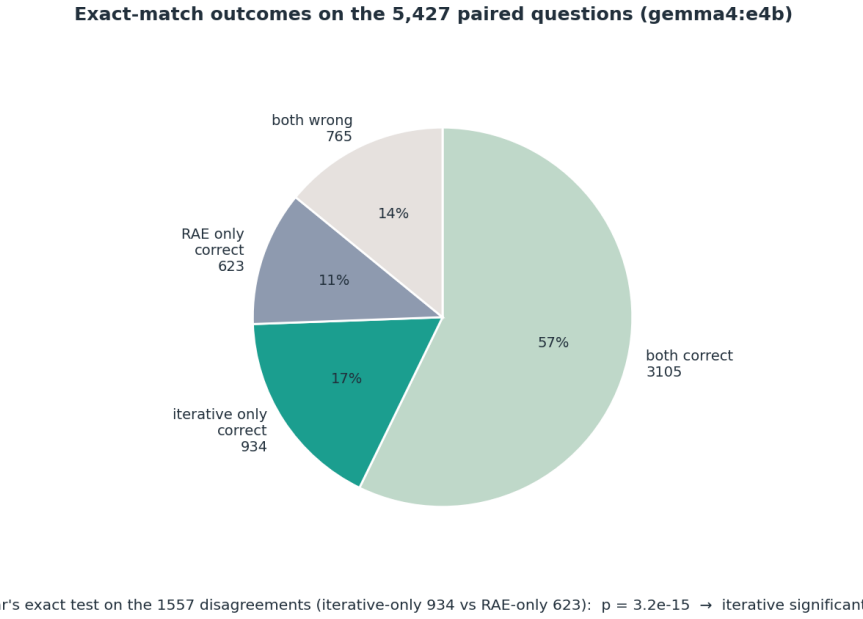

Figure 7.2: Paired Exact-Match outcomes for the main experiment, Gemma4 e4b on the 5427-pair run. Both methods agree on 71% of questions (57% both correct, 14% both wrong). The disagreements favour the iterative method, 934 to 623, and McNemar's exact test (McNemar, 1947) over these gives $p = 3.2 \times 10^{-15}$.

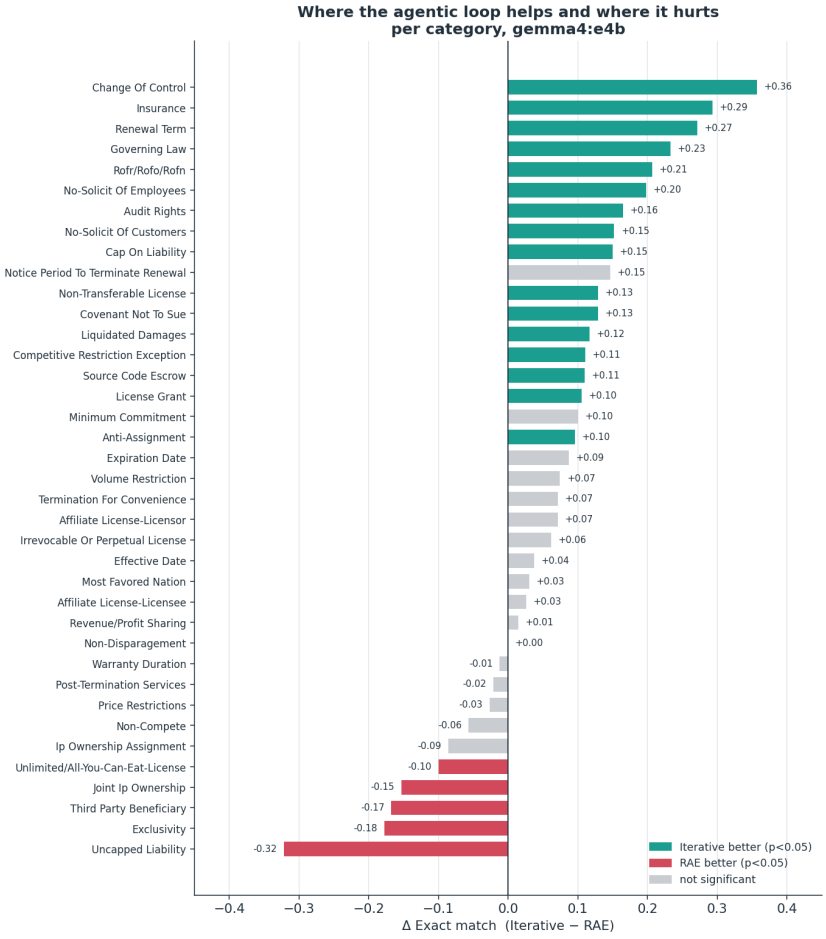

Figure 7.3: Change in Exact Match per clause category, iterative minus baseline, for the main experiment (Gemma4 e4b, 5427-pair run). Most categories improve, a handful regress, and the regressions are the absent-clause categories discussed in Section 7.4.

gains, which confirms that the iterative method mainly recovers more complete clause text rather than only flipping verdicts.

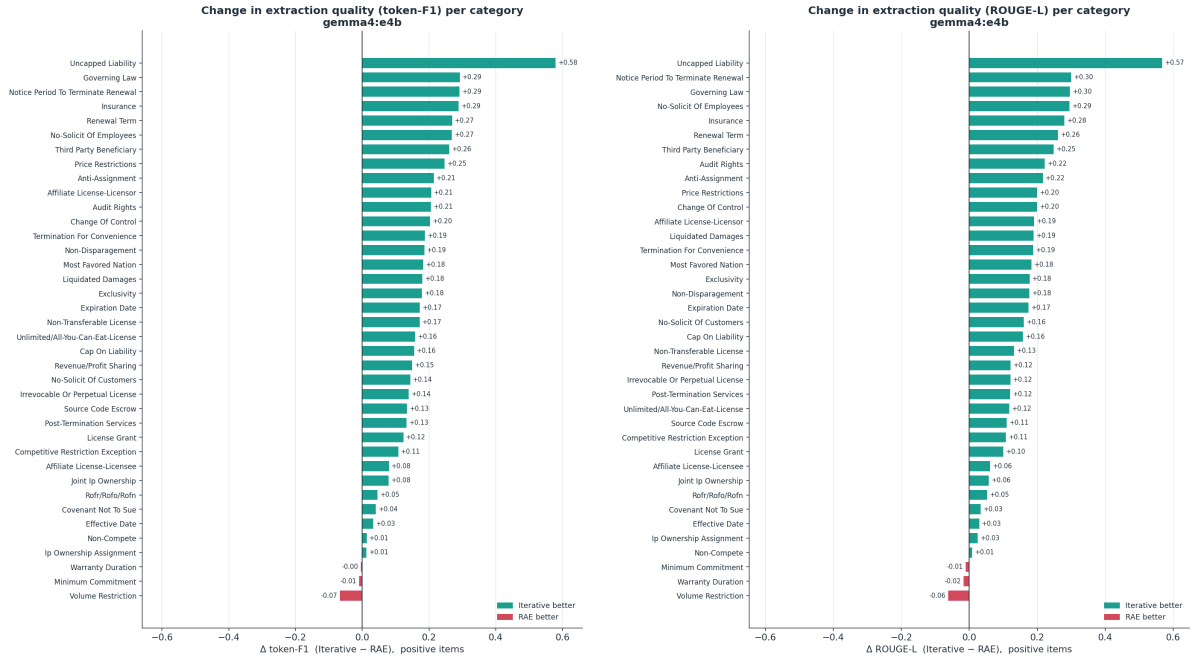

Figure 7.4: Per-category change in the span metrics, iterative minus baseline, for the main experiment (Gemma4 e4b, 5427-pair run). Token F1 (left) and ROUGE-L (right). Both rise across almost every category, and by more than Exact Match in Figure 7.3.

A smaller group of categories moves the other way. Exclusivity drops from 0.79 to 0.62 and Joint IP Ownership from 0.75 to 0.60, both significant. These regressions are not random. They sit almost entirely in categories where the clause is usually absent, and they share one cause, which the next section examines directly.

7.4. Cost of the Gain: False Positives on Absent Clauses

The cost of searching harder is more false positives on absent clauses. When the iterative loop keeps searching, it sometimes assembles enough loosely related text to claim a clause exists when it does not. Figure 7.5 shows the five categories where the baseline scores higher, measured only on absent instances, where a perfect score of 1.0 means no false positives. In every one of these categories the iterative method produces more false positives than the baseline, for example dropping from 0.90 to 0.41 on Uncapped Liability and from 0.85 to 0.41 on Exclusivity.

This trade-off is the central behaviour of the iterative method. Overall it is favourable, since the method gains far more present clauses than it loses to false positives, which is why every aggregate metric improves. Whether the trade is worth it in a given setting depends on which error is more costly, a point taken up in Section 8.1.

7.5. Generalisation of the Gain Across Models

A natural follow-up is whether the gain depends on the particular model used. The dependence on model is tested on the 1000-pair balanced set, the only set run on both models. Table 7.2 lists all four conditions, and Figure 7.6 shows them with the per-model gains. In that figure the Exact Match group is the strict present-or-absent verdict scored over all items, while Token F1 and ROUGE-L score how much of the gold clause text is recovered and only on present instances, so the much larger span gains there are not directly comparable to the Exact-Match gains.

Both models improve under the iterative method, and both gains are significant. GPT-5-mini, a large closed API model, gains 0.089 on overall Exact Match (McNemar $p = 5.5 \times 10^{-8}$), and Gemma4 e4b gains 0.065 ($p = 5.2 \times 10^{-5}$), with both span metrics rising by 0.15 or more for each model (Wilcoxon $p < 10^{-15}$ in every case). The gain therefore does not come from the scale or the API access of one model. It comes from

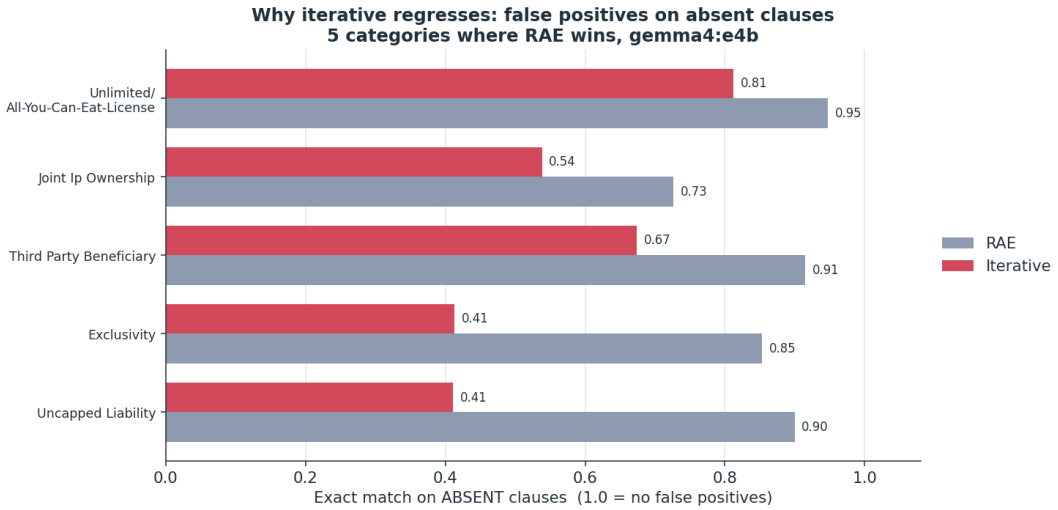

Figure 7.5: Exact Match on absent instances only, for the five categories where the baseline beats the iterative method (Gemma4 e4b, 5427-pair run). A score of 1.0 means no false positives. The iterative method scores lower in each, so its regressions trace back to false positives on clauses that are not there.

Table 7.2: Cross-model comparison on the 1000-pair balanced set, run on both models. Columns are as in Table 7.1.

Condition	EM Overall	EM Pos	EM Abs	Token F1	ROUGE-L
R&E (Gemma4 e4b)	0.676	0.652	0.700	0.346	0.315
R&E (GPT-5-mini)	0.711	0.536	0.886	0.314	0.283
Iterative (Gemma4 e4b)	0.741	0.714	0.768	0.500	0.472
Iterative (GPT-5-mini)	0.800	0.860	0.740	0.538	0.490

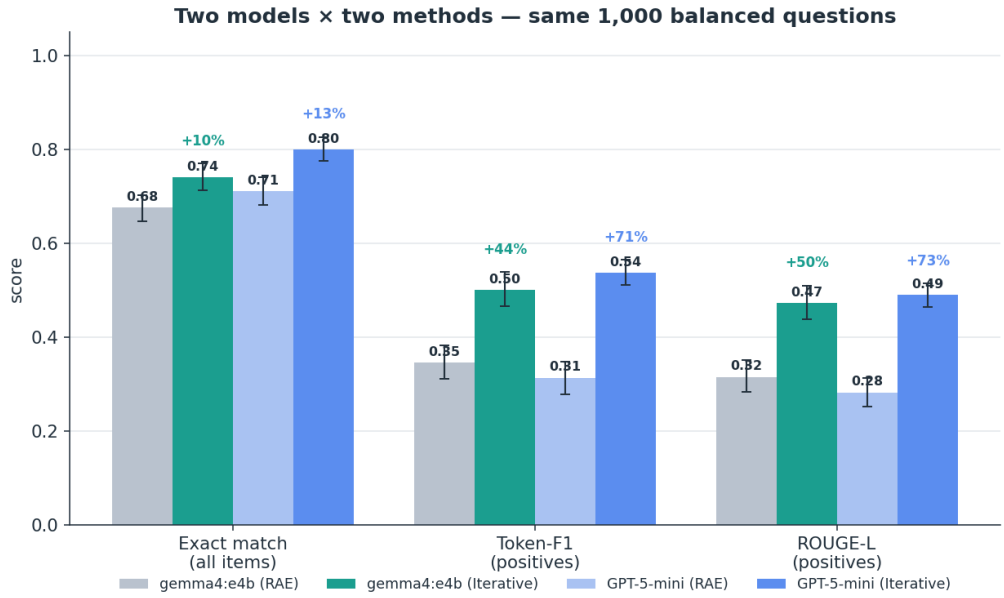

Figure 7.6: Cross-model comparison on the 1000-pair balanced set, with the iterative-versus-baseline gain shown above each model's bars. The left group is Exact Match, the strict present-or-absent verdict scored over all 1000 items, including absent clauses. The other two groups are different metrics: Token F1 and ROUGE-L score how much of the gold clause text the system recovers, and only on instances where a clause is actually present. The difference in scoring scope is why the Exact-Match gains (about 10 to 13 percent) look smaller than the span gains (44 percent and above): the metrics measure different things, not the same thing at different sizes. Both models still improve on all three, and under the iterative method the small local model matches the large API model on the span metrics. Error bars are 95% confidence intervals.

the retrieval design, which both models share. The Gemma4 e4b result on this balanced set, a rise from 0.676 to 0.741, matches the 0.687 to 0.744 rise of the main experiment, so the two sets agree and the balanced set can be read against the larger run with confidence.

A further comparison sharpens the point. Under the iterative method, GPT-5-mini still leads on overall Exact Match, 0.800 against 0.741, and that lead is significant ($p = 3.5 \times 10^{-5}$). On span quality, however, the two models are statistically tied. Wilcoxon's test finds no significant difference in Token F1 ($p = 0.07$) or ROUGE-L ($p = 0.50$) between the two iterative conditions. In plain terms, once the small local model is allowed to direct its own search, it extracts clause text about as well as the large API model. The practical reading is that a team that cannot use a commercial API can still gain most of the benefit with a small model run on its own hardware.

8

Discussion

The results answer the research question directly. Letting the model direct retrieval improves clause extraction accuracy over the single-pass baseline. In the main experiment, the 5427-pair run on Gemma4 e4b, the iterative method beats the baseline on every metric, and the gain is statistically significant in every case (Section 7.2).

The gain is also not specific to one model. A large closed API model and a small open-weight model run on local hardware both improve under the iterative method, and by a comparable margin (Section 7.5). The comparable improvement across both models is the strongest sign that the benefit is a property of the retrieval design rather than of any one model's scale or access. The sharpest version of this point is that, on the quality of the extracted text, the small local model catches up to the large API model once both are allowed to direct their own search. The practical implication is that the extraction-quality benefit does not require a frontier model or a paid API, which lowers the barrier for teams that work under cost or data-privacy limits. The rest of this chapter traces the mechanism behind the gain, examines the error trade-off the iterative method introduces, and discusses what its use would mean in practice.

The accuracy gain extends to legal contracts despite two structural differences from the general-domain settings where iterative retrieval was first studied. IRCoT, FLARE, Iter-RetGen, and Self-RAG all showed gains from iterative retrieval, but they were tested on Wikipedia-based benchmarks where every question has a positive answer (Trivedi et al., 2023; Shao et al., 2023; Jiang et al., 2023; Asai et al., 2024).

Legal contracts differ in two ways that could reduce this advantage: most clause categories are absent rather than present, and the task is to extract exact text rather than generate a free-form answer. The gain still appears, which aligns with the finding of Y. Lin et al. (2025) that nearly half of traditional RAG failures happen because the right passage is never retrieved in the first place. The failure to retrieve the right passage is a retrieval problem, and it applies regardless of what the downstream task requires.

The Token F1 gains are larger than the Exact Match gains, and this gap shows what the iterative loop actually changes. Exact Match only checks whether the presence-or-absence verdict is right. Token F1 also measures how much of the correct clause text was retrieved. When F1 improves more than EM, as it does for both models, the model is finding more complete clause text, not just getting the verdict right more often. The larger F1 gain makes sense given how legal contracts are structured, as described in Section 2.2. A termination clause might refer to a notice period defined in a different section, and an assignment restriction might depend on conditions stated under a separate heading. A single retrieval pass cannot follow these cross-references. A ReAct loop can, because it can issue a follow-up query once it has located a pointer to the relevant section.

The small, steady gap between Token F1 and ROUGE-L points the same way. ROUGE-L stays a little below Token F1, by 0.03 to 0.05 in every case, which is the expected pattern for verbatim extraction. Token F1 gives credit for any shared tokens regardless of order, while ROUGE-L requires the same order, so a small gap means the extracted spans keep the original word order of the contract rather than reshuffling or paraphrasing it. Because both metrics rise together whenever the iterative method wins, the span gains are a real improvement in retrieval quality and not an artefact of one metric's formula.

The most important finding is the shift in how GPT-5-mini balances false positives and false negatives between the two conditions. In the single-pass condition, GPT-5-mini is strongly conservative: it correctly identifies most absent clauses but misses many clauses that are actually present. Hendrycks et al. (2021) note

that models trained on CUAD "typically learn to always output the empty span, since this is usually the correct answer" due to extreme class imbalance. The same pattern appears in the RAE condition at inference time: when the model does not find strong evidence that a clause exists, it defaults to reporting it as absent. The iterative condition changes this. By searching across multiple queries, the model collects enough evidence to commit to a positive answer in cases where a single pass found nothing. The cost is a drop in absent Exact Match. The drop in absent Exact Match is not a flaw in the architecture. FLARE's ablation study shows a similar effect in general-domain retrieval: retrieving too often can hurt performance because the extra passages introduce noise (Jiang et al., 2023). Here, the same effect appears as a shift in error types: more true positives but also more false positives, while overall accuracy still improves. Which condition to use should therefore depend on what type of error is more costly. In due diligence, where a missed clause can have real legal consequences, the higher recall on present clauses is likely worth the increase in false positives. In high-volume screening, where false positives create extra review work, the single-pass condition may be the better choice. The Gemma4 e4b results are worth noting here: it improves on both positive and absent instances without shifting toward more false positives, suggesting that smaller or more conservative models may suit use cases where false positives are costly.

Iterative retrieval is more expensive to run. The added cost is a recognised issue across the literature. IRCoT notes that its gains "come with an additional computational cost" because the framework makes a separate model call at each reasoning step (Trivedi et al., 2023). FLARE similarly observes that interleaving generation and retrieval "increases both overheads and the cost of generation" (Jiang et al., 2023). In a legal retrieval setting, L-MARS finds that deep search is roughly 40 times slower than shallow retrieval. (Z. Wang and Yuan, 2025). The accuracy gains are real but come with a cost in latency and API usage, and whether that trade-off is worth it depends on the use case. Self-RAG shows that tuning how often retrieval is triggered can keep most of the accuracy gain while reducing the number of retrieval calls (Asai et al., 2024). Whether a smaller retrieval budget would keep most of the gain for clause extraction is an open question this study does not settle.

The cross-model consistency is the clearest evidence that the gain comes from the retrieval design rather than from any particular model. Both models benefit from iterative retrieval despite being very different in size and architecture, which points to the gain coming from the retrieval design itself rather than from any particular model's capabilities. Watson et al. (2025) describe their agentic legal system as "applicable for both closed and open-source models," in contrast to fine-tuned systems tied to a specific architecture. The results here extend that observation: both a large closed API model and a small locally-run open-weight model benefit from the ReAct loop, without any legal-domain fine-tuning. The main requirement for capturing this gain appears to be the ability to produce reliable structured tool calls, a capability that is now available across a wide range of models.

8.1. Temporal and Societal Implications

The problem this thesis studies is current and pressing. Legal hallucination rates between 58% and 88% were measured only in 2024 (Dahl et al., 2024), the EU AI Act entered into force in 2024 (European Parliament and Council of the European Union, 2024), and most of the legal retrieval work cited here is from 2024 and 2025. The question of how to use language models safely on legal documents is being decided now, while the tools are already reaching practice, so the trade-offs below are not hypothetical.

The role of the lawyer. A system like the one studied here does not replace a lawyer. It changes what the lawyer spends time on. The verbatim-extraction design returns the actual clause text or a null, so the lawyer reads a specific passage and decides what it means, rather than reading a generated summary and trying to trust it. The model does the slow work of finding candidate text across a long contract, and the human keeps the judgment about what the text implies, a division between locating text and interpreting it that the legal NLP literature already draws (Ariai et al., 2024). This division is also what the EU AI Act asks for. Article 14 requires that a person can oversee and correct the system's output (European Parliament and Council of the European Union, 2024), and that is only possible when the output is a piece of the document the lawyer can check.

Benefits and risks. The main benefit is speed with traceability. Every answer points to text the lawyer can verify, and because the system can run on a small local model, a firm with privacy or budget limits can use it without sending contracts to an external API. The main risk is over-reliance. If a reviewer treats the system as a final answer rather than a starting point, the value of human oversight is lost. The two error types carry very different weight here, which is the next point.

Why a false positive is the safer error. The results show that the iterative method finds more real clauses but also raises false positives on absent clauses (Section 7.4). In the legal setting this trade is usually worth it. A false positive shows the lawyer a passage that turns out not to be the clause, which the lawyer can read and dismiss in seconds. A false negative tells the lawyer that no such clause exists, which invites them to stop looking. A missed liability cap or change-of-control term can have real legal consequences, while a wrongly flagged passage mostly costs a moment of review. For uses that touch the administration of justice or legal interpretation, which the EU AI Act places in its high-risk category (European Parliament and Council of the European Union, 2024), the system must support human judgment rather than quietly close off a line of inquiry. Showing the human something to check, even when it is wrong, fits that duty better than confidently reporting that nothing is there. This reframes the higher false-positive rate of the iterative method as a feature for high-stakes review, not only as a cost.

8.2. Limitations

Several limitations constrain what this study can and cannot conclude.

Single dataset. The evaluation uses only CUAD (Hendrycks et al., 2021), which consists of commercial contracts sourced from the SEC EDGAR system. These are heavily negotiated agreements between large, publicly traded companies and may not represent the full range of contracts encountered in legal practice. Employment agreements, consumer contracts, real estate leases, and non-disclosure agreements follow different structural conventions and use different vocabulary. Whether the findings generalise beyond EDGAR-type agreements is an open question that can only be answered by replication on other contract collections.

Sampled evaluation sets and API cost. The main experiment covers 5427 of the roughly 18000 QA pairs, stopped there because the local run already took about two weeks of compute. These pairs are a continuous slice of the dataset in contract order rather than a random sample, so although they span every clause category, they do not draw uniformly from across the whole pool, and results carry sampling uncertainty. The cross-model comparison is smaller still, a balanced 1000-pair set, mainly because GPT-5-mini is a paid API model and a larger run on it would have been costly. The cross-model conclusions therefore rest on less data than the main experiment, and all figures should be read as estimates rather than exact values.

No domain fine-tuning. Both the orchestrator and the extraction language model are general-purpose models with no legal-domain fine-tuning. The query-reformulation strategies the Iterative condition uses are therefore derived from general language patterns rather than legal vocabulary or contract structure. A model fine-tuned on contract language and cross-referencing patterns might produce larger or smaller gains than those observed here.

Tool-call reliability as a deployment constraint. The Iterative condition requires the language model to produce valid structured tool calls at each step of the retrieval loop. Both GPT-5-mini and Gemma4 e4b demonstrated reliable tool-call behaviour in this study, confirming that iterative retrieval is not limited to frontier API models. However, not all models handle structured tool-call schemas equally well. During development, attempts with other local open-weight models produced malformed JSON, ignored the tool schema, or returned free-text responses where a structured call was expected. Tool-call reliability therefore remains a meaningful deployment constraint: models that cannot consistently produce valid tool calls will fail in the iterative loop regardless of their general language quality.

Computational cost of the Iterative condition. Each run of the Iterative condition can issue up to 35 retrieval calls per question. Across 1000 evaluation pairs this makes the Iterative condition substantially more expensive in API calls and wall-clock time than the Retrieve-and-Extract condition, which issues a single retrieval call per question. In a production setting, cost and latency matter alongside accuracy. This study measures accuracy only and does not report cost or latency figures.

Evaluation metric strictness. Exact Match is a strict metric: a prediction that captures the correct clause but differs from the ground-truth annotation by even a single token is counted as wrong. CUAD annotations can include slight variations in span boundaries for the same underlying clause, and the dataset does not provide multiple acceptable spans (Hendrycks et al., 2021). EM may therefore undercount correct answers that differ only in how far the span extends before or after the clause text. Token-level F1 partially mitigates this by giving partial credit for overlapping spans, but it is not immune to span-boundary variation either.

Fixed retrieval parameters. The chunk size, overlap, and retrieval budget ($k = 5$) are held constant across both retrieval conditions. These parameters were chosen to enable a controlled comparison, but they may not be optimal for either condition. A larger k might allow the Retrieve-and-Extract condition to close some of the gap with the Iterative condition by giving the language model more context in its single pass. Conversely,

a smaller chunk size might better preserve clause boundaries and improve precision for both conditions. The effect of these choices is not isolated in this study.

Open-corpus difficulty layered on the comparison. The evaluation is open-corpus: every question is answered by searching all 510 contracts, and the question names only the contracting parties, so the system must first identify which contract is meant and then find the clause within it. This mirrors the realistic setting in which a practitioner knows the parties but retrieves from a collection of agreements rather than browsing a single file. It also adds a difficulty that the research question does not isolate, since both conditions must solve the contract-identification problem on top of the clause-extraction problem. The reported scores therefore reflect performance under this combined burden rather than clause extraction alone, and a single-document setup might show different absolute numbers for both conditions.

Cross-document comparison not evaluated. Although the evaluation is open-corpus and the system searches across all 510 contracts, each question targets a single contract and a single clause. The system is not asked to compare clauses across contracts, check consistency between an amendment and a master agreement, or aggregate information from multiple related documents. In practice, legal review often involves exactly these cross-document tasks, and the evaluation design cannot speak to them.

9

Conclusion

This thesis posed a single research question: what is the change in clause extraction accuracy when the language model directs retrieval, through LLM-orchestrated iterative retrieval, compared to single-pass retrieve-and-extract on legal contracts. To answer it, a controlled experiment was run on the CUAD benchmark (Hendrycks et al., 2021), comparing single-pass retrieve-and-extract against LLM-orchestrated iterative retrieval while keeping all other components fixed.

Letting the language model direct retrieval produces a measurable improvement in clause extraction accuracy. In the main experiment, the 5427-pair run on the locally hosted Gemma4 e4b, the iterative method improves overall Exact Match from 0.687 to 0.744 and token-level F1 from 0.341 to 0.505, and every gain is statistically significant under paired tests. The F1 gains are especially notable because they reflect more complete span retrieval, not just a shift in the presence-or-absence verdict, and ROUGE-L tracks F1 closely throughout, so the improvement is a real retrieval-quality gain rather than an artefact of the metric.

The gain is also not specific to one model. On the balanced cross-model set, GPT-5-mini improves from an overall Exact Match of 0.711 to 0.800 and Gemma4 e4b from 0.676 to 0.741, the latter matching the main experiment. Both a large closed API model and a compact locally-run open-weight model therefore benefit from the iterative method, and on span quality the small local model is statistically tied with the large API model. The fact that the gains are consistent across models suggests that the improvement comes from the retrieval design rather than from model scale, with the practical consequence that a deployment that cannot use a commercial API can still capture most of the benefit with a local model. The verbatim extraction formulation used throughout, returning the actual clause text or null rather than a generated answer, is also consistent with the human oversight requirements of the EU AI Act (European Parliament and Council of the European Union, 2024), since it gives the reviewing lawyer a specific piece of contract text to verify rather than a synthesized conclusion to accept or reject.

This thesis contributes a controlled gap study and a reproducible baseline for the legal retrieval domain. To the best of our knowledge, no prior work has directly compared single-pass and iterative LLM-directed retrieval for extractive clause identification on CUAD while holding all other components fixed. The results quantify that gap, support the use of iterative retrieval for this task, and provide a reproducible reference point for future work comparing retrieval architectures in the legal domain.

10

Future Work

Several directions remain open for future work.

First, the evaluation covers only EDGAR contracts from the CUAD dataset. These are heavily negotiated commercial agreements from publicly traded US companies and may not represent the full range of contract types found in legal practice. Testing the approach on other collections, such as employment agreements, consumer contracts, or real estate documents, would show whether the findings hold beyond EDGAR-type agreements.

Second, CUAD queries are answered within a single document, so the current evaluation does not test cross-document retrieval. In practice, contract review often involves comparing clauses across multiple agreements or checking amendments against a master agreement. Extending the approach to multi-document retrieval is a natural next step.

Third, the query-reformulation strategies in the iterative system rely on general-purpose models with no legal-domain fine-tuning. Fine-tuning the model on legal vocabulary or cross-referencing patterns could produce larger gains than those observed here.

Fourth, the iterative retrieval loop does not keep any memory across clause categories within the same contract. Information found while answering one category may be directly useful for another category in the same document. A memory mechanism that carries intermediate findings across categories could reduce redundant retrieval and make answers within a single contract more consistent.

Fifth, the iterative retrieval issues up to 35 search calls per question, compared to one call for the retrieve and extract case. The higher call count makes it substantially more expensive in tokens and wall-clock time. Future work could investigate whether a smaller retrieval budget, for example a maximum of five calls, retains most of the accuracy gain while reducing cost, or whether a larger budget is needed to achieve the observed performance. Such a cost-accuracy analysis would help practitioners decide how much of the iterative advantage is achievable at a given cost constraint.

Finally, Gemma4 e4b achieved competitive results at only 4 billion parameters. Future work could test even smaller models in the 1 to 3 billion parameter range to identify the minimum model size that can reliably produce valid tool calls and benefit from the iterative loop. A smaller model would further lower the barrier to deploying iterative retrieval in settings where GPU resources or API budgets are limited.

11

Code Availability and Use of AI

11.1. Code Availability

The full source code for this thesis is publicly available at `https://github.com/AsthenicDog390/master_thesis_public`. The repository contains the retrieval pipeline, the implementation of the baseline and the proposed method, the question-generation and evaluation scripts, and the statistical analysis, so that the experiments and results reported here can be reproduced.

11.2. Use of AI

This section declares how artificial intelligence tools were used in the course of this thesis.

In the experiments. The systems studied in this thesis are large language models, which are used in both the baseline and the proposed method (Chapters 5 and 6). AI coding assistants were also used to help write, debug, and improve the experimental code and evaluation scripts. All generated code was reviewed, tested, and validated. The experimental design, implementation, analysis, and interpretation of the results were carried out by me.

In writing. AI-based writing assistants were used to improve the grammar, clarity, and wording of the text. These tools were used only on text that I had already written, and all suggested changes were reviewed before being included. The ideas, methodology, analysis, and conclusions presented in this thesis are my own. AI tools were used only to help improve the writing and did not contribute to the research itself.

In the cover image. The image on the title page was generated with an OpenAI image-generation model. It is purely decorative and plays no part in the research, the methodology, or the results.

A

Additional Result Figures

This appendix collects the per-category figures that support the analysis in Chapter 7 but are not needed to follow the main argument. The per-category results for the main experiment are shown inline in Section 7.3; this appendix holds the same breakdown for the balanced cross-model set and a direct model-versus-model comparison. Each forest plot shows the change per clause category, with bars to the right favouring the iterative method (or, for the cross-model plots, the stronger model).

A.1. Per-Category Changes on the Balanced Set

Figures A.1 to A.3 show the iterative minus baseline change per category on the balanced 1000-pair set, one metric per figure with Gemma4 e4b on the left and GPT-5-mini on the right. The same broad pattern holds for both models, most categories improve and a small absent-heavy group regresses, which mirrors the main experiment in Section 7.3.

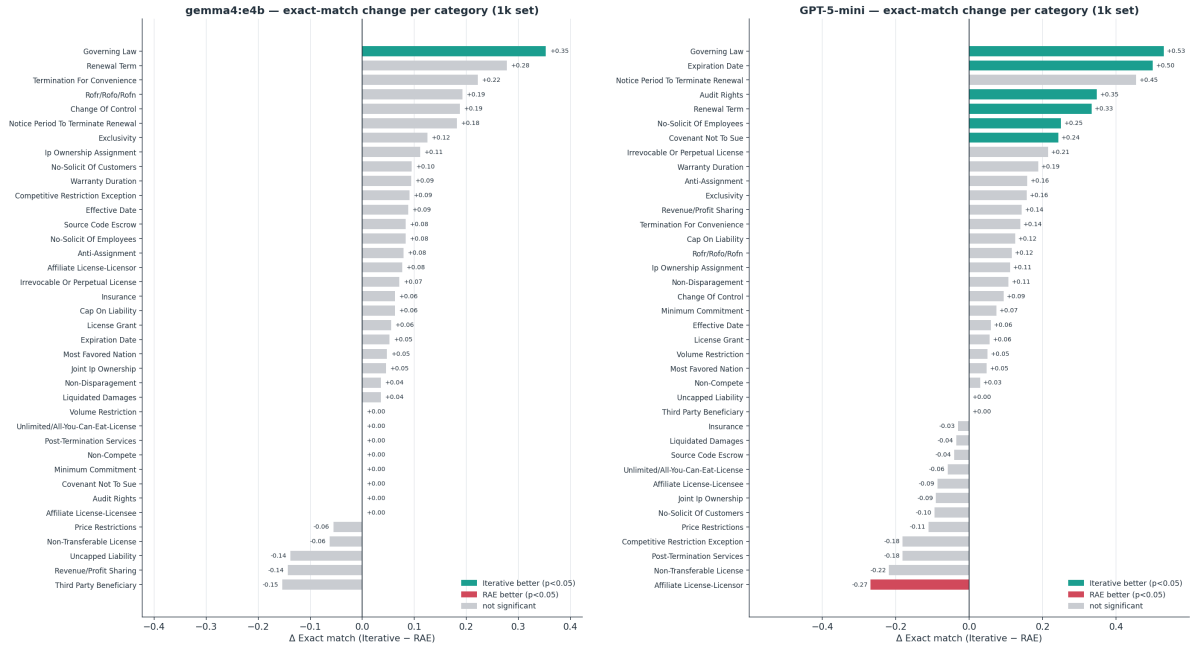

Figure A.1: Per-category change in Exact Match on the balanced 1000-pair set, iterative minus baseline. Gemma4 e4b (left) and GPT-5-mini (right). Bars to the right favour the iterative method.

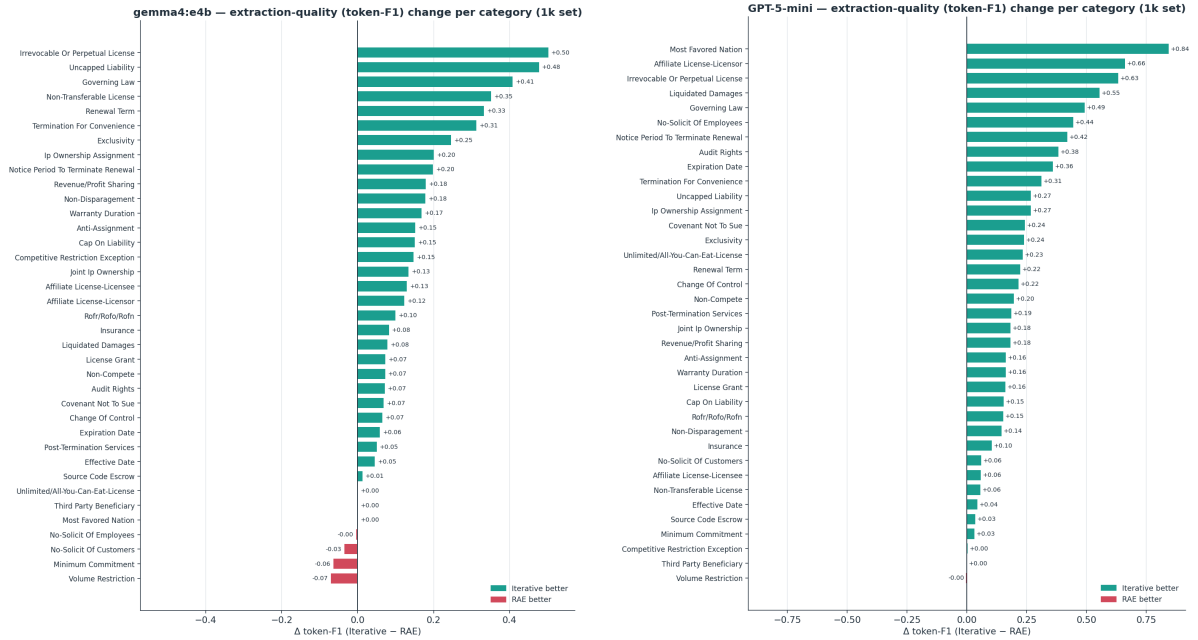

Figure A.2: Per-category change in Token F1 on the balanced 1000-pair set, iterative minus baseline. Gemma4 e4b (left) and GPT-5-mini (right). Bars to the right favour the iterative method.

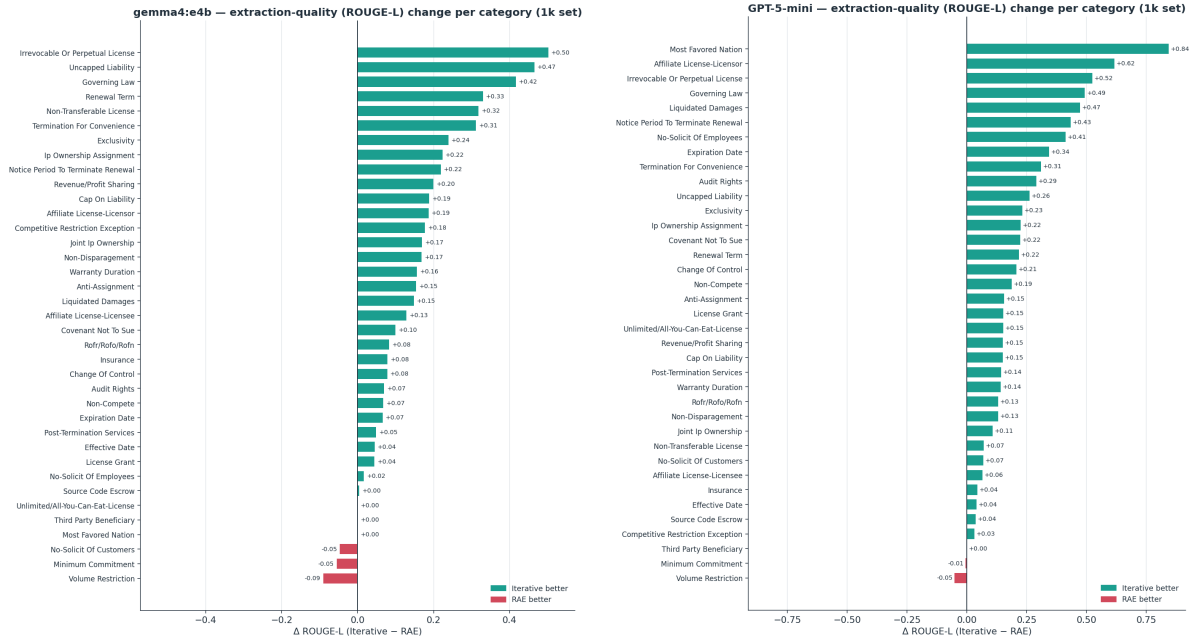

Figure A.3: Per-category change in ROUGE-L on the balanced 1000-pair set, iterative minus baseline. Gemma4 e4b (left) and GPT-5-mini (right). Bars to the right favour the iterative method.

A.2. Model-versus-Model Comparison

Figures A.4 and A.5 compare the two models directly under the iterative method on the balanced 1000-pair set, GPT-5-mini minus Gemma4 e4b per category. Bars near zero mean the two models behave alike on that category.

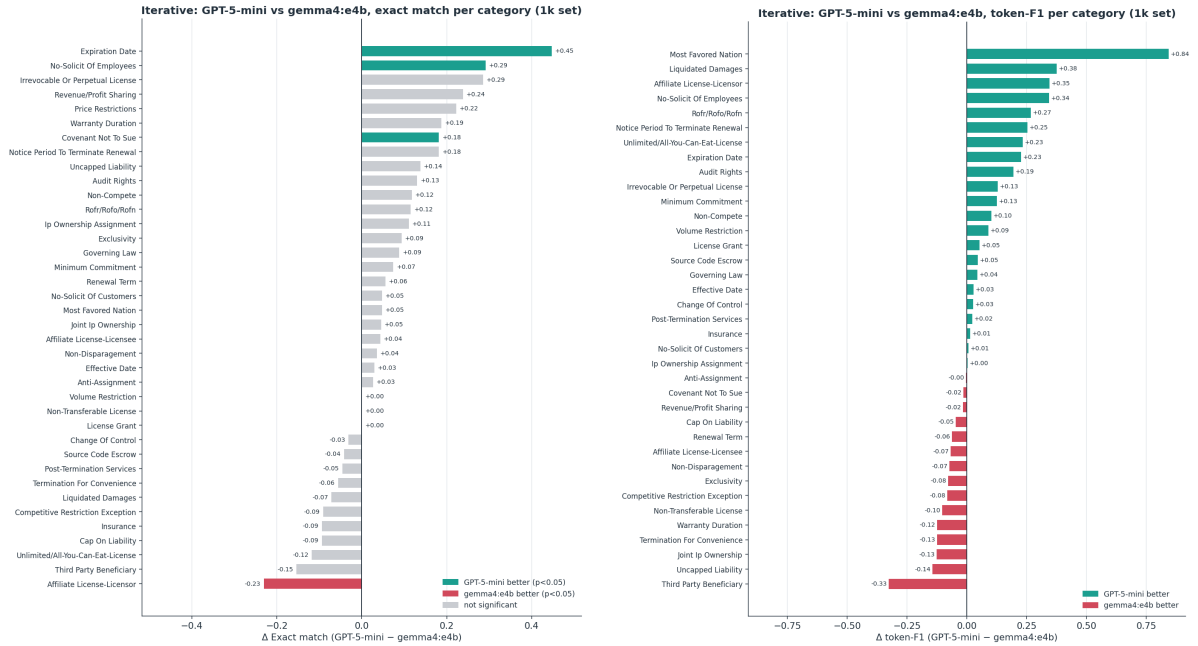

Figure A.4: GPT-5-mini minus Gemma4 e4b under the iterative method, per category, on the balanced 1000-pair set. Exact Match (left) and Token F1 (right). Bars to the right favour GPT-5-mini.

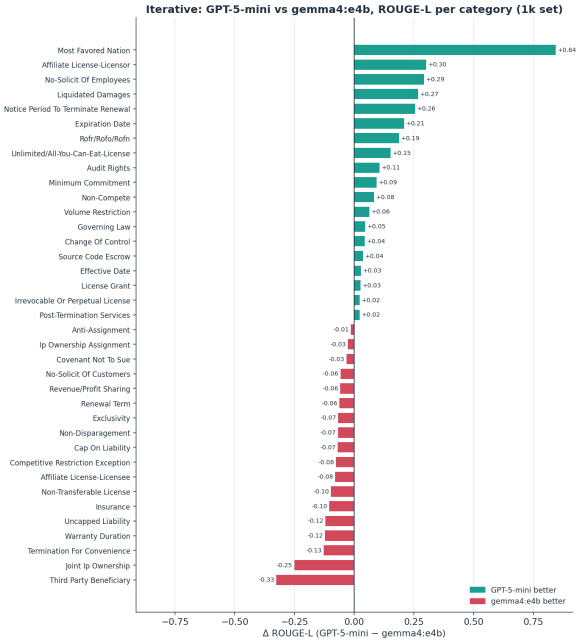

Figure A.5: GPT-5-mini minus Gemma4 e4b under the iterative method, per category, on the balanced 1000-pair set, for ROUGE-L. Bars to the right favour GPT-5-mini.

Bibliography

Ariai, Farid, Joel Mackenzie, and Gianluca Demartini (2024). "Natural Language Processing for the Legal Domain: A Survey of Tasks, Datasets, Models, and Challenges". In: *ACM Computing Surveys*. DOI: 10.1145/3777009.

Asai, Akari, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi (2024). "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection". In: *The Twelfth International Conference on Learning Representations*.

Braun, Christian, Alexander Lilienbeck, and Daniel Mentjukov (2025). *The Hidden Structure — Improving Legal Document Understanding Through Explicit Text Formatting*. Preprint. arXiv: 2505.12837.

Clark, Peter, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord (2018). "Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge". In: *arXiv preprint arXiv:1803.05457*.

Cormack, Gordon V., Charles L. A. Clarke, and Stefan Buettcher (2009). "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pp. 758–759. DOI: 10.1145/1571941.1572114.

Dahl, Matthew, Varun Magesh, Mirac Suzgun, and Daniel E. Ho (2024). "Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models". In: *Journal of Legal Analysis* 16.1, pp. 64–93. DOI: 10.1093/jla/laae003.

Efron, Bradley (1979). "Bootstrap Methods: Another Look at the Jackknife". In: *The Annals of Statistics* 7.1, pp. 1–26. DOI: 10.1214/aos/1176344552.

European Parliament and Council of the European Union (2024). *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, OJ L 2024/1689. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng.

Ferguson, James, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi (2020). "IIRC: A Dataset of Incomplete Information Reading Comprehension Questions". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1137–1147. DOI: 10.18653/v1/2020.emnlp-main.86.

Geva, Mor, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant (2021). "Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies". In: *Transactions of the Association for Computational Linguistics* 9, pp. 346–361. DOI: 10.1162/tacl_a_00370.

Guha, Neel, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, et al. (2023). "LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc.

Hayashi, Hiroaki, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig (2021). "WikiAsp: A Dataset for Multi-domain Aspect-Based Summarization". In: *Transactions of the Association for Computational Linguistics* 9, pp. 211–225. DOI: 10.1162/tacl_a_00362.

Hendrycks, Dan, Collin Burns, Anya Chen, and Spencer Ball (2021). "CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc.

Ho, Xanh, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa (2020). "Constructing A Multi-Hop QA Dataset for Comprehensive Evaluation of Reasoning Steps". In: *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, pp. 6609–6625. DOI: 10.18653/v1/2020.coling-main.580.

Hui, Yulong, Chao Chen, Zhihang Fu, Yihao Liu, Jieping Ye, and Huanchen Zhang (2025). *Interact-RAG: Reason and Interact with the Corpus, Beyond Black-Box Retrieval*. Preprint. arXiv: 2510.27566.

Jiang, Zhengbao, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, et al. (2023). "Active Retrieval Augmented Generation". In: *Proceedings of the 2023 Conference on Empirical*

Methods in Natural Language Processing. Association for Computational Linguistics, pp. 7969–7992. DOI: 10.18653/v1/2023.emnlp-main.495.

Joshi, Mandar, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer (2017). "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1601–1611. DOI: 10.18653/v1/P17-1147.

Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih (2020). "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550.

Koreeda, Yuta and Christopher D. Manning (2021). "ContractNLI: A Dataset for Document-level Natural Language Inference for Contracts". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, pp. 1907–1919. DOI: 10.18653/v1/2021.findings-emnlp.164.

Lai, Jinqi, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S. Yu (2024). "Large Language Models in Law: A Survey". In: *AI Open* 5.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 9459–9474.

Lin, Chin-Yew (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Association for Computational Linguistics, pp. 74–81. URL: https://aclanthology.org/W04-1013.

Lin, Yuncheng, Prasad Tadepalli, and Stefan Lee (2025). *Fishing for Answers: Exploring One-Shot vs. Iterative Retrieval Strategies for RAG*. Preprint. arXiv: 2509.04820.

Louis, Antoine, Gijs Van Dijck, and Gerasimos Spanakis (2024). "Interpretable Long-Form Legal Question Answering with Retrieval-Augmented Large Language Models". In: *Proceedings of the 38th AAAI Conference on Artificial Intelligence*. AAAI Press. URL: https://arxiv.org/abs/2309.17050.

Mallen, Alex, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi (2023). "When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 9802–9822. DOI: 10.18653/v1/2023.acl-long.546.

McNemar, Quinn (1947). "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages". In: *Psychometrika* 12.2, pp. 153–157. DOI: 10.1007/BF02295996.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135.

Pipitone, Nicholas and Ghita Houir Alami (2024). *LegalBench-RAG: A Benchmark for Retrieval-Augmented Generation in the Legal Domain*. Preprint. arXiv: 2408.10343.

Press, Ofir, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis (2023). "Measuring and Narrowing the Compositionality Gap in Language Models". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, pp. 5687–5711. DOI: 10.18653/v1/2023.findings-emnlp.378.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang (2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2383–2391. DOI: 10.18653/v1/D16-0264.

Shao, Zhihong, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen (2023). "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, pp. 9248–9274. DOI: 10.18653/v1/2023.findings-emnlp.620.

Siino, Marco, Mariana Falco, Daniele Croce, and Paolo Rosso (2025a). "Exploring LLMs Applications in Law: A Literature Review on Current Legal NLP Approaches". In: *IEEE Access* 13. DOI: 10.1109/ACCESS.2025.3533217.

— (2025b). "Exploring the Use of LLMs in the Italian Legal Domain: A Survey on Recent Applications". In: *Computer Law & Security Review* 58, p. 106164. DOI: 10.1016/j.clsr.2025.106164.

Stelmakh, Ivan, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang (2022). "ASQA: Factoid Questions Meet Long-Form Answers". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 8273–8288. DOI: 10.18653/v1/2022.emnlp-main.566.

Topollai, Kristi, Tolga Dimlioglu, Anna Choromanska, Simon Odie, and Reginald Hui (2025). *Streamlining Industrial Contract Management with Retrieval-Augmented LLMs*. Preprint. arXiv: 2511.14671.

Trivedi, Harsh, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal (2022). "MuSiQue: Multihop Questions via Single-hop Question Composition". In: *Transactions of the Association for Computational Linguistics* 10, pp. 539–554. DOI: 10.1162/tacl_a_00475.

— (2023). "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 10014–10037. DOI: 10.18653/v1/2023.acl-long.557.

Wang, Steven, Kosta Kuchta, Jonathan Hofer, Benjamin Wanger, and Julian Nyarko (2025). "ACORD: An Expert-Annotated Retrieval Dataset for Legal Contract Drafting". In: *Findings of the Association for Computational Linguistics: ACL 2025*. Association for Computational Linguistics.

Wang, Steven H., Antoine Scardigli, Leonard Tang, Wei Chen, Dmitry Levkin, Anya Chen, Spencer Ball, Thomas Woodside, et al. (2023). "MAUD: An Expert-Annotated Legal NLP Dataset for Merger Agreement Understanding". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Wang, Ziqi and Boqin Yuan (2025). *L-MARS: Legal Multi-Agent Workflow with Orchestrated Reasoning and Agentic Search*. Preprint. arXiv: 2509.00761.

Watson, Oliver, Cynthia Anggoro, and Stuart Aitken (2025). "LAW: Legal Agentic Workflows for Custody and Fund Services Contracts". In: *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*. Association for Computational Linguistics, pp. 594–606. URL: https://aclanthology.org/2025.coling-industry.50.

Wilcoxon, Frank (1945). "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6, pp. 80–83. DOI: 10.2307/3001968.

Xu, Ziwei, Sanjay Jain, and Mohan Kankanhalli (2025). *Hallucination is Inevitable: An Innate Limitation of Large Language Models*. Preprint. arXiv: 2401.11817.

Yang, Zhiwei, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning (2018). "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2369–2380. DOI: 10.18653/v1/D18-1259.

Yao, Shunyu, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao (2023). "ReAct: Synergizing Reasoning and Acting in Language Models". In: *The Eleventh International Conference on Learning Representations*.