



Master Thesis

Master of Science in Engineering

School of Engineering - Centre for Artificial Intelligence

On the Limits of RAG Evaluation: Modeling Error and Uncertainty Propagation in Retrieval Augmented Generation

Author

Felix Matthias Saaro

Supervisor

Prof. Dr. Mark Cieliebak

Secondary Supervisors

Dr. Jan Milan Deriu

Pius von Däniken

July 31th, 2026

The proof of the pudding is in the eating.

Abstract

Retrieval-Augmented Generation (RAG) offers a simple and quick way to combine search with generation, but is difficult to evaluate reliably. In this project we set out to understand why. We introduce two complementary models: a *System* model that expresses a RAG pipeline as a joint distribution over query, retrieval, and generation success, and a *Behavior* model that distinguishes task success, whether the system produced a correct answer, from generator success, whether the system behaved as expected, including abstaining when retrieval fails. Building on these models, we propose a simulation-based approach to study how errors and uncertainty in individual component estimates propagate through the pipeline to the system level outcome.

We validate this framework empirically across 27 RAG configurations, combining three retrievers (sparse, dense, hybrid) and three generators. These configurations are evaluated on four QA and fact-verification datasets (FEVER, NQ, HotpotQA, C4). We run a series of targeted experiments, including a probabilistic retriever for isolating retrieval success from generation, oracle and context-manipulation studies, and a comparison of automated LLM-based judges. These experiments show that evaluation results obtained under one combination of dataset, metric, retriever, and generator do not generalize to another. This holds even when a metric such as retrieval success is held fixed. This occurs because retrieval and generation are coupled through abstention behavior. An improvement in retrieval can, in some cases, lower overall system success, because the generator behaves differently on the new set of retrieval failures it produces. We further show that uncalibrated LLM-as-judge evaluation introduces an additional source of disagreement between judges. This evaluation approach is commonly used where reference-based metrics are unavailable, and the disagreement compounds the uncertainty already present in the non-generalization result.

RAG systems are very difficult to evaluate, not for lack of available metrics, but because the metrics that exist describe individual components in isolation while the behavior that matters emerges from their interaction. The framework proposed in this project offers a starting point for making that interaction explicit.

Preface

In 2023, while completing my Bachelor's degree, I worked at a company that had built a large internal documentation platform. Part of this project was a chatbot meant to help users find information faster. This was where I first encountered RAG systems. A RAG system requires a solid knowledge base, and since our documentation platform provided exactly that, I decided to give it a try. The implementation was straightforward, but I quickly noticed something was missing: I had no grounds to decide which components to use or which to improve, and no way to measure whether a change made the system better or worse.

Throughout my Master's, I encountered the same problem of measuring outcomes across several projects. In the VT2, I investigated how conditional generation tasks could be automatically evaluated using calibrated classifiers, modeling two sources of uncertainty: the generative model and the classifier itself. This raised the question of how the same technique could be extended to systems with additional sources of uncertainty. This thesis is the result of pursuing that question, returning to the project from three years earlier to find an answer to how a RAG system can actually be evaluated.

This work would not have been possible without the dedication and support of Jan Deriu and Pius von Däniken, who introduced me to the problem of evaluation and spent countless hours in our weekly discussions.

Felix Saaro
Winterthur, July 31st 2026

Glossary

Apt Apertus 8B. 23–25, 27–30

C4 Catechism of the Catholic Church. i, iv, v, 12–14, 16, 51–53, 55, 58, 59, 62, 63

FEVER Fact Extraction and VERification. i, iv, 12, 14, 24–45, 47, 51, 57–59, 62, 63

Gem Gemma3 12B. 23–25, 27–29, 31

GPT GPT 5.4-mini. 53, 54

HotpotQA HotpotQA. i, iv, v, 12–14, 24–30, 32–43, 46–51, 58, 59, 62, 63

IR Information Retrieval. 2

KILT Knowledge Intensive Language Tasks. 12

LLM Large Language Model. i, 2, 3, 16, 23, 24, 51–53, 55, 58–61

MAP Mean Average Precision. 2

MRR Mean Reciprocal Rank. 2

NQ Natural Questions. i, iv, 12–14, 24–30, 32–45, 47, 51, 58, 59, 62, 63

OI Oracle Intersection. 36, 37, 41, 62

PR Probabilistic Retriever. 31–33, 62

QA Question Answer. i, 1, 2, 12, 14, 58, 60

Qwn Qwen 3.5 9B. 23–25, 27–30, 33, 36, 39, 41, 44, 46, 48, 51, 53–55, 62

RAG Retrieval Augmented Generation. i, ii, iv, 1–5, 8–10, 13, 15, 20, 24, 25, 27–29, 57–63

RRF Reciprocal Rank Fusion. 19, 22

Contents

1	Introduction	1
1.1	Related Work	2
2	Methods	3
2.1	Retrieval Augmented Generation	3
2.2	Theoretical Models	5
2.2.1	Definitions	5
2.2.2	System Model	6
2.2.3	Behavior Model	7
2.3	Simulation	9
2.3.1	Error Propagation	9
2.3.2	Uncertainty Propagation	10
2.4	Datasets	12
2.4.1	FEVER	12
2.4.2	NQ	12
2.4.3	HotpotQA	13
2.4.4	C4	13
2.5	Metrics	15
2.5.1	Retrieval Quality	15
2.5.2	Task Quality	15
2.5.3	Generator Quality	17
2.5.4	Lexical Overlap	17
2.6	Retrieval	18
2.6.1	Sparse	18
2.6.2	Dense	18

2.6.3	Hybrid	19
2.7	Implementation	20
2.7.1	Pipeline Orchestration	20
2.7.2	Data Model	21
2.7.3	Knowledge Base	21
2.7.4	Retriever	22
2.7.5	Generator	22
2.7.6	Judge	23
3	Results	24
3.1	System Model Evaluation	24
3.2	Behavior Model Evaluation	27
3.3	Probabilistic Retriever	31
3.4	Probabilistic Approximated	33
3.5	Oracle Intersection	36
3.6	Compare Context Size	39
3.7	Compare Context Switch	41
3.8	Compare Context Shuffle	44
3.9	Compare Context Randomize	46
3.10	HotpotQA Partial Retrieval Success	48
3.11	C4 Retrieval Recall	51
3.12	Compare Generator and Judges	53
3.13	Lexical Overlap	55
4	Discussion	57
4.1	Limitations	58
4.2	Future Work	60
4.3	Conclusion	61

1 Introduction

Evaluating Retrieval Augmented Generation (RAG) systems is challenging and often tedious. It is not sufficient to evaluate only the system’s end-to-end behavior. Each component must be evaluated both in isolation and within the pipeline. Since RAG systems are built in a pipelined approach, errors tend to propagate through the pipeline. If retrieval fails, the generator has no basis for producing a correct answer and should abstain from answering the question. Most benchmarks developed for RAG evaluation consider each component in isolation [1, 2, 3], or evaluate generators using adversarial retrieval results [4]. However, a holistic view of the evaluation that accounts for the dependencies between components is still missing.

A key shortcoming of end-to-end evaluation is that it conflates two distinct notions of success. Task success measures whether the user received a correct answer, whereas generator success measures whether the generator behaved appropriately given the retrieval outcome, i.e., answering when retrieval succeeded and abstaining when it failed. These two quantities can diverge substantially. A generator that never abstains may achieve reasonable task success while systematically violating the desired policy, and two systems with identical task success can exhibit very different behaviors under retrieval failure. To make such distinctions explicit, we need a model that captures the dependencies between retrieval, abstention, and answer correctness.

In this project, we address this gap in two steps. First, we introduce a *System* model and a *Behavior* model, which formalize a RAG pipeline as a joint distribution over query, retrieval, and generation success, and which make the distinction between task success and generator success explicit. We extend these models with a simulation-based approach to study how errors and uncertainty in individual component estimates propagate to the system-level outcome, supported by an interactive tool for exploring this propagation. Second, we validate this framework empirically, implementing a configurable RAG pipeline and evaluating it across 27 configurations, combining three retrievers and three generators, on four QA datasets.

The results of this validation point to a finding that motivates the framework itself: evaluation results obtained under one combination of dataset, metric, retriever, and generator do not generalize to another, even when a metric such as retrieval success is held fixed. We trace this to a coupling between retrieval and generation through abstention behavior, and show that this coupling, together with the reliability of the metrics used to measure it, limits how far any single RAG evaluation result can be trusted beyond the exact setup it was measured under.

The remainder of this thesis is structured as follows. Section 1.1 reviews existing approaches to RAG evaluation and situates our contribution relative to them. Section 2 introduces the *System* and *Behavior* models, the simulation-based approach to error and uncertainty propagation, the datasets and metrics used, and the implementation of the RAG pipeline. Section 3 presents the empirical validation of this framework across a series of experiments, from a naive end-to-end evaluation to targeted studies isolating the effect of retrieval success, context composition, and judge reliability on system behavior. Section 4 discusses these results, and Section 4.3 concludes with practical takeaways, limitations, and directions for future work.

1.1 Related Work

RAG is the process of incorporating the output of an Information Retrieval (IR) system into the context of a Large Language Model (LLM), so that the LLM’s answer is informed by relevant documents or passages [5, 6]. The process is composed of two components: retrieval and generation. The user’s question is transformed into a query for an IR engine, which retrieves a set of relevant documents or passages, which are then given to the LLM to generate an answer to the user’s query. This setting, while straightforward to describe, is highly challenging to evaluate. In general, the two components are evaluated separately. For an extensive overview of RAG evaluation, we refer the reader to [7].

Retrieval Evaluation

There are two main settings: either there is a gold standard available, that is, for a set of queries, all relevant documents, or there is no gold standard available. In the case of an existing gold standard, which is highly costly to create, one applies the standard scores used in the IR literature [8], such as Mean Reciprocal Rank Mean Reciprocal Rank (MRR), or Mean Average Precision (MAP) [9]. In cases where no gold standard is available, two approaches are commonly used: either silver-standard generation or an LLM-as-a-judge [10]. The generation of silver standards consists of an inverted process by providing an LLM with a document or passage, and letting the LLM generate questions [1]. Then the provided document serves as the relevant document (while disregarding other potentially relevant documents). For LLM-as-a-judge, one provides the LLM with the result of the retrieval and the user question, and asks to rate whether the question can be answered with the provided retrieval [2].

Generator Evaluation

The evaluation of the generator often entangles two concepts: the end-to-end success, and the evaluation of the generator under different retrieval settings. For instance, [4] evaluates the behavior of the generator under different adversarial retrieval settings. [11] also highlights the importance of evaluating the impact of the retrieval on the generation in isolation. They also state the need to handle cases where the retrieval does not find relevant sources in the generator by allowing abstention from answering, and, in turn, evaluating the LLM’s behavior in these cases.

Pipeline Evaluation

[3] introduces a retrieval benchmarking library that focuses on evaluating pipeline components. Ares [2] introduces a LLM-as-a-judge-based evaluation framework to evaluate each component of the pipeline separately using various dimensions. RAGAs [1] introduces a reference-free framework for evaluating multiple dimensions of RAG pipelines without relying on ground-truth human annotations, covering context relevance, faithfulness, and answer relevance. CRUD-RAG [12] is a benchmark oriented towards real-world scenarios rather than the Question Answer (QA) tasks popular in academic settings. They include evaluation of multiple components, such as chunk size selection and reranking. While these frameworks evaluate multiple dimensions, they treat each dimension independently and do not model the statistical dependencies between retrieval and generation outcomes.

2 Methods

This section describes the methods used to model, simulate, and empirically evaluate RAG systems. We first describe the general structure of a RAG pipeline (Section 2.1), before introducing the *System* and *Behavior* models used to formalize it as a joint distribution over component successes (Section 2.2). Section 2.3 extends these models with a simulation-based approach for studying how errors and uncertainty in individual component estimates propagate to the system-level outcome. We then describe the datasets (Section 2.4) and metrics (Section 2.5) used to instantiate the models empirically, the retrieval strategies compared throughout this thesis (Section 2.6), and finally the implementation of the RAG pipeline used to run all experiments in Section 3 (Section 2.7).

2.1 Retrieval Augmented Generation

In the original RAG implementation [5], the pipeline consisted of a non-parametric retriever R and a pretrained seq2seq generator G . The retriever itself was composed of a query encoder and a document index, a design that would today be classified as a dense retriever. Since then, numerous variants of the original RAG architecture have emerged, including Graph RAG and Agentic RAG. Despite differing in implementation, these variants share the same three basic building blocks: (1) the *Query* Q , (2) a *Retriever* R , and (3) a *Generator* G .

Figure 1 illustrates the data flow of a RAG system. A user submits a question, referred to as the query, which is passed to the retrieval component. The retriever uses this query to search the knowledge base index for relevant paragraphs, applying either lexical or semantic matching depending on the retrieval strategy in use. The retrieved paragraphs are then ranked by a reranker, and the top k paragraphs are forwarded to the generator.

The generator combines these paragraphs with the original query, or in some cases a rewritten version of it, along with a system prompt specifying the desired behavior. This combined input is passed to an LLM, which produces the final answer presented to the user.

Numerous adaptations of this basic design exist to address different use cases or to improve performance, such as query rewriting, incorporating web search into the retrieval step, or routing queries to different LLMs based on question difficulty. Regardless of these variations, all such systems rely on the same underlying principle: a retrieval component and a generator component working in close coordination with one another.

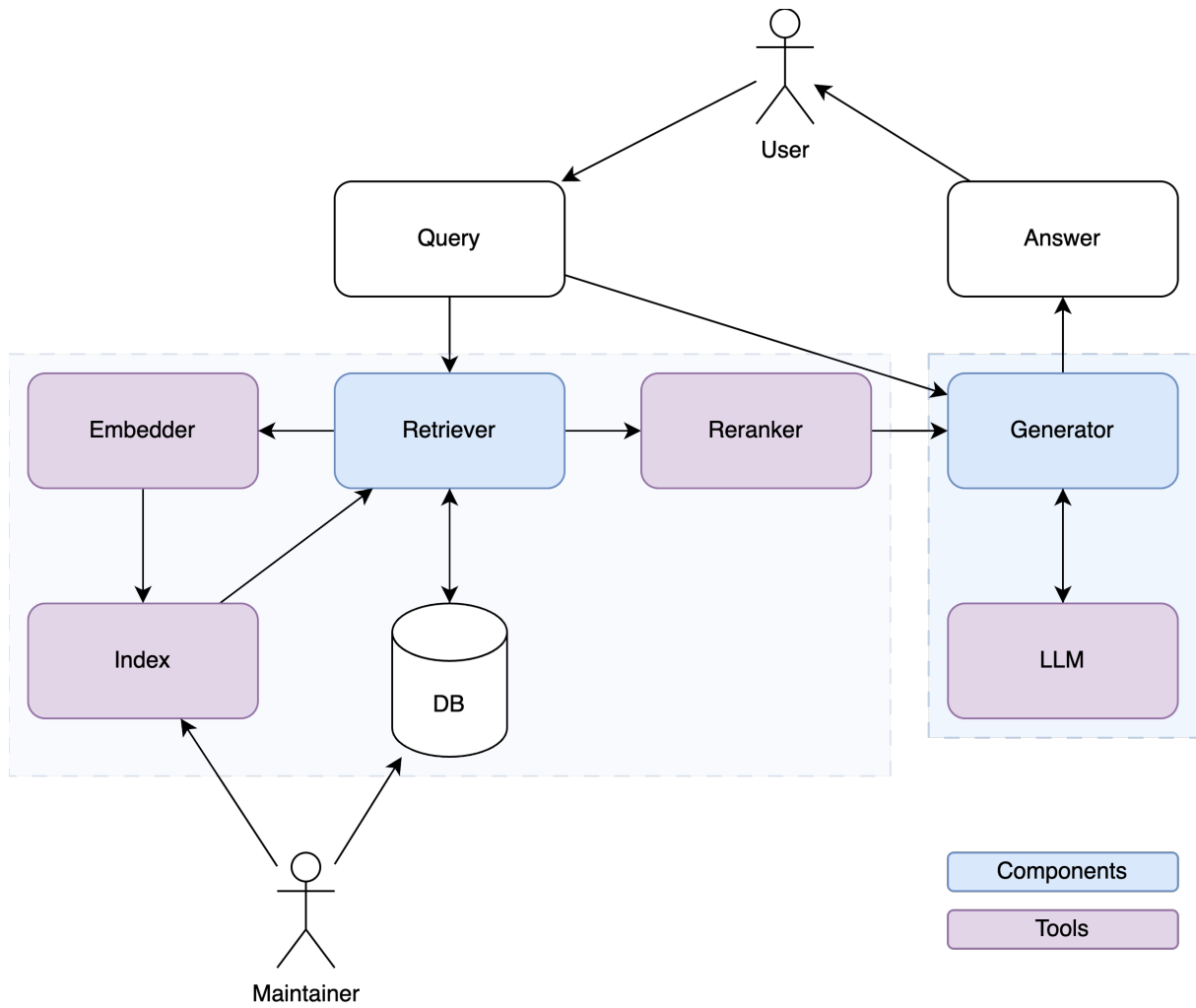


Figure 1: Overview of a RAG system including the two main components *Retriever* and *Generator* that coordinate the data flow within the system.

2.2 Theoretical Models

To reason about RAG systems in a structured way, it is useful to abstract away from any concrete implementation and instead describe the system in terms of the success or failure of its individual components. Such a formalization allows us to study how the performance of the retriever and generator jointly determine the overall system performance, and to identify which component contributes most to a failure. In the following, we introduce two complementary models: a *System* model, which captures the system as a joint probability distribution over its components, and a *Behavior* model, which captures the possible states the system can transition through, including abstention.

2.2.1 Definitions

We define the following components, each treated as a binary random variable indicating success (1) or failure (0):

- q : a given query
- r : the retriever paragraphs
- g : the generator answer

Query

The probability that a query is well-formed, or *good*, is denoted $P(q = 1)$. Conversely, the probability that a query is *bad* is $P(q = 0) = 1 - P(q = 1)$.

Retriever

The probability that the retriever succeeds given a good query is:

$$P(r = 1 \mid q = 1) = \frac{P(q = 1 \mid r = 1) \cdot P(r = 1)}{P(q = 1)}$$

Analogously, the probability that the retriever succeeds given a bad query is:

$$P(r = 1 \mid q = 0) = \frac{P(q = 0 \mid r = 1) \cdot P(r = 1)}{P(q = 0)}$$

Generator

The probability that the generator produces a correct answer given a good query and successful retrieval is:

$$P(g = 1 \mid q = 1, r = 1)$$

These conditional probabilities form the basis for both models introduced below.

2.2.2 System Model

In this thesis, we are primarily interested in the interplay between the query Q , the retriever R , and the generator G . We therefore simplify the system to these three components, as shown in Figure 2.

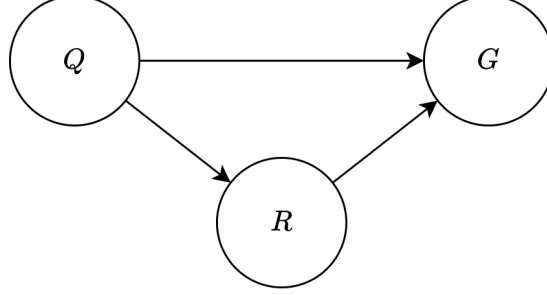


Figure 2: The *System* model captures how information flows through the system: the query Q is passed to the generator G together with the output of the retriever R .

This simplification allows us to express the system’s overall success probability as a joint distribution:

$$P(Q, R, G) = P(Q) \cdot P(R | Q) \cdot P(G | Q, R) \quad (1)$$

The model can then be used to assess how the marginal probabilities of retriever success $P(R)$ and generator success $P(G)$ are shaped by the underlying conditional probabilities.

As an illustrative example, consider the following observed conditional probabilities:

$$\begin{aligned}
 P(q = 1) &= 0.95 \\
 P(r = 1 | q = 1) &= 0.70 \\
 P(r = 1 | q = 0) &= 0 \\
 P(g = 1 | q = 1, r = 1) &= 0.90 \\
 P(g = 1 | q = 1, r = 0) &= 0.50 \\
 P(g = 1 | q = 0, r = 1) &= 0.50 \\
 P(g = 1 | q = 0, r = 0) &= 0.0
 \end{aligned} \quad (2)$$

From these, the marginal probabilities can be computed as follows:

$$\begin{aligned}
P(r = 1) &= P(q = 1) \cdot P(r = 1 \mid q = 1) + P(q = 0) \cdot P(r = 1 \mid q = 0) \\
&= 0.95 \cdot 0.70 + 0.05 \cdot 0 = 0.665 \\
P(g = 1) &= \sum_{q,r} P(g = 1 \mid q, r) \cdot P(q, r) \\
&= P(q=1)P(r=1|q=1)P(g=1|q=1, r=1) \\
&\quad + P(q=1)P(r=0|q=1)P(g=1|q=1, r=0) \\
&\quad + P(q=0)P(r=1|q=0)P(g=1|q=0, r=1) \\
&\quad + P(q=0)P(r=0|q=0)P(g=1|q=0, r=0) \\
&= (0.95 \cdot 0.70 \cdot 0.90) + (0.95 \cdot 0.30 \cdot 0.50) \\
&\quad + (0.05 \cdot 0.00 \cdot 0.50) + (0.05 \cdot 1.00 \cdot 0.00) \\
&= 0.5985 + 0.1425 + 0 + 0 = 0.741
\end{aligned} \tag{3}$$

Beyond illustrating the calculation, this model can be used to estimate component-wise performance from plausible conditional probabilities, or to test how an improvement in the retriever propagates to generator performance. It is worth noting that some conditional probabilities are considerably easier to obtain empirically than others, particularly those corresponding to rare event combinations.

2.2.3 Behavior Model

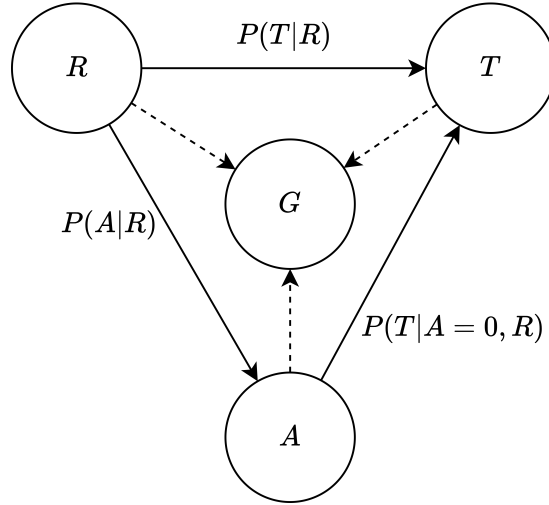


Figure 3: The *Behavior* model shows the possible behavioral states of the system, with R denoting retrieval success, A abstention, T task success, and G generator success.

While the System Model captures the overall success probability, the Behavior Model instead describes how the system behaves given a set of observed conditionals, shown in Figure 3. Conceptually, this can be understood as a stochastic state machine in which each component (R , A , T , G) takes a value of 0 or 1, and transitions between states are governed by conditional probabilities. For instance, the transition from abstention to task success is given by $P(T \mid A = 0, R)$.

Here, R denotes retrieval success, A denotes abstention, T denotes task success, and G denotes generator success. Task success indicates whether a given answer is correct, while generator success indicates whether the system behaved as expected: if retrieval fails, it is preferable for the system to abstain, which results in task failure but is nonetheless the desired outcome from the user’s perspective. As a consequence, generator success is a deterministic function of the other variables:

$$G = [(R = 0) \wedge (A = 1)] \vee [(R = 1) \wedge (A = 0) \wedge (T = 1)] \quad (4)$$

That is, generator success occurs if and only if either (1) retrieval fails and the generator abstains from answering, or (2) retrieval succeeds, the generator does not abstain, and it produces the correct answer.

The remaining components are modeled as:

$$P(R, A, T) = P(R) \cdot P(A \mid R) \cdot P(T \mid A, R) \quad (5)$$

subject to the constraint that task success cannot be achieved if the generator abstained from answering: if $A = 0$, then $T = 1$.

Both the *System* and *Behavior* models rely on the simplifying assumption that component success can be measured as a binary variable. Despite their simplicity, with only three to four nodes each, they provide a compact framework for reasoning about failure propagation in RAG systems.

2.3 Simulation

The *System* and *Behavior* models introduced above rely on point estimates of conditional probabilities such as $P(r = 1 \mid q = 1)$ or $P(t = 1 \mid a = 0, r = 1)$. In practice, these estimates are derived from a number of observations and are therefore themselves subject to error and uncertainty.

To investigate this, we conduct two related but distinct analyses: first, we assess how errors in the individual component-level estimates bias the overall system estimate (*error propagation*), using the explicit conditional probability values obtained from the evaluation set; second, we extend this analysis by placing *Beta distributions* as prior distributions over each conditional probability and propagating them through both models using Monte Carlo sampling, to observe how much uncertainty about the true component probabilities translates into uncertainty about the overall system outcome (*uncertainty propagation*).

2.3.1 Error Propagation

Error propagation is concerned with how a bias or inaccuracy in a single conditional probability estimate affects the point estimate of the overall system output, e.g. $P(G)$ or $P(T)$. For this analysis, each conditional probability is set to the explicit point estimate observed in the evaluation set, e.g. the proportion of samples with a good query for which retrieval succeeded, without modeling it as a distribution.

To assess error propagation, we systematically perturb one conditional probability at a time, while holding all others fixed at their observed point estimate, and evaluate the corresponding model equation (Equation 1 for the System Model, or Equations 4 and 5 for the Behavior Model) for each perturbed value. Comparing the resulting output to the output obtained using the original, unperturbed point estimates indicates how sensitive the overall system estimate is to errors in individual components: a component whose perturbation causes a large shift in the model output is one whose estimation error propagates strongly through the system, while a component with little effect on the output is comparatively robust to estimation error.

This sensitivity analysis makes it possible to identify which components of the RAG system, retriever or generator, most strongly determine the reliability of the overall system estimate, and therefore which conditional probabilities warrant the most careful empirical measurement.

To make this sensitivity analysis more accessible, an interactive tool ¹ was built, shown in Figure 4. This allows any acyclic system pipeline to be replicated by connecting components and specifying their conditional probabilities. The tool then automatically computes the resulting marginal probability of each component. Figure 4 shows this applied to the RAG system introduced in section 2.2.2, with the Query, Retriever, and Generator components and their associated conditional probabilities. By adjusting the conditionals, the marginals are recomputed in real time, giving an intuition of how errors introduced at one component propagate through the rest of the system.

¹<https://github.com/MrF3lix/system-failure-simulator>

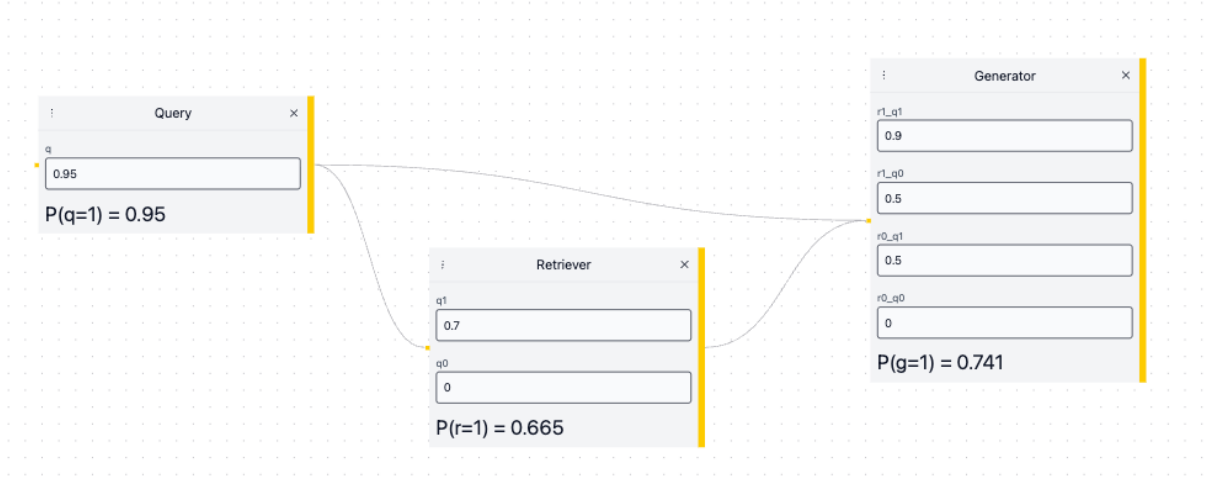


Figure 4: Interactive tool for exploring error propagation in an example RAG pipeline consisting of Query, Retriever, and Generator components. Adjusting the conditional probabilities (left inputs of each node) automatically recomputes the marginal probabilities (bottom of each node), illustrating how estimation errors propagate through the system.

2.3.2 Uncertainty Propagation

Where error propagation (Section 2.3.1) treats each conditional probability as a single point estimate, uncertainty propagation instead treats it as an unknown quantity about which we hold a belief, expressed as a probability distribution. This is the standard Bayesian view of parameter estimation: we start with a prior belief about a parameter, in this case a conditional success probability, and update it using observed data to obtain a posterior distribution over plausible values of that parameter, rather than a single number. For a binary outcome such as success or failure, the Beta distribution is the conjugate prior of the Binomial likelihood, meaning that updating a Beta prior with observed successes and failures again yields a Beta distribution, with parameters obtained simply by adding the observed counts. This is what makes the approach in this section both principled and computationally cheap: no numerical fitting is required to obtain the posterior, only counting.

We extend the model by replacing each explicit conditional probability with a random variable drawn from a Beta distribution:

$$P(x = 1) \sim \text{Beta}(\alpha, \beta) \quad (6)$$

where α and β are obtained directly from the evaluation set: for a given conditional probability, α corresponds to the number of observed successes and β to the number of observed failures among the relevant subset of samples. For instance, to estimate the Beta distribution for $P(r = 1 \mid q = 1)$, we set α to the number of samples with a good query for which retrieval succeeded, and β to the number of samples with a good query for which retrieval failed. To avoid an initial probability of exactly 0 or 1 when a conditional has no observed failures or successes, 1 is added to both the success and failure counts, corresponding to starting from a uniform prior, $\text{Beta}(1, 1)$, before updating with the observed data. Conditionals estimated from few samples naturally yield a wider, less concentrated Beta distribution,

while conditionals supported by many observations yield a narrower one, correctly reflecting the amount of evidence behind each estimate.

The Beta distribution was chosen because it is the natural fit for modeling a success/failure probability: given α observed successes and β observed failures, $\text{Beta}(\alpha, \beta)$ directly represents our belief about the true success probability. This makes the approach both easy to justify, since it follows directly from the observed counts, and easy to compute, since samples can be drawn without any additional numerical fitting.

We rely on Monte Carlo sampling to observe the resulting posterior distribution over the model output: for each of N iterations, a value is drawn independently from the Beta distribution of every conditional probability in the model, and the corresponding model equation is evaluated using this full set of sampled values. Repeating this process N times yields an empirical posterior distribution over the model output (e.g., $P(G)$ or $P(T)$), rather than a single point estimate.

This output distribution characterizes the overall uncertainty of the system’s performance, taking into account that both the retriever and the generator’s true success probabilities are not perfectly known. Summary statistics such as the mean, credible intervals, or the standard deviation of this distribution can then be reported alongside the point estimate, giving a more complete picture of how confident we can be in the system’s estimated performance.

Together, error and uncertainty propagation provide two complementary perspectives on the same underlying model: the former identifies which components a system estimate is most sensitive to using explicit point estimates, while the latter quantifies how much overall uncertainty remains once the imprecision of every component estimate is properly modeled through its posterior distribution.

2.4 Datasets

To evaluate the models introduced in this thesis, we use four Question Answer (QA) and fact verification datasets: FEVER, NQ, HotpotQA, and C4. All datasets except C4 are part of the Knowledge Intensive Language Tasks (KILT) benchmark [13], which unifies a diverse set of knowledge-intensive tasks around a single, shared Wikipedia-based knowledge source. This is particularly convenient for our purposes, since it allows us to reuse the same dataset loading and knowledge base implementation across all three datasets, rather than maintaining a separate retrieval pipeline for each.

2.4.1 FEVER

Fact Extraction and VERification (FEVER) [14] is a fact-verification dataset in which a given claim must be classified as *SUPPORTS*, *REFUTES*, or *NOT ENOUGH INFORMATION* based on evidence retrieved from the knowledge base. The dataset consists of approximately 126'000 samples. Compared to the other datasets used in this thesis, FEVER represents a considerably simplified task, since the system only needs to select one of three fixed labels rather than generate a free-form answer. Which makes it convenient to evaluate the generated answer by using exact match.

Question	Answer
Tetris has sold millions of physical copies.	SUPPORTS
Reference Paragraphs	
#30339	Electronic Gaming Monthly's 100th issue had Tetris as first place in the Greatest Game of All Time. In 2007, Tetris came in second place in IGN's 100 Greatest Video Games of All Time. In January 2010, it was announced that the Tetris franchise had sold more than 170 million copies, approximately 70 million physical copies and over 100 million copies for cell phones, making it the best-selling paid-downloaded game of all time. In 1991, PC Format named Tetris one of the 50 best computer games ever. The editors called it incredibly addictive and one of the best games of all time

Table 1: Example from the Fact Extraction and VERification (FEVER) dataset.

2.4.2 NQ

NQ [15] is a question answering dataset built from real questions submitted by users to the Google search engine. Answers are provided in two forms: a short answer (e.g., a number, a yes/no response, a date, or a named entity) and a long answer consisting of a passage from the source document. The dataset consists of approximately 91'700 samples. For this project only the short answers are used since they are also easy to evaluate by using exact match, however they have to be normalized before by removing punctuation and lowercase it.

Question	Answer
how i.met your mother who is the mother	Tracy McConnell

Reference Paragraphs

#40262098	Tracy McConnell (1984–2024) (colloquial: “The Mother”) is the title character from the CBS television sitcom “How I Met Your Mother”. The show, narrated by Future Ted (Bob Saget), tells the story of how Ted Mosby (Josh Radnor) met The Mother. Tracy McConnell appears in eight episodes, from “Lucky Penny” to “The Time Travelers”, as an unseen character; she was first seen fully in “Something New” and was promoted to a main character in season 9. The Mother is played by Cristin Milioti.
-----------	--

Table 2: Example from the Natural Questions (NQ) dataset.

2.4.3 HotpotQA

HotpotQA [16] is a multi-hop question answering dataset constructed from pairs of Wikipedia pages, requiring a system to combine information from two distinct passages in order to answer a question correctly. As with NQ, answers are short, consisting of numbers, yes/no responses, dates, or named entities. The dataset consists of approximately 100’000 samples.

Question	Answer
Which magazine was started first, Arthur’s Magazine or First for Women?	Arthur’s Magazine

Reference Paragraphs

#27290714	Arthur’s Magazine (1844–1846) was an American literary periodical published in Philadelphia in the 19th century. Edited by Timothy Shay Arthur, it featured work by Edgar A. Poe, J.H. Ingraham, Sarah Josepha Hale, Thomas G. Spear, and others. In May 1846 it was merged into “Godey’s Lady’s Book”.
#21232019	First for Women is a woman’s magazine published by Bauer Media Group in the USA. The magazine was started in 1989. It is based in Englewood Cliffs, New Jersey. In 2011 the circulation of the magazine was 1,310,696 copies.

Table 3: Example from the HotpotQA (HotpotQA) dataset.

2.4.4 C4

Catechism of the Catholic Church (C4) [17] is a multilingual benchmark based on the Catechism of the Catholic Church and its Compendium. The Catechism is a structured document with numbered paragraphs that has been officially translated into many languages under strict editorial alignment, while the Compendium reformulates its content into a question-answer format with explicit citations to the corresponding paragraphs. Together, these two texts form parallel monolingual corpora that share an identical semantic structure, which the authors leverage to enable direct, controlled comparison of RAG performance across languages. Beyond its theological origin, the dataset closely mirrors real-world institutional use cases of RAG, such as querying internal policy documents together with an associated

FAQ-style summary.

Compared to the other datasets used in this thesis, C4 is considerably smaller, consisting of only 598 QA pairs. Unlike FEVER, NQ, and HotpotQA, its answers are free-form text rather than short spans or fixed labels, and its questions are abstractive, requiring between 1 and 25 reference paragraphs to answer correctly.

Question	Answer
Why does man have a desire for God?	God himself, in creating man in his own image, has written upon his heart the desire to see him. Even if this desire is often ignored, God never ceases to draw man to himself because only in God will he find and live the fullness of truth and happiness for which he never stops searching. By nature and by vocation, therefore, man is a religious being, capable of entering into communion with God. This intimate and vital bond with God confers on man his fundamental dignity.

Reference Paragraphs

#27	The desire for God is written in the human heart, because man is created by God and for God; and God never ceases to draw man to himself. Only in God will he find the truth and happiness he never stops searching for.
#28	In many ways, throughout history down to the present day, men have given expression to their quest for God in their religious beliefs and behaviour: in their prayers, sacrifices, rituals, meditations, and so forth.
#29	But this “intimate and vital bond of man to God” can be forgotten, overlooked, or even explicitly rejected by man. Such attitudes can have different causes: revolt against evil in the world; religious ignorance or indifference; the cares and riches of this world.
#30	Although man can forget God or reject him, He never ceases to call every man to seek him, so as to find life and happiness. But this search for God demands of man every effort of intellect, a sound will, “an upright heart”.
#44	Man is by nature and vocation a religious being. Coming from God, going toward God, man lives a fully human life only if he freely lives by his bond with God.

Table 4: Example instance #2 from the Catechism of the Catholic Church (C4) dataset.

2.5 Metrics

Since the *System* and *Behavior* models introduced above are used to assess not only individual components but also their interplay within the full pipeline, we organize our metrics by what aspect of quality they capture, rather than by which component produces them. We distinguish three categories: *Retrieval Quality*, which assesses whether the retriever surfaces relevant paragraphs; *Task Quality*, which assesses the quality of a generated answer, either against a reference answer or independently of one; and *Generator Quality*, which assesses whether the system as a whole, including its decision to answer or abstain, produced the outcome expected by the user. Depending on the analysis, these metrics are computed either in isolation, e.g., evaluating the generator on gold reference paragraphs, or end-to-end, e.g., evaluating the generator on paragraphs returned by the retriever.

2.5.1 Retrieval Quality

Retrieval quality metrics assess how well the retriever identifies paragraphs relevant to a given query, independent of whether the generator subsequently produces a correct answer.

Binary

Binary evaluation of the retrieval quality is the obvious match for the proposed *System* and *Behavior* models. If the retriever finds all relevant paragraphs the retriever success score is 1 otherwise 0. This works well for the benchmark datasets that require exactly one paragraph, or multi-hop benchmarks that require all paragraphs to be found to answer the questions. However, for benchmarks where some relevant paragraphs are not marked as relevant or the answer could be found in multiple paragraphs but not all of them are required, then the binary metric falls short. Also it does not take into account at which position the paragraph is in the retrieved list of paragraphs, only that it exists within it.

Recall@K

To combat the shortcomings of the binary score the recall value can be calculated. It measures the fraction of all relevant paragraphs in the knowledge base that are successfully retrieved. Typically it is evaluated at a specific context size k , which means that if the number of relevant paragraphs exceeds k the recall cannot be properly measured. Using recall@ k also ignores the ranking of the retrieved paragraphs.

2.5.2 Task Quality

Task quality metrics assess the quality of a generated answer, either by comparing it to a gold reference answer (*reference-based*) or by evaluating it independently of any reference (*reference-free*).

Binary

Similar to the retrieval quality, the task quality can be measure binary using exact matching of the generated answer and the reference answer. 1 means the answers are the same and 0 means they differ. This also works well within the previous defined models of the RAG system. Both the generated and reference answer are normalized beforehand by converting to lowercase, removing stop words, and removing punctuation. This works well on benchmarks that provide a short answer such as a name, entity, or

number.

CheckEval Inspired

A CheckEval [18] inspired LLM-based evaluation is used to judge the generated answers similarly to the evaluation used in for the C4 dataset [17].

Dimension	Item
Correctness	Does the generated answer express the same essential doctrinal meaning as the canonical answer?
	Does the generated answer avoid any statement that contradicts the canonical answer?
	Are key terms and concepts used with the same meaning and nuance?
Completeness	Does the generated answer include all major elements present in the canonical answer?
	Does the generated answer provide sufficient explanation for the key ideas?
	Is the emphasis of the generated answer roughly proportional to the canonical answer?
Focus	Does the generated answer directly answer the question asked?
	Does the generated answer avoid tangential or unrelated information?
	Does the generated answer stay within the canonical answer’s scope (no speculative additions)?
	Is the generated answer appropriately concise?

Table 5: Evaluation Checklist

The judge is given the query, reference paragraphs and generated answer and evaluates based on the list of questions in Table 5.

```

1 <evaluation-item>
2   <question>{{ question }}</question>
3   <canonical-answer>{{ reference }}</canonical-answer>
4   <generated-answer>{{ hypothesis }}</generated-answer>
5 </evaluation-item>
6
7 <checklist>
8   <dimension name="correctness">
9     ...
10  </dimension>
11  <dimension name="completeness">
12    ...
13  </dimension>
14  <dimension name="focus">
15    ...
16  </dimension>
17 </checklist>

```

Each of the question results in a binary value produced by the judge and summed up create a score from 1 to 10 describing the task success.

2.5.3 Generator Quality

Generator quality metrics assess whether the system as a whole produced the outcome expected by the user, taking into account not only the correctness of a generated answer but also the system’s decision to abstain when appropriate. Since abstention is treated as a favorable outcome under failed retrieval, as introduced in the Behavior Model, task success is computed by combining an answer quality judgment (e.g., an EM, LLM-based, or reference-free score) with the system’s abstention decision: a task is considered successful if the system either abstains when retrieval fails, or produces a correct answer when retrieval succeeds.

2.5.4 Lexical Overlap

The lexical overlap can be computed between two strings s_1 and s_2 , e.g. the question and the reference paragraphs. To calculate it the question s_1 and reference paragraphs s_2 are preprocessed by converting to lower case, tokenizing, stemming and stopword removal. This produces two sets of words, one from the query W_1 and one from the reference paragraphs W_2 .

The lexical overlap is then calculated:

$$\text{LexicalOverlap}(s_1, s_2) = \frac{|W_1 \cap W_2|}{\min(|W_1|, |W_2|)} \quad (7)$$

In this equation, $|W_1 \cap W_2|$ is the number of words that are common to both sets. The term $\min(|W_1|, |W_2|)$ is the size of the smaller set. The score shows the proportion of shared words, relative to the shorter sentence. If one of the sets is empty, the function returns a score of 0.

2.6 Retrieval

Three retrievers are used in this project: a sparse retriever based on keyword matching, a dense retriever based on embeddings, and a hybrid retriever combining both. These represent the three broad categories most retrieval approaches fall into. As the goal of this project is not to identify the best-performing retrieval implementation but to study differences in retrieval behavior, one representative retriever was chosen per category rather than exhaustively comparing implementations within each.

For all three retrievers, the retrieval depth k (i.e., the number of paragraphs retrieved) is a configurable parameter set per experiment. In this project, $k \in \{5, 10, 20, 100\}$ is used.

2.6.1 Sparse

In this project sparse retrieval is based on the BM25 ranking function, implemented using the **retriv** library². The same preprocessing applied during index building (see Section 2.7.3) is applied to queries at retrieval time. Given a query Q containing the keywords $q_1, \dots, q_n$, the BM25 score of a paragraph D is

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \quad (8)$$

where $|D|$ is the length of paragraph D and avgdl is the average paragraph length in the entire knowledge base. k_1 and b are free parameters, chosen in this setting as $k_1 = 1.2$ and $b = 0.75$.

$\text{IDF}(q_i)$ is the inverse document frequency weight of the query term q_i :

$$\text{IDF}(q_i) = \ln \left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right) \quad (9)$$

where N is the total number of paragraphs in the knowledge base and $n(q_i)$ is the number of paragraphs containing q_i .

2.6.2 Dense

Dense retrieval uses the **jina-embeddings-v3** model [19] via the **SentenceTransformer**³ library. Following the model’s task-specific encoding scheme, embeddings for the index are computed with the **text-matching** task, while query embeddings are computed with the **retrieval.query** task. Retrieval is performed via nearest-neighbor search over a Faiss index built from the paragraph embeddings. Only the resulting ranking of paragraphs is used downstream, while the underlying similarity scores are not used further.

²<https://github.com/AmenRa/retriv>

³<https://huggingface.co/sentence-transformers>

2.6.3 Hybrid

The hybrid retriever combines the dense and sparse retrievers by applying Reciprocal Rank Fusion (RRF) to the two rankings. RRF is particularly suited to this setting, as it does not require the retrieval methods to produce comparable scores. Only the rank position of each retrieved paragraph is used:

$$\text{RRF}(d) = \sum_{i=1}^n \frac{1}{k + \text{rank}_i(d)} \quad (10)$$

where $k = 60$.

2.7 Implementation

The pipeline is built around a shared data model, `Query`, which is passed through and extended by each pipeline task. Each task is implemented as a subclass of an abstract base class operating on `Query`: the Retriever adds the retrieved paragraphs, the Generator adds the generated answer, and the Judge adds the evaluation metrics. Figure 5 shows the resulting class structure.

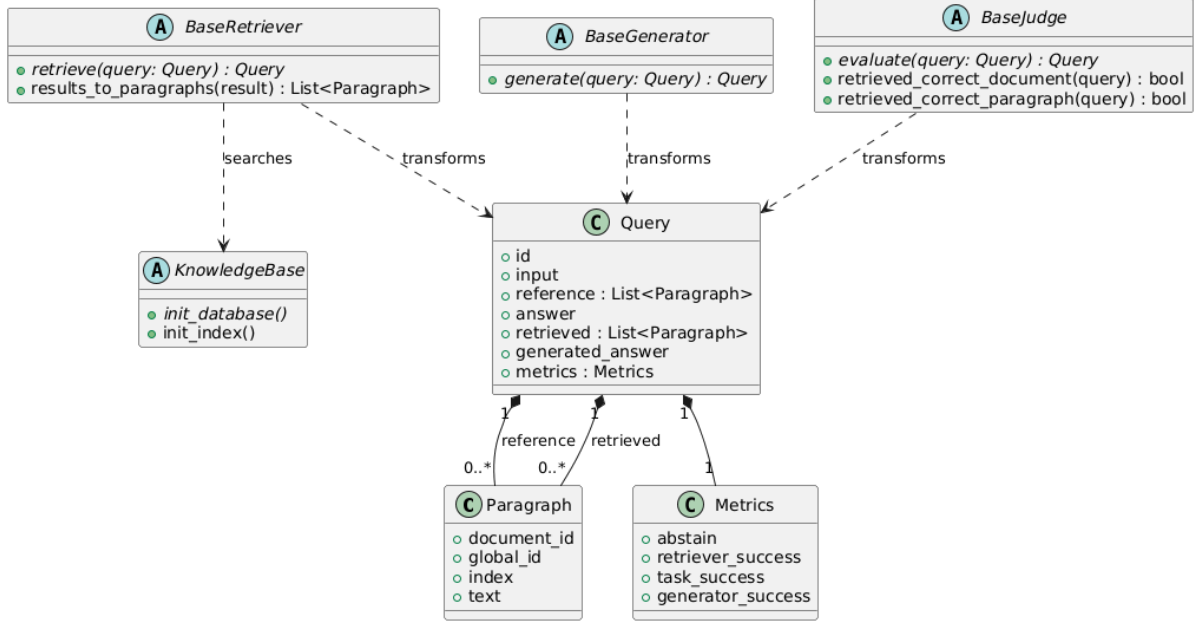


Figure 5: Class diagram of the RAG implementation.

2.7.1 Pipeline Orchestration

A pipeline run consists of five tasks, configured via YAML files and executable either as a full run or independently:

- **Init** sets up the knowledge base and preprocesses the dataset according to its dataset specific implementation.
- **Precompute** an optional step that runs retrieval with $k = 100$ ahead of time, useful when evaluating multiple context sizes without repeating retrieval.
- **Retrieve** takes the top- k paragraphs from the precomputed results if available, otherwise retrieves them directly using the configured retriever implementation. Also resolves paragraph `global_id`'s to their texts via the knowledge base.
- **Generate** produces the LLM calls to generate an answer. Multiple LLM backends are supported (OpenAI, Ollama, vLLM), depending on where the model is hosted and which API it exposes.
- **Judge** evaluates the generated answers and the retrieval results using the configured metrics.

2.7.2 Data Model

Each dataset is preprocessed into a common data model. Retrieved and reference paragraphs are represented using the `Paragraph` class:

```

1 class Paragraph(BaseModel):
2     document_id: int
3     index: int
4     global_id: Optional[int] = None
5     text: Optional[str] = None

```

The `global_id` is a unique paragraph identifier that is used across the different indices.

```

1 class Query(BaseModel):
2     id:
3     input:
4     answer: str
5     reference: list[Paragraph] = []
6     retrieved: list[Paragraph] = []
7     generated_answer: Optional[str] = None
8     retriever_success = None
9     task_success = None
10    abstain = None
11    generator_success = None

```

The overall data model, referred to as `Query`, is used across all pipeline tasks. All tasks (Retriever, Generator, Judge) accept and return a `Query` object, each adding the information it produces: the Retriever task adds the retrieved paragraphs, the Generator task adds the generated answer, and the Judge task adds the evaluation metrics.

2.7.3 Knowledge Base

For each dataset, a knowledge base is created containing all paragraphs of the dataset in a searchable form. The knowledge base consists of a DuckDB⁴ database holding all paragraphs, enabling fast lookup of paragraph texts by `global_id`, and an index built from it. The index depends on the retrieval strategy used. At retrieval time, the index is first used to search for the relevant paragraph `global_id`'s, then the corresponding texts are retrieved from the DuckDB database.

Sparse Index

- BM25-based, implemented using the `retriv` library.
- Paragraphs are preprocessed using whitespace tokenization, stemming, stopword removal, lower-casing, ampersand normalization, special character normalization, and punctuation removal.

⁴<https://duckdb.org/>

Dense Index

- Paragraphs are embedded using the `jinaai/jina-embeddings-v3` model via the SentenceTransformer library.
- Embeddings are computed using the task-specific `text-matching` mode.
- Embeddings are stored in an HNSWFlat index provided by the Faiss library [20].

Since datasets vary in structure, each requires its own implementation of the abstract `KnowledgeBase` class, exposing an `init_database` and an `init_index` method.

2.7.4 Retriever

Each retrieval strategy implements the abstract `BaseRetriever` class, which takes a `Query` and returns it with the retrieved paragraphs attached.

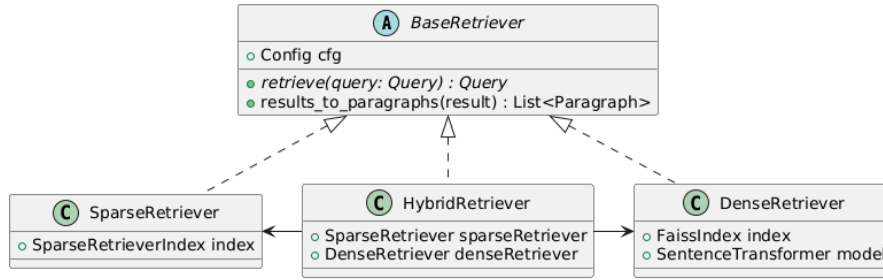


Figure 6: Class diagram of the different retriever implementations.

Figure 6 shows that each retrieval strategy implements the abstract class and in addition to that the hybrid retriever also has an instance of the sparse and dense retriever because it reuses these implementation to retrieve the paragraphs before merging them using RRF.

2.7.5 Generator

Each generation strategy implements the abstract `BaseGenerator` class, which takes a `Query` containing the input statement and retrieved paragraphs, and returns it with the generated answer attached. Depending on where a model is hosted, e.g. locally using vLLM, a different generator implementation is required. At the moment the Ollama API⁵ and any OpenAI styled API⁶ such as the OpenAI models or any models that run using vLLM with the OpenAI compatibility API⁷.

For the generator the following configurations can be set:

⁵<https://docs.ollama.com/api/introduction>

⁶<https://developers.openai.com/api/reference/overview>

⁷https://docs.vllm.ai/en/stable/serving/online_serving

```
1 generator:
2   type: openai
3   model: Qwen/Qwen3.5-9B
4   base_url: http://fresko.zhaw.ch:8650/v1
5   api_key: dummy
6   temperature: 0.7
7   max_tokens: 512
8   max_threads: 15
9   system_prompt_template: system_prompt_template.jinja
10  user_prompt_template: user_prompt_template.jinja
```

The `max_thread` configuration sets how many threads can be started asynchronously to send the generator requests.

For the experiments three LLM were employed as generators:

- **Apertus 8B (Apt)** [21], a fully open-source, decoder-only transformer model with 8 billion parameters, developed with an emphasis on full transparency of both model weights and training data.
- **Gemma3 12B (Gem)**[22], a lightweight open multimodal model developed by Google, comprising 12 billion parameters and trained on 12 trillion tokens spanning more than 140 languages.
- **Qwen 3.5 9B (Qwn)** [23], a multimodal model developed by Alibaba Cloud with 9 billion parameters, supporting multilingual inference across 201 languages. Although the model includes a dedicated reasoning mode, this functionality was not employed in the present work

2.7.6 Judge

Two implementations for the **BaseJudge** exists, the **StaticJudge** and the **LLMJudge**. The **StaticJudge** computes the metrics that are deterministic, such as the binary successes on retriever and task success, while the **LLMJudge** creates another generation call to a judge model.

3 Results

This section presents the empirical validation of the *System* and *Behavior* models introduced in Section 2. We proceed from a naive, end-to-end evaluation across 27 RAG configurations (Section 3.1) to a series of increasingly targeted experiments that isolate specific factors in the pipeline: the distinction between task and generator success (Section 3.2), the relationship between retrieval and task success under controlled retrieval conditions (Sections 3.3 to 3.5), the effect of context size, composition, and ordering (Sections 3.6 to 3.9), and the reliability of retrieval and generation metrics themselves, including automated LLM-based judges (Sections 3.10 to 3.13).

Each experiment states its model, dataset, configuration, and hypothesis before presenting results, followed by a discussion of the findings and their limitations.

3.1 System Model Evaluation

In the first experiment, we run all pipeline tasks. We count the retrieval and generator successes according to the system model in Section 2.2.2. We assume that all queries are good, $q = 1$, because they are unaltered and taken directly from the benchmarks. Under this assumption, the model simplifies to:

$$P(Q = 1, R, G) = P(R) \cdot P(G | R) \quad (11)$$

We define the marginal generator success probability as:

$$P(g = 1) = P(r = 1) \cdot P(g = 1 | r = 1) + P(r = 0) \cdot P(g = 1 | r = 0) \quad (12)$$

We run this experiment across 27 RAG configurations, three datasets, three retrievers, and three generators.

Model: System Mode: $P(Q = 1, R, G)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, Hybrid, K=5

Retriever Success: Binary

Generator: Apertus 8B (Apt), Gemma3 12B (Gem), Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: A higher retrieval success leads to a higher generator success.

Results

Table 6 shows the marginal success probabilities for retrieval success $P(R = 1)$ and generator success $P(G = 1)$ across 27 RAG configurations. We run each configuration with 10'000 queries from the corresponding dataset. The table also shows the conditional generator success given retrieval success, $P(G = 1 | R = 1)$, and given retrieval failure, $P(G = 1 | R = 0)$.

On the FEVER dataset, the hybrid retriever performs best, with $P(r = 1) = 0.66$. Across all generators, the hybrid retriever also gives the best result, and the Gem generator reaches the highest score at

Table 6: Retrieval and generator binary evaluation across 27 RAG configurations for 10’000 queries per dataset. The evaluation shows the marginal success probabilities for retrieval success $P(R = 1)$ and generator success $P(G = 1)$. It also shows the conditional generator success given retrieval success $P(G = 1|R = 1)$ and given retrieval failure $P(G = 1|R = 0)$.

Dataset	Retriever	Generator	$P(R = 1)$	$P(G = 1)$	$P(G = 1 R = 1)$	$P(G = 1 R = 0)$
FEVER	Dense	Gem	0.60	0.78	0.94	0.54
		Apt		0.75	0.85	0.61
		Qwn		0.75	0.93	0.48
	Sparse	Gem	0.46	0.74	0.95	0.57
		Apt		0.76	0.87	0.67
		Qwn		0.71	0.93	0.52
	Hybrid	Gem	0.66	0.86	0.94	0.71
		Apt		0.80	0.85	0.70
		Qwn		0.84	0.93	0.66
HotpotQA	Dense	Gem	0.15	0.25	0.56	0.20
		Apt		0.26	0.49	0.22
		Qwn		0.26	0.61	0.20
	Sparse	Gem	0.22	0.32	0.63	0.23
		Apt		0.32	0.55	0.25
		Qwn		0.33	0.69	0.23
	Hybrid	Gem	0.21	0.32	0.60	0.24
		Apt		0.30	0.52	0.25
		Qwn		0.33	0.65	0.24
NQ	Dense	Gem	0.35	0.24	0.47	0.12
		Apt		0.24	0.40	0.15
		Qwn		0.24	0.47	0.12
	Sparse	Gem	0.24	0.22	0.49	0.13
		Apt		0.23	0.40	0.18
		Qwn		0.22	0.47	0.13
	Hybrid	Gem	0.35	0.27	0.49	0.15
		Apt		0.25	0.41	0.17
		Qwn		0.27	0.47	0.16

$P(G = 1) = 0.86$. On this retriever, the generator success after retrieval failure is higher than on the sparse and dense retrievers. For this dataset, the generator success after retrieval success is high across all retriever and generator pairs.

On the HotpotQA dataset, the retrieval success is lower than on FEVER. The dense retriever reaches a success rate of only $P(R = 1) = 0.15$, while the sparse retriever reaches $P(R = 1) = 0.22$. Even when the dense retriever finds both required documents, it scores the lowest of all generators.

The NQ benchmark is the most difficult dataset for generating a correct answer. The retrievers perform better than on the HotpotQA dataset. The dense and hybrid retrievers reach the same success rate,

$P(r = 1) = 0.35$. Even when retrieval succeeds, the generator success stays below 0.5. The generator success after retrieval failure is also low.

Discussion

The high generator success on the FEVER dataset with the hybrid retriever appears to result from the high generator success after retrieval failure. This suggests that some paragraphs lead to the right answer without being marked as relevant, and that the dense and sparse retrievers succeed and fail on different queries. As a result, the hybrid retriever combines the strengths of both. On the FEVER dataset, the generator success is high when retrieval succeeds. On the other datasets, the generators still struggle to produce the correct answer even when retrieval succeeds. One possible reason is that the query only needs one of three possible labels to be answered correctly, which may make the task too easy for the generators. The generators may also have seen these questions during training.

The low retrieval success on HotpotQA may result from the two required paragraphs per query: if one paragraph is harder to find than the other, retrieval fails easily. The higher generator success after retrieval failure may indicate that some questions do not need both paragraphs to be answered correctly.

On the NQ dataset, the dense and hybrid retrievers perform equally well, while the sparse retriever performs worse. This shows that the hybrid retriever ranks the dense retrieval results higher when it combines them. When the sparse retriever finds the correct paragraph, the dense retriever most likely finds it too. In the other cases, the dense retriever ranks the correct paragraph higher.

Limitations

- This experiment does not take abstention into account. In some cases, the generator success after retrieval failure is high, but this behavior may not be desirable.
- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.2 Behavior Model Evaluation

In the previous experiment (see Section 3.1) we have shown a very simplified evaluation setup using the system model (see Section 2.2.2) and the results showed a high generator success on retrieval failure for some of the benchmark datasets.

In the system model we do not take into consideration what kind of behavior we expect from the RAG system. To model this we created the behavior model (see Section 2.2.3) which distinguishes between generator success and task success. To summarize, the task success is whether the generator produced the correct answer and solves the task while generator success is whether the generator behaved correctly, e.g. abstained from answering the question if the retrieval failed. This allows us a closer look at what is actually happening in the tested RAG setups.

We conduct this experiment across the 27 RAG configurations from the previous experiment.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, Hybrid, $K=5$

Retriever Success: Binary

Generator: Apertus 8B (Apt), Gemma3 12B (Gem), Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: Two generators might have the same task success but differ in the expected behavior a.k.a generator success.

Results

Table 7 shows the marginal success probabilities for retrieval success $P(R = 1)$, abstention $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ across 27 RAG configurations. Each configuration is run with 10'000 queries from the respective dataset.

The columns for retrieval success and task success are equivalent to the Table 6 in the previous experiment. While in the previous experiment task success and generator success are equal, here the generator success takes the definition of the behavior model (see Section 2.2.3) where the generator success reflects the desired behavior of the system.

In the FEVER dataset the generator success drops compared to the task success across all configurations. The abstention rate on the sparse and dense retriever is higher than on the hybrid retriever. Across the generators Apt shows the lowest abstention rate and the largest drop between the task success and generator success.

On the HotpotQA the generator success is generally higher than the task success except on the Apt generator. This is also reflected in the abstention rate across the configurations. Abstention is the highest on the dense retriever with Qwn abstaining almost half of the time $P(A = 1) = 0.49$, this is the highest abstention rate across all the 27 configurations. This configuration also shows the highest generator success in the HotpotQA dataset with $P(G = 1) = 0.57$.

The NQ dataset shows a similar pattern to the HotpotQA dataset with a generally higher generator success rate compared to the task success except on the Apt generator. On the dense retriever the task success rates are identically across the generators with $P(T = 1) = 0.24$, however the generator success varies a lot with Apt only achieving $P(G = 1) = 0.16$ and Qwn $P(G = 1) = 0.50$.

Table 7: Retrieval and generator binary evaluation across 27 RAG configurations for 10’000 queries per dataset. Evaluation shows marginal success probabilities for retrieval success $P(R = 1)$, abstention $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$

Dataset	Retriever	Generator	$P(R = 1)$	$P(A = 1)$	$P(T = 1)$	$P(G = 1)$
FEVER	Dense	Gem	0.60	0.18	0.78	0.74
		Apt		0.13	0.75	0.61
		Qwn		0.20	0.75	0.74
	Sparse	Gem	0.46	0.22	0.74	0.64
		Apt		0.12	0.76	0.50
		Qwn		0.25	0.71	0.66
	Hybrid	Gem	0.66	0.10	0.86	0.70
		Apt		0.09	0.80	0.61
		Qwn		0.11	0.84	0.70
HotpotQA	Dense	Gem	0.15	0.31	0.25	0.39
		Apt		0.04	0.26	0.11
		Qwn		0.49	0.26	0.57
	Sparse	Gem	0.22	0.23	0.32	0.36
		Apt		0.03	0.32	0.15
		Qwn		0.39	0.33	0.53
	Hybrid	Gem	0.21	0.22	0.32	0.34
		Apt		0.03	0.30	0.14
		Qwn		0.37	0.33	0.50
NQ	Dense	Gem	0.35	0.26	0.24	0.41
		Apt		0.03	0.24	0.16
		Qwn		0.36	0.24	0.50
	Sparse	Gem	0.24	0.30	0.22	0.41
		Apt		0.04	0.23	0.14
		Qwn		0.43	0.22	0.53
	Hybrid	Gem	0.35	0.20	0.27	0.36
		Apt		0.02	0.25	0.16
		Qwn		0.31	0.27	0.45

Across all datasets we can observe that a higher retrieval success leads to a higher task and generator success.

Table 8 gives a closer look at the abstentions and task success by showing the conditional success probabilities for abstention given retrieval failure $P(A = 1|R = 0)$ and retrieval success $P(A = 1|R = 1)$ as well as task success given non-abstention and retrieval failure $P(T = 1|R = 0, A = 0)$ and retrieval success $P(T = 1|R = 1, A = 0)$.

Across the datasets the Apt generator barely abstains while also having the lowest task success across the different configurations with or without retrieval success. However, on retrieval success in the FEVER dataset the Apt generator has the highest abstention rate.

Table 8: Retrieval and generator binary evaluation across 27 RAG configurations for 10’000 queries per dataset. Evaluation shows the conditional success probabilities for abstention given retrieval failure $P(A = 1|R = 0)$ or retrieval success $P(A = 1|R1)$ as well as task success given non-abstention and retrieval failure $P(T = 1|R = 0, A = 0)$ and retrieval success $P(T = 1|R = 1, A = 0)$.

Dataset	Retriever	Generator	$P(A = 1 R0)$	$P(A = 1 R1)$	$P(T = 1 R0, A0)$	$P(T = 1 R1, A0)$
FEVER	Dense	Gem	0.42	0.02	0.94	0.96
		Apt	0.25	0.06	0.81	0.90
		Qwn	0.46	0.03	0.89	0.95
	Sparse	Gem	0.39	0.02	0.94	0.96
		Apt	0.20	0.04	0.83	0.91
		Qwn	0.43	0.03	0.92	0.96
	Hybrid	Gem	0.24	0.02	0.94	0.96
		Apt	0.16	0.05	0.83	0.90
		Qwn	0.27	0.03	0.91	0.95
HotpotQA	Dense	Gem	0.36	0.03	0.31	0.57
		Apt	0.04	0.01	0.23	0.49
		Qwn	0.57	0.04	0.46	0.64
	Sparse	Gem	0.29	0.01	0.33	0.63
		Apt	0.04	0.01	0.26	0.55
		Qwn	0.49	0.03	0.45	0.70
	Hybrid	Gem	0.27	0.02	0.33	0.61
		Apt	0.04	0.01	0.26	0.52
		Qwn	0.46	0.04	0.46	0.68
NQ	Dense	Gem	0.38	0.02	0.19	0.49
		Apt	0.04	0.01	0.16	0.40
		Qwn	0.51	0.06	0.26	0.49
	Sparse	Gem	0.39	0.03	0.22	0.51
		Apt	0.05	0.01	0.19	0.41
		Qwn	0.55	0.08	0.29	0.51
	Hybrid	Gem	0.29	0.02	0.21	0.50
		Apt	0.03	0.01	0.18	0.41
		Qwn	0.44	0.06	0.28	0.50

The FEVER dataset shows the highest task success rates even when retrieval fails.

Across all the configuration the task success is consistently improved whenever the retrieval succeeds. However, the improvement on the FEVER dataset is not as much as on the other datasets.

On the NQ dataset with the dense retriever we can see where the large difference in task success and generator success in Table 7 comes from. The generator Qwn abstention on retrieval failure is $P(A = 1|R = 0) = 0.55$ while the Apt generator only abstains $P(A = 1|R = 0) = 0.05$.

Discussion

Apt abstains the least from all the generators, combined with the large drop between the task and generator success it shows that this generator does not adhere to the behavior we desire.

The high abstention rate on the HotpotQA, dense, Qwn configuration is in line with the low retrieval success rate, this is the behavior we look for in a system that does not retrieve well.

The results on FEVER show again a different behavior across the different configuration with a much higher task success rate with and without retrieval success. This indicates again that the question in this dataset might be too easy to answer or the task of fact checking is not suitable for this purpose.

These results show that while generators can have the same task success rate their generator success rate can vary a lot and that this is driven by the difference in abstention behavior.

Limitations

- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.3 Probabilistic Retriever

In the previous experiments we showed that a higher retriever success implies a higher task and generator success. To investigate how an improvement in retrieval success translates to a higher task success we implement a Probabilistic Retriever (PR) strategy for the sparse, dense, and hybrid base retrievers.

PR is a retrieval strategy where a base retriever is defined and the target success rate can be set before hand. As an example: Strategy = Dense, Success Rate = 0.8 This means that in 80% of the times the retriever returns the correct paragraphs and fills up the rest of the context using the dense retriever strategy. In the other 20% of the times the PR returns the top ranked paragraphs from the dense retriever but filters out all the correct paragraphs.

The idea of using the PR is to measure the task success rate independent of an actual retriever implementation but rather given a fixed retriever success. This enables comparing different retrieval successes and their influence on the task success rate.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, $N = 10'000$

Retriever: PR, Dense, Sparse, Hybrid, $K=5$

Retriever Success: Binary

Generator: Gemma3 12B (Gem)

Generator Success: Binary, Exact Match

Hypothesis: A higher retrieval success leads to a higher task success.

Results

Figure 7 shows the task success $P(T = 1)$ on the y-axis and the retriever success $P(R = 1)$ on the x-axis. For each retriever strategy (Dense, Sparse, Hybrid) the retriever success and the task success are measured. These are shown as the independent crosses in the plot. Then for each retriever strategy the PR is applied for all retriever success rates between 0 and 1 in 0.1 increments. These are shown as the lines with the error bars.

This shows that for a higher retriever success a higher task success is shown in this experiment. However the rate at which the task success increases seems to be dependent on the retriever strategy. The Sparse retriever is performing the worst but seems to increase at the highest rate. Also it is worth noting the estimates using just the retriever strategy does not overlap with the probabilistic version. Overall this shows that the relation between task success and retriever success is linear but depends on the retriever.

When the retrievals fails completely $P(R = 1) = 0$ the task success differs across the retriever strategies.

When the retrieval succeeds the task success converges at around $P(T = 1) \approx 0.92$.

Discussion

It is not clear why the point estimates for a retriever strategy does not fall within the error bar of the PR for the given base retriever. One explanation could be the difference in the order of the paragraphs within the context given to the generator because the PR strategies place the correct paragraphs at the beginning.

The difference at $P(R = 1) = 0$ could be explained by the different behaviors of the retrieval strategy,

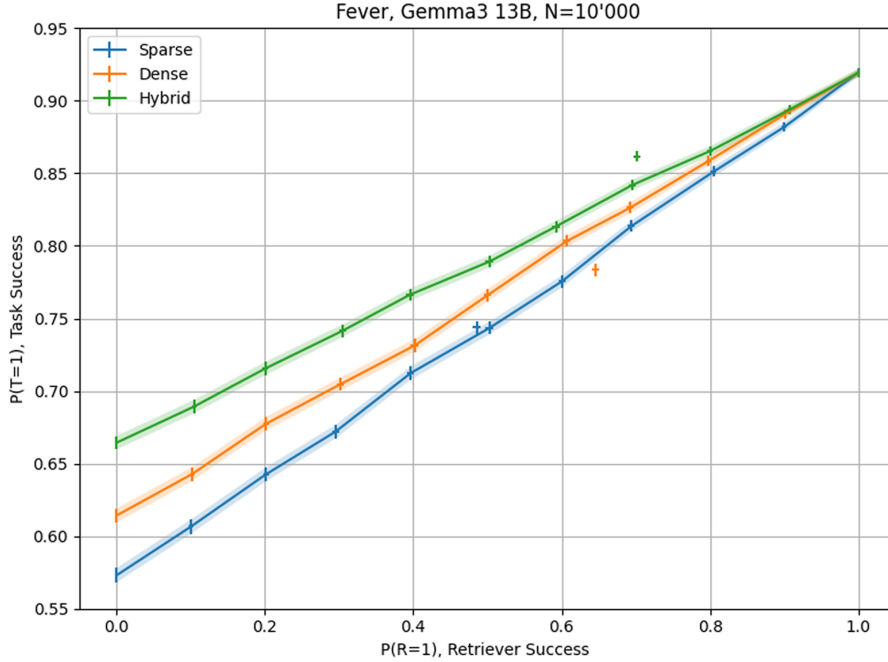


Figure 7: Probabilistic Retriever (PR) with the three base retrieval strategies (Dense, Sparse, Hybrid) showing the relationship between retrieval success and task success. The changes in the PR are shown as a line plot with an error bar and the results from the base retriever are shown as a dot with error bars.

for this dataset the lexical search of the sparse retriever might not be advantageous compared to the semantic similarity of the dense retriever.

The difference in the steepness indicates that some retrievers benefit from paragraphs found that are in the dataset not marked as relevant but are helpful to produce the correct answer.

The convergence at $P(T = 1) \approx 0.92$ suggests that if all necessary paragraphs are found the task can be solved and the difference in the "non-relevant" paragraphs can be neglected.

Limitations

- The PR experiments are resource intensive and take a lot of time, ca. 22 hours per retriever which is why this experiment was conducted on a single dataset, generator pair instead of the previous configurations.
- Correct paragraphs are placed at the beginning of the retrieved context. This could be overcome by keeping the positional index of the correctly retrieved paragraph.
- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.4 Probabilistic Approximated

The previous experiment (see Section 3.3) has shown that the relationship between the retriever success and the task success seems to be linear.

This is supported by the following:

$$\begin{aligned}
 P(T) &= P(T|R) \cdot P(R) \\
 P(T = 1) &= P(T = 1|R = 1) \cdot P(R = 1) + P(T = 1|R = 0) \cdot P(R = 0) \\
 &= P(T = 1|R = 1) \cdot P(R = 1) + P(T = 1|R = 0) \cdot (1 - P(R = 1)) \\
 &= \underbrace{(P(T = 1|R = 1) - P(T = 1|R = 0))}_{a} \cdot \underbrace{P(R = 1)}_x + \underbrace{P(T = 1|R = 0)}_b \\
 &= a \cdot x + b
 \end{aligned}$$

So the relation can be formulated as the difference in probability of task success given retrieval success and the probability of task success given retrieval failure multiplied by the target retrieval success and added to the probability of task success given retrieval failure. In practical terms, when the retriever fails the task success is equivalent to the probability of task success given retrieval failure. The difference in task success probability when retrieval succeeds and fails is the slope of the function. This also ties in with the behavior model and can use the observations from the experiments as priors which means that we can easily add error bars to the posterior function.

With this we can extract the linear relationship between the retriever and task success from a single experiment rather which reduces the experiment time by a factor of 10.

As part of this experiment we also test two additional retrieval strategies:

- **Oracle:** This strategy takes all the paragraphs from the reference exactly as is. It varies in size because not all questions have the same amount of reference paragraphs. This gives an approximation of perfect retrieval without "non-relevant" paragraphs.
- **Empty:** This strategy retrieves no paragraphs. This is useful to show how well the generator works without any additional information.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: PR, Dense, Sparse, Hybrid, Oracle, Empty, $K=5$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: The relationship between the retrieval success and task success can be modeled using a linear function and estimated from the results of a single experiment.

Results

In Figure 8 we show the approximated relation between retriever success $P(R = 1)$ and task success $P(T = 1)$ across multiple datasets (FEVER, HotpotQA, NQ) and multiple retriever strategies (Sparse, Dense, Hybrid, Oracle, Empty).

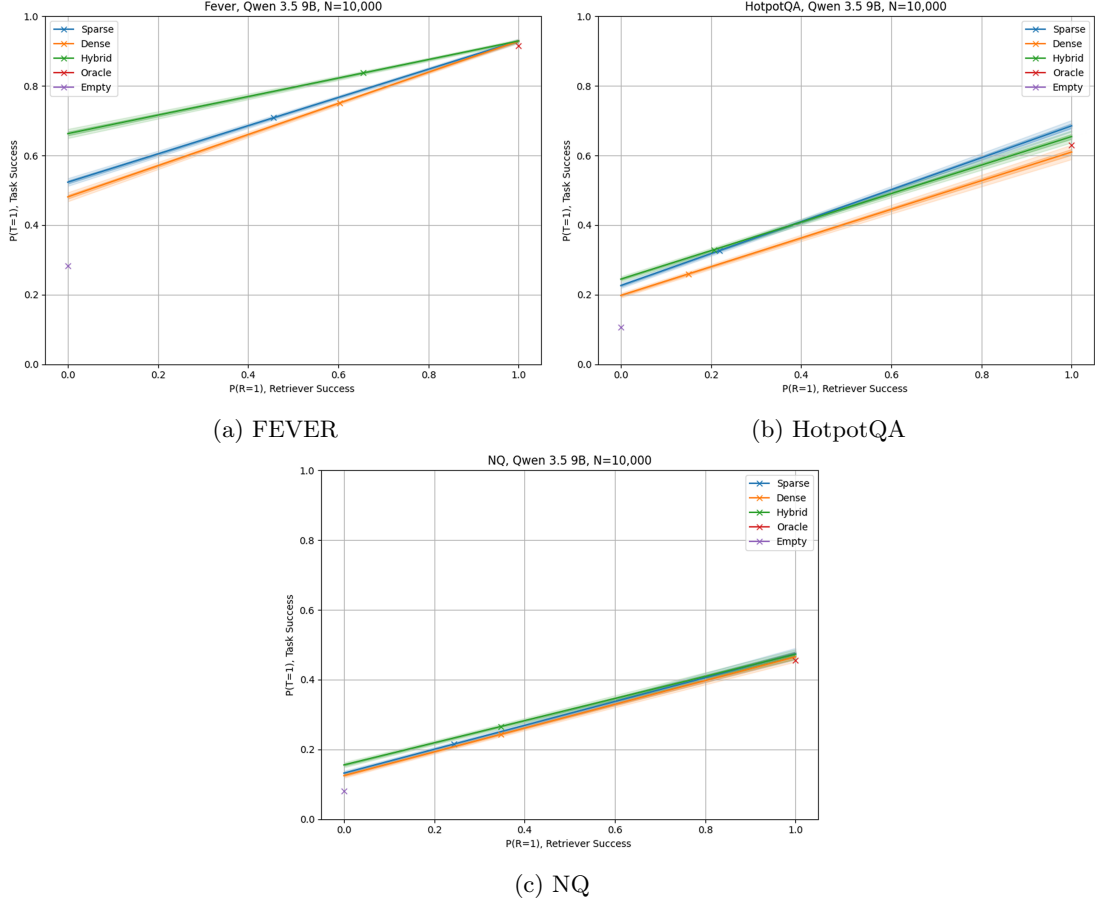


Figure 8: Approximated relation between retriever success $P(R = 1)$ and task success $P(T = 1)$ across multiple datasets (FEVER, HotpotQA, NQ) and multiple retriever strategies (Sparse, Dense, Hybrid, Oracle, Empty).

Figure 8a shows that for a given retriever success the task success is different depending on the retrieval strategy. The dense retriever performs the worst but has the steepest improvement. The hybrid retriever performs the best. The empty retriever shows a task success $P(T = 1) \approx 0.3$ without any retrieved documents. All retriever strategies seem to converge near the result of the oracle retriever. However the oracle retriever seems to show a little lower task success.

On the HotpotQA dataset the dense retriever performs worse as well, while sparse and hybrid overlap with sparse being slightly steeper. The retrieval strategies do not converge at a single point.

Figure 8c shows that in the NQ dataset the difference between the retriever strategies is minimal. The retrievers however seem to be above the oracle retriever strategy.

Discussion

This experiment showed that there is some confounding factor, other than whether a retriever found all relevant paragraphs, that influences the task success. Two retriever with the same retrieval success do not necessary have the same task success.

Interestingly the oracle retriever does not seem to be the upper bound of the task success. Indicating that some documents that are not marked as relevant might have an influence on answering a question correctly.

The difference between the retrievers might be explained by the different types of paragraphs they retrieve. The sparse retriever is likely to retrieve paragraphs with similar words while the dense retriever might pick up paragraphs with paraphrased content.

Overall it seems like all the retrieval strategy retrieve something that is better than not having any additional context.

There does not seem to be a best retrieval strategies across the different datasets.

Limitations

- The difference in the retriever strategies is not apparent but might be explained by the "non-relevant" paragraphs they retrieve.
- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.5 Oracle Intersection

The linear approximations in the previous experiment (see Section 3.4) showed that two retrievers with the same retrieval success do not necessarily have the same task success. There seems to be another factor, one explanation could be all the paragraphs that are retrieved which are "non-relevant", meaning that in the dataset they are not marked as reference paragraphs but might still contain information that could be used to answer the question.

In this experiment we remove all "non-relevant" paragraphs from the context before passing it to the generator. We call this process *Oracle Intersection (OI)*, since the intersection of the reference and retrieved paragraph sets are used to pass to the generator.

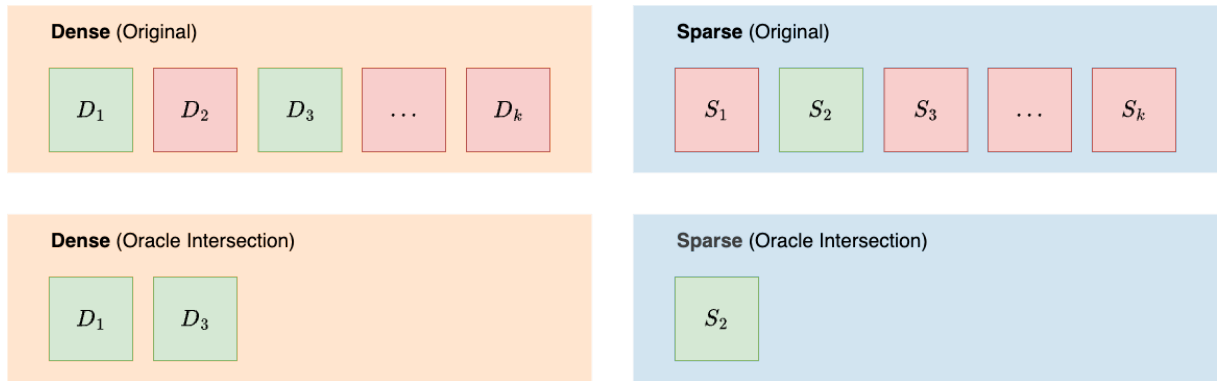


Figure 9: Example of the Oracle Intersection (OI) process on the dense and sparse retrieval strategies.

Figure 9 shows how the OI process looks like. The paragraphs are retrieved using the known strategies (Sparse, Dense, Hybrid). From the retrieved paragraphs, the paragraphs that do not appear in the reference paragraph list of the dataset are removed.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: OI, Dense, Sparse, Hybrid, Oracle, Empty, $K=5$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: When all "non-relevant" paragraphs are removed the retriever strategies should behave the same.

Results

Figure 10 shows the approximate relationship between the retriever success $P(R = 1)$ and task success $P(T = 1)$ on the datasets FEVER, HotpotQA, NQ datasets with the adjusted OI retrievals.

On the FEVER dataset (Figure 10a) the difference between the retrievers is now much smaller. They converge at the point estimate for the oracle retriever. However, they still exhibit some difference at $P(R = 1) = 0$.

Figure 10b shows that on the HotpotQA dataset the OI does not affect results, they are nearly identical

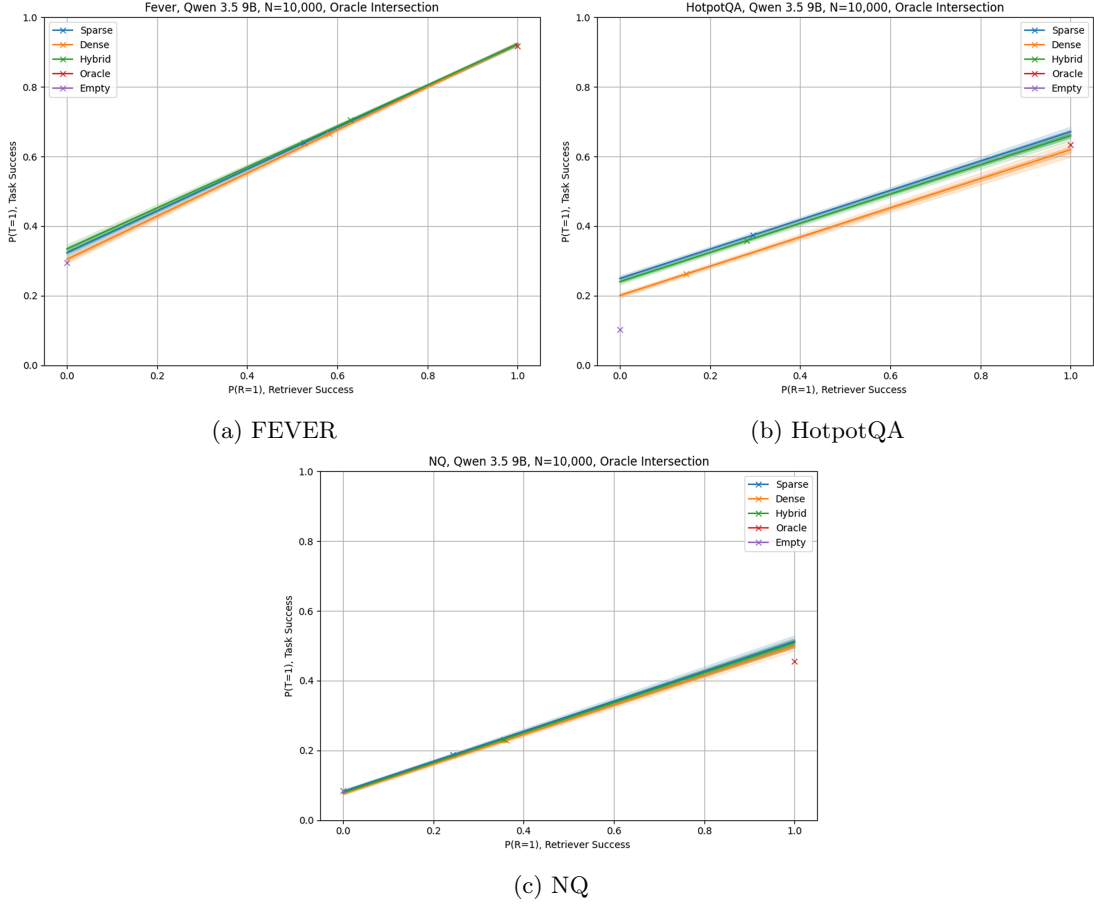


Figure 10: Approximated relation between retriever success $P(R = 1)$ and task success $P(T = 1)$ across multiple datasets (FEVER, HotpotQA, NQ) and multiple retriever strategies adjusted with the Oracle Intersection (OI) process (Sparse, Dense, Hybrid, Oracle, Empty).

to the results in the previous experiment (see Section 3.4).

Figure 10c shows that in the NQ dataset the different retrievers are nearly identical. However, on $P(R = 1) = 1$ all the retrievers lie above the oracle strategy.

Discussion

This shows that there is some factor that the approximation does not take into account. On $P(R = 1) = 0$ all retrievals across all strategies are empty so they should overlap with the point estimate of the empty retriever. On the HotpotQA dataset this does not seem to be the case, there seems to be something missing or not expressed by using binary success rates.

Limitations

- The difference in the retriever strategies is not apparent but might be explained by the "non-

relevant” paragraphs they retrieve.

- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.6 Compare Context Size

In the previous experiments we always used a fixed context size of $k = 5$. This means that the retrievers always return 5 paragraphs.

We want to test how the retrieval, abstention, and task success behaves under different context sizes $k \in [5, 10, 20, 100]$.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, Hybrid, $k \in [5, 10, 20, 100]$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: A higher context size leads to better retrieval success, lower abstention, and better task success.

Results

Figure 11 shows the retriever success, abstention, and task success for $k \in [5, 10, 20, 100]$ across the three datasets FEVER, HotpotQA, and NQ on the three retrieval strategies (Sparse, Dense, Hybrid).

On the Fever dataset (Figure 11a) the retriever success increased with a large context size. The dense retriever success does not increase the same as the others, on the context size with 100 paragraphs. The abstention dropped across all retrieval strategies, the decrease after $k = 20$ is minimal. The task success increased slightly across all retrieval strategies, after $k = 20$ it seems to plateau.

In the HotpotQA dataset the improvements between $k = 20$ and $k = 100$ on the retrieval success looks steeper than on the other datasets. Here the dense retriever performs the worse and shows the least improvement across different k sizes.

The NQ dataset also shows an improvement on the retriever success with a larger k as well as a drop in abstention. However the task success does not improve much.

Across the datasets the task success does not change as much as the retriever success.

Discussion

While a larger context size k leads to a higher retrieval success, this performance gain does not translate as much to the task success. One explanation could be that the relevant paragraphs are retrieved but appear at a much lower position in the context. Or the generator struggles with a larger context size.

Limitations

- It is not clear when the model exceeds its context window and thus it is not clear if it plateaus at an earlier context window size.

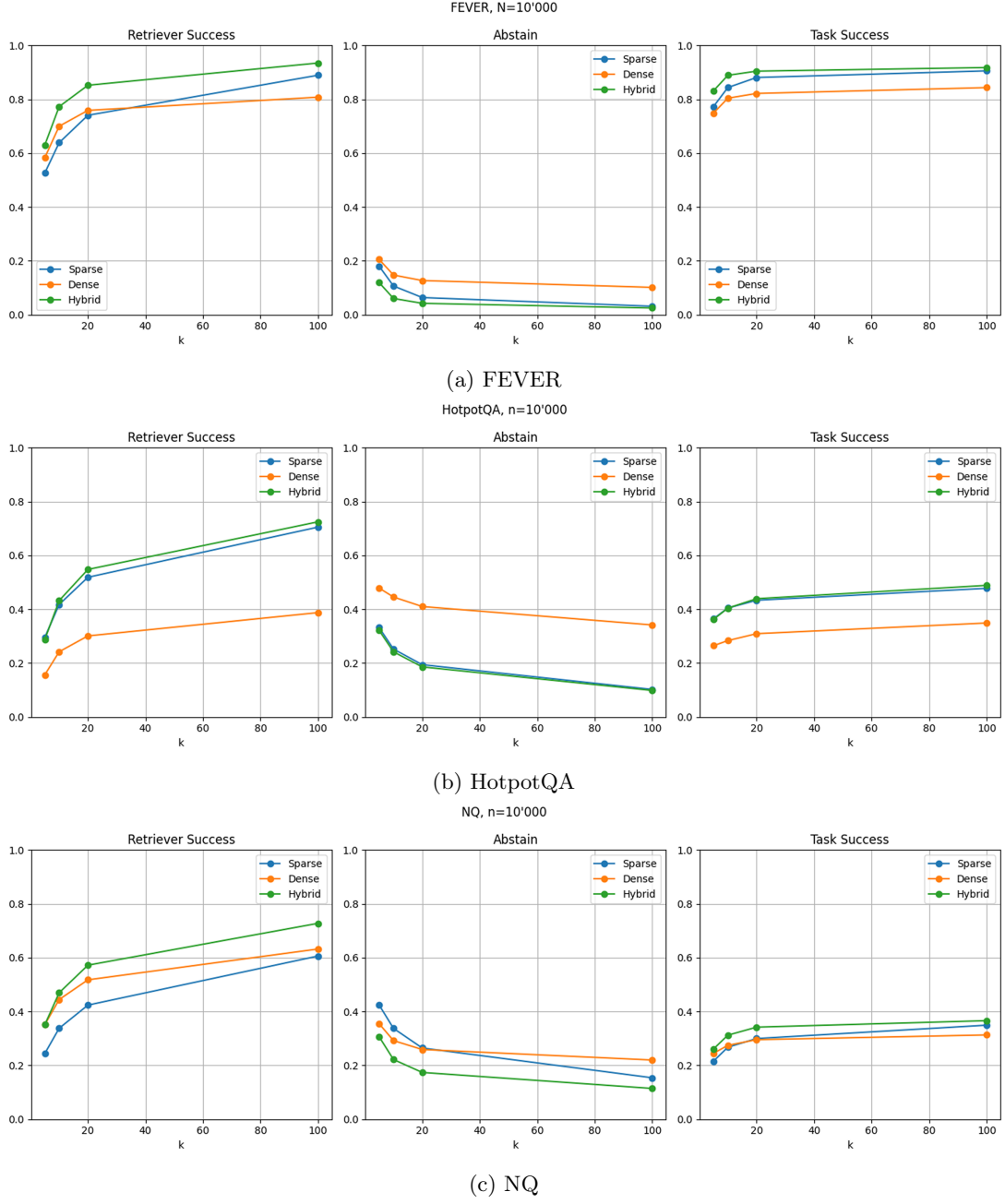


Figure 11: Retriever success, abstention, and task success for $k \in [5, 10, 20, 100]$ across the three datasets FEVER, HotpotQA, and NQ on the three retrieval strategies (Sparse, Dense, Hybrid).

3.7 Compare Context Switch

The oracle intersection experiment in Section 3.5 showed that by removing the "non-relevant" parameters the different retrieval strategies seem to behave very similar, except on the HotpotQA dataset where the dense retriever still performed considerably worse. The idea of this experiment is to switch the "non-relevant" paragraphs between the dense and sparse retriever. This means that both dense and sparse retrieve the paragraphs, the paragraphs that are the same as the reference are kept and the others are exchanged.

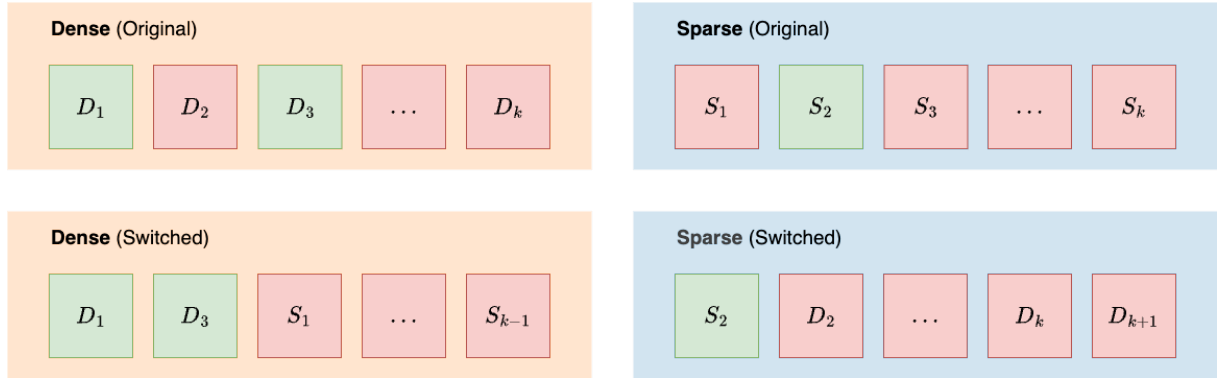


Figure 12: Example of the context switch process on the dense and sparse retrieval strategies.

Figure 12 shows how the context switch is performed. First the dense and sparse strategies retrieve as normal, with a higher k . Then on the dense retriever the "non-relevant" paragraphs are removed, similar to the OI experiment (see Section 3.5). To fill the context back to the size k the "non-relevant" paragraphs from the sparse retrievers, in this case S_1 to $S_k - 1$. Then the same is performed on the sparse retrieval.

In practice this adds two new retrieval strategies:

- **Sparse Switched:** The retrieved relevant paragraphs from the sparse retriever and the "non-relevant" paragraphs from the dense retriever
- **Dense Switched:** The retrieved relevant paragraphs from the dense retriever and the "non-relevant" paragraphs from the sparse retriever

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, $k = 10$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: Sparse and Dense Switched should behave similar since they have the same "non-relevant" paragraphs as additional context.

Results

Table 9 shows the marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are switched and when not across the three dataset FEVER, HotpotQA, and NQ. In this table we compare dense switched with sparse and dense with sparse switched.

Table 9: Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are switched and when not across the three dataset FEVER, HotpotQA, and NQ.

Dataset	Retriever	Switched	$P(R = 1)$	$P(A = 1)$	$P(T = 1)$	$P(G = 1)$
FEVER	Dense	False	0.70	0.15	0.80	0.78
		True	0.70	0.10	0.85	0.72
	Sparse	False	0.64	0.10	0.84	0.68
		True	0.64	0.09	0.86	0.67
HotpotQA	Dense	False	0.24	0.45	0.29	0.59
		True	0.24	0.37	0.31	0.51
	Sparse	False	0.42	0.26	0.41	0.51
		True	0.42	0.27	0.41	0.52
NQ	Dense	False	0.45	0.29	0.28	0.47
		True	0.45	0.28	0.30	0.46
	Sparse	False	0.34	0.34	0.27	0.48
		True	0.34	0.28	0.28	0.41

Between the switched and non-switched retrievers the $P(R = 1)$ stays the same because the paragraphs relevant to achieve retrieval success have not been touched.

In the FEVER dataset the abstention on the dense retriever lowers when the context is switched and is similar to the sparse retrievers abstention. With this the task success increases slightly but the generator success drops from 0.78 to 0.72. On the sparse retriever the switch does not change the abstention, task success or generator success by more than 0.02.

For the HotpotQA dataset the same pattern repeats, on dense switched the abstention decreases, task success increases and generator success decreases while the sparse switched stays close to the numbers of the original sparse retriever.

On the NQ dataset the opposite is shown. While the original dense retriever and dense switched are nearly identical the sparse retriever drops in abstention and generator success.

Table 10 shows the conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.

It shows the same pattern as the previous Table 9 with the dense retrievers showing a difference between the original and switched version on the FEVER and HotpotQA dataset while the sparse retriever does not seem to be affected by this change. And the opposite happening on the NQ dataset.

Discussion

Table 10: Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.

Dataset	Retriever	Switched	$P(A = 1 R1)$	$P(A = 1 R0)$	$P(T = 1 R1, A0)$	$P(T = 1 R0, A0)$
FEVER	Dense	False	0.02	0.44	0.96	0.89
		True	0.03	0.25	0.95	0.90
	Sparse	False	0.03	0.25	0.95	0.92
		True	0.02	0.22	0.95	0.93
HotpotQA	Dense	False	0.04	0.58	0.63	0.44
		True	0.04	0.47	0.63	0.40
	Sparse	False	0.03	0.42	0.65	0.42
		True	0.03	0.44	0.66	0.44
NQ	Dense	False	0.06	0.48	0.48	0.27
		True	0.07	0.45	0.51	0.30
	Sparse	False	0.07	0.48	0.51	0.31
		True	0.07	0.38	0.51	0.30

Switching the "non-relevant" paragraphs does not yield the expected behavior. The experiment showed that the "non-relevant" paragraphs do not have a large enough effect on the generators as previously thought.

Depending on the dataset a different retriever is affected more by the switch of the "non-relevant" paragraphs. This is most likely due to the fact that on each dataset one of the retriever is better suited and when it retries the correct paragraphs the generator is not distracted by the "non-relevant" paragraphs.

Limitations

- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.
- The FEVER dataset may be too easy, because it has only three possible labels.

3.8 Compare Context Shuffle

Previous experiments suggested that the order of the retrieved paragraphs might have an influence on the generators ability to produce the correct answer. With this experiment we want to see if by shuffling the retrieved paragraphs we can observe a difference in the generator behavior.

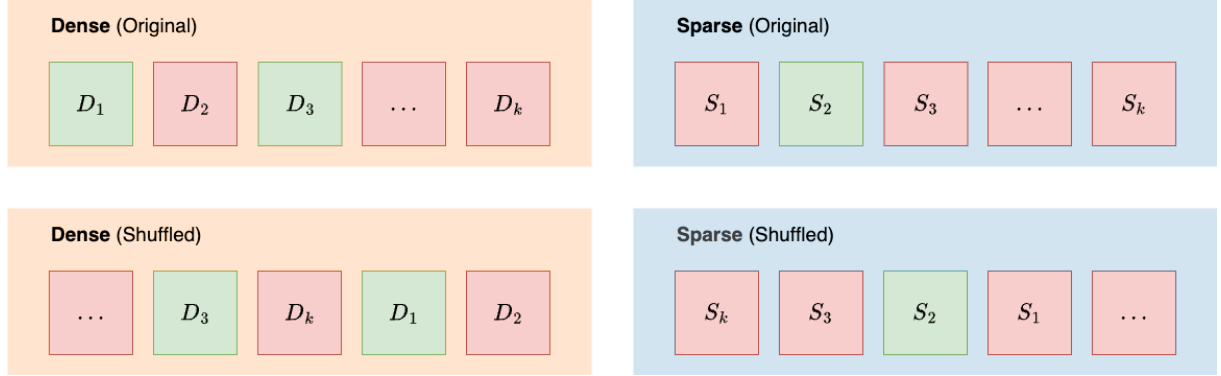


Figure 13: Example of the context shuffle process on the dense and sparse retrieval strategies.

Model: Behavior Mode: $P(R, A, T)$

Datasets: FEVER, NQ, $N = 10'000$

Retriever: Dense, Sparse, $k = 5$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: The order in which the retrieved paragraphs are presented to the generator matters.

Results

Table 11 shows the marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when their retrieval order is shuffled and when not.

Shuffling the retrieved paragraphs does not show an observable effect on the abstention, task success and generator success across the FEVER and NQ dataset for both tested retrieval strategies.

The conditionals reveal slight differences when the retrieval is shuffled, however not larger than 0.01.

Discussion

The generator performance is not shown to be influenced by the order of the retrieved paragraphs. However, this only holds for the tested generator and the small context size of $k = 5$.

Limitations

- A context size of $k = 5$ might be too small to actually see a difference. This should be repeated using $k \in [5, 10, 20]$

Table 11: Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when their retrieval order is shuffled and when not.

Dataset	Retriever	Shuffled	$P(R = 1)$	$P(A = 1)$	$P(T = 1)$	$P(G = 1)$
FEVER	Dense	False	0.58	0.21	0.75	0.73
		True	0.58	0.20	0.75	0.73
	Sparse	False	0.53	0.18	0.77	0.66
		True	0.53	0.18	0.78	0.65
NQ	Dense	False	0.35	0.36	0.24	0.50
		True	0.35	0.36	0.24	0.50
	Sparse	False	0.24	0.43	0.22	0.53
		True	0.24	0.43	0.22	0.53

Table 12: Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.

Dataset	Retriever	Shuffled	$P(A = 1 R1)$	$P(A = 1 R0)$	$P(T = 1 R1, A0)$	$P(T = 1 R0, A0)$
FEVER	Dense	False	0.03	0.46	0.96	0.90
		True	0.03	0.45	0.96	0.91
	Sparse	False	0.03	0.35	0.95	0.93
		True	0.02	0.34	0.95	0.93
NQ	Dense	False	0.06	0.52	0.49	0.26
		True	0.06	0.52	0.49	0.26
	Sparse	False	0.08	0.54	0.52	0.29
		True	0.07	0.55	0.52	0.30

3.9 Compare Context Randomize

Previous experiments have suggested that "non-relevant" paragraphs have an influence on the task and generator success. In this experiment we test whether replacing the "non-relevant" paragraphs with randomly selected paragraphs changes the generator behavior.

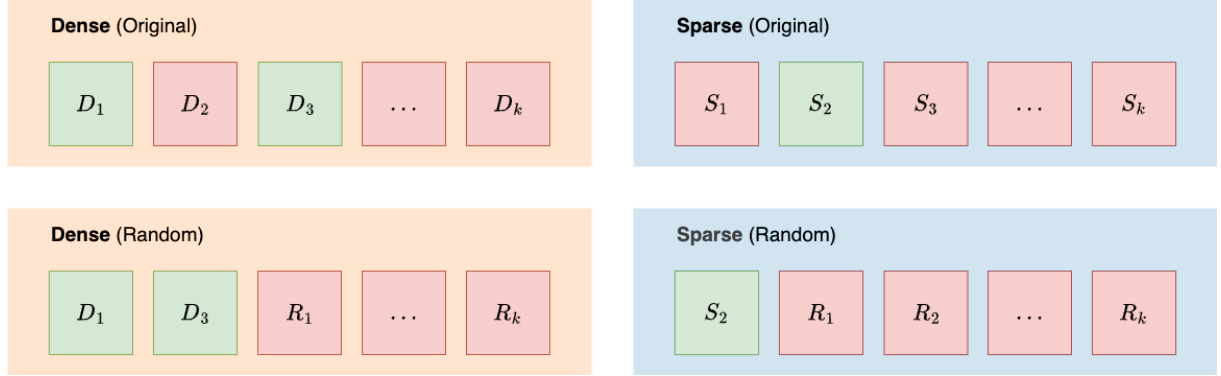


Figure 14: Example of the context randomize process on the dense and sparse retrieval strategies.

In practice this adds two new retrieval strategies:

- Sparse Random: The retrieved relevant paragraphs from the sparse retriever and the rest is filled with random paragraphs
- Dense Random: The retrieved relevant paragraphs from the dense retriever and the rest is filled with random paragraphs

Model: Behavior Mode: $P(R, A, T)$

Datasets: HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, $k = 10$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: Removing the "non-relevant" paragraphs from the retrieval and adding random paragraphs should not change the overall task success.

Results

Table 11 shows the marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are replaced by random paragraphs.

Replacing the "non-relevant" paragraphs with random documents increases the abstention rate across both retrievers and the task success drops 1% while the generator success increases by 6% on the dense retriever and 9% on the sparse retriever.

The conditionals in Table 14 show that the increase in abstention is only when the retrieval fails. Task success actually improves across retrieval failure and success for both retrieval strategies.

Table 13: Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are replaced by random paragraphs.

Dataset	Retriever	Random	$P(R = 1)$	$P(A = 1)$	$P(T = 1)$	$P(G = 1)$
HotpotQA	Dense	False	0.24	0.45	0.29	0.59
		True	0.24	0.51	0.28	0.65
	Sparse	False	0.42	0.26	0.41	0.51
		True	0.42	0.34	0.40	0.60

Table 14: Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.

Dataset	Retriever	Random	$P(A = 1 R1)$	$P(A = 1 R0)$	$P(T = 1 R1, A0)$	$P(T = 1 R0, A0)$
HotpotQA	Dense	False	0.04	0.58	0.63	0.44
		True	0.04	0.67	0.64	0.50
	Sparse	False	0.03	0.42	0.65	0.42
		True	0.03	0.57	0.67	0.51

Discussion

The small drop in task success can be explained by the higher abstention rate on the retrieval failure. This means that when no relevant paragraph is found the context passed to the generator is completely random which increases the chance of abstention which in turn leads to a higher generator success.

The increase in task success on retrieval failure is much stranger, either the number of cases where $P(T = 1|R = 0, A = 0)$ is really small and by chance another similar paragraph is added by the random selection that leads the generator to the correct answer or, much more plausible, some of the questions can be answered no matter the context.

Limitations

- This experiment uses a binary metric for retrieval success. This metric works for FEVER, which requires only one paragraph, and for HotpotQA, which requires both paragraphs to answer the question. It does not work well for NQ.

3.10 HotpotQA Partial Retrieval Success

The Context Swap and Context Random experiments show that on HotpotQA, binary retrieval success is not informative. In HotpotQA, the queries are created by showing two pages to a human annotator. The annotator constructs a question that requires both documents to answer. For this experiment, we therefore split the retriever success into:

- **SUC**: both paragraphs are retrieved
- **ParA**: only the first paragraph is found
- **ParB**: only the second paragraph is found
- **FAIL**: neither paragraph is found

With this partial success measure, we can compare the sparse and dense retriever both where they agree (e.g. both SUC) and where they disagree (all other combinations). This measure also lets us compare the retrieved paragraphs using the Jaccard distance d_J :

$$d_J(A, B) = 1 - J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (13)$$

A high Jaccard distance means the retriever results have little overlap. A low Jaccard distance means the results are similar.

Model: Behavior Mode: $P(R, A, T)$

Datasets: HotpotQA, $N = 10'000$

Retriever: Dense, Sparse, $k = 10$

Retriever Success: Binary

Generator: Qwen 3.5 9B (Qwn)

Generator Success: Binary, Exact Match

Hypothesis: The task success should be similar whether the retrievers partially or fully succeed.

Results

Table 15 shows, for each combination of retriever success (SUC, ParA, ParB, FAIL) between the sparse and dense retriever, the number of queries. For each retriever, the table also shows the retrieval success (as defined in the previous experiments), the abstention rate, the task success, and the conditional task success given abstain=0. The first two columns show the filter conditions (e.g. both sparse and dense retrievers reach full retriever success). The remaining columns show the results per retriever. The table reads as follows:

For all queries where $Sparse = ParB$ and $Dense = SUC$, the sparse retriever reaches $P(T|A = 0) = 0.51$ and the dense retriever reaches $P(T|A = 0) = 0.57$.

For the queries where the sparse retriever succeeds, the abstention rate is lowest and the task success is highest. The same holds for the dense retriever. If the retriever finds only one paragraph, the abstention rate increases and the task success decreases. When the retriever finds only the second paragraph,

Table 15: Comparing sparse and dense retriever on partial retrieval results.

Sparse	Dense	#	Sparse				Dense				d_J
			R	A	T	$P(T A=0)$	R	A	T	$P(T A=0)$	
SUC	SUC	1750	1750	56	1098	0.65	1750	54	1096	0.65	0.69
	ParB	425	425	19	240	0.59	0	164	159	0.61	0.83
	ParA	996	996	29	658	0.68	0	323	362	0.54	0.82
	FAIL	992	992	41	633	0.67	0	786	77	0.37	0.95
ParB	SUC	207	0	42	84	0.51	207	10	113	0.57	0.78
	ParB	250	0	74	77	0.44	0	89	75	0.47	0.77
	ParA	182	0	49	68	0.51	0	99	43	0.52	0.91
	FAIL	420	0	153	132	0.49	0	341	27	0.34	0.95
ParA	SUC	356	0	87	128	0.48	356	24	200	0.60	0.76
	ParB	132	0	35	38	0.39	0	81	28	0.55	0.91
	ParA	1655	0	605	436	0.42	0	619	437	0.42	0.78
	FAIL	1089	0	424	292	0.44	0	927	43	0.27	0.96
FAIL	SUC	111	0	50	24	0.39	111	6	58	0.55	0.83
	ParB	157	0	95	15	0.24	0	69	34	0.39	0.82
	ParA	352	0	220	37	0.28	0	186	57	0.34	0.85
	FAIL	806	0	547	82	0.32	0	644	40	0.25	0.92

the abstention rate increases less and the task success decreases less than when it finds only the first paragraph. However, the retriever finds the second paragraph far less often than the first.

The Jaccard distance is highest when one retriever fails and lowest when both retrievers succeed.

Table 16: Results when sparse and dense retriever agree

Sparse	Dense	#	Sparse				Dense				d_J
			R	A	T	$P(T A=0)$	R	A	T	$P(T A=0)$	
SUC	SUC	1750	1750	56	1098	0.65	1750	54	1096	0.65	0.69
ParB	ParB	250	0	74	77	0.44	0	89	75	0.47	0.77
ParA	ParA	1655	0	605	436	0.42	0	619	437	0.42	0.78
FAIL	FAIL	806	0	547	82	0.32	0	644	40	0.25	0.92

Table 16 filters Table 15 for the cases where both retrievers agree. This table shows that when both retrievers succeed, or partially succeed (ParA and ParB), the task success is nearly identical between retrievers. When both retrievers fail, the Jaccard distance shows that the retrieved paragraphs do not overlap much.

Discussion

The slightly lower task success on ParA may result from the higher number of cases where the retriever finds only the first paragraph. When the retriever finds only one paragraph, the task success drops by 11-13%. When the retriever finds both paragraphs, the task success is high and the abstention rate is low. When the retriever finds no relevant paragraphs, the abstention rate is high and the task success is low. This decomposition of retriever success into partial successes shows that the two retrieval strategies behave in a similar way when they succeed or partially succeed. The sparse retriever appears to retrieve paragraphs that help answer the question even when these paragraphs are not in the reference set.

Overall, these results show that the first paragraph is easier to find than the second. This points to a systemic issue with the dataset. The task success under partial retrieval also shows that at least some questions can be answered using only one of the two required paragraphs.

Limitations

- HotpotQA may be less useful for evaluating multi-hop question answering than expected.

3.11 C4 Retrieval Recall

FEVER showed to be a bit too easy HotpotQA showed that in a lot of cases it requires just the first document (no multiphop) NQ questions are ambiguous or underspecified, and answerable without any retrieval

C4 is a human curated dataset with a wide range of questions. Answers are free text and require between 1 and 25 reference paragraphs

To make use of this dataset the retriever has to be evaluated using Recall@k and the task success is measured using an LLM-based metric (see Section 2.5.2).

Datasets: C4, $N = 598$

Retriever: Dense, Sparse, Hybrid, $k = 10$

Retriever Success: Recall@k

Generator: Qwen 3.5 9B (Qwn)

Generator Success: LLM-based

Judge: Qwen 3.5 9B (Qwn)

Hypothesis: A higher retriever recall should lead to a higher task success but the increase is expected to be different across the retrievers similar to the experiments in Section 3.4.

Results

Figure 15 shows the density map of retriever recall@k and task success as well as the regression line in red fitted on the density map on the C4 dataset across the sparse, dense and hybrid retriever.

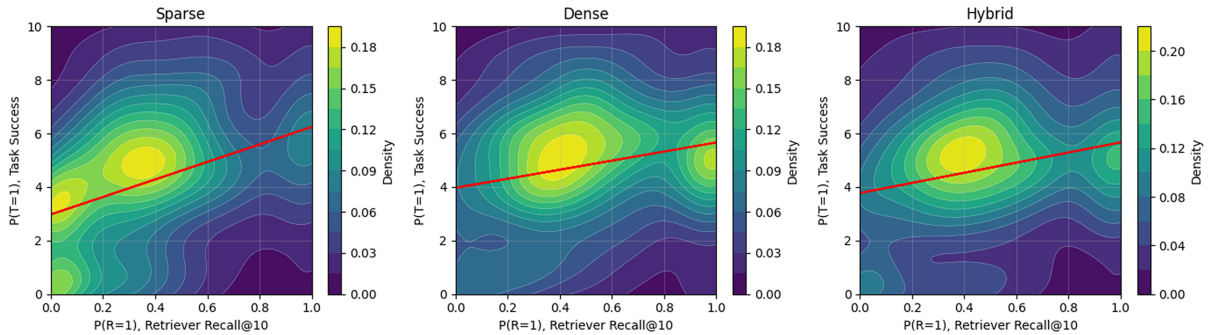


Figure 15: Density map of retriever recall@k and task success as well as the regression line in red fitted on the density map on the C4 dataset across the sparse, dense and hybrid retriever using the Qwen 3.5 9B (Qwn) as a judge model.

Across all retrievers a cluster slightly left of the center is shown. On the sparse retriever there are two other clusters at $P(R = 1) \approx 0$ with one of them at $P(T = 1) \approx 0$ and the other higher around $P(T = 1) \approx 3$. The dense retriever shows another cluster at $P(R = 1) \approx 1$ and $P(T = 1) \approx 5$. For the hybrid retrieval strategy the cluster in the center has the highest density.

For the sparse retriever the average recall@10 is 0.38 and task success at 0.41. For the dense retriever the average recall@10 is 0.52 and task success at 0.48. For the hybrid retriever the average recall@10 is 0.50 and task success at 0.46.

The sparse retriever is shown to have the steepest regression line.

Discussion

The steep regression line on the sparse retriever suggests retrieval fails often but when it succeeds the task success is higher than on the other retrieval strategies. Sparse shows a clear indication that a higher retriever recall leads to a higher task success score which validates that the LLM-based judge provides some useful information. However, due to the uncalibrated nature of the judge it is unclear how certain the task success scores are.

These results indicates that there are some questions where the sparse retriever performs well and these yield a higher task success score. Most likely some of the questions have a good lexical overlap with the relevant paragraphs, if this is not given the sparse retriever performs poorly and the task success score is low.

Limitations

- Automated evaluation using LLM-based judges is limited without human annotation to calibrate the judge.
- Linear regression line might not be suitable, especially on the sparse retriever.

3.12 Compare Generator and Judges

The previous experiment has shown that using a judge to evaluate task success comes with some caveats. To get a feel for the evaluation different generators judge combinations are used. Ideally the judges rank the generators the same, otherwise it becomes difficult to get a clear picture of which generator is actually the best without some form of calibration that most likely requires human annotations.

Datasets: C4, $N = 598$

Retriever: Dense, Sparse, Hybrid, $k = 10$

Retriever Success: Recall@k

Generator: Qwen 3.5 9B (Qwn), Qwn Reasoning, GPT

Generator Success: LLM-based

Judge: Qwen 3.5 9B (Qwn), Qwn Reasoning, GPT

Hypothesis: If the judge models are robust the ranking order of the generators should be the same.

Results

Table 17 shows the task success (with std) for all retriever, generator, judge combinations. The recall is only dependent on the retriever strategy and thus repeats across the rows.

Dense retriever has the highest recall, with hybrid closely following. Sparse performs the worst. Qwn performs the worst across all judges with an average of 4.54 across all judge/retriever combinations, and GPT performs the best with an average task success of 5.91.

All judges rank the Qwn model as the worst. The worst generator ranking is by the GPT model, which ranks the Qwn generator with an average task success of 3.66. Qwn Reasoning generator has the highest task success score from the Qwn Reasoning judge.

Discussion

While the judges agree on which model performs worst, they don't agree on which performs the best. The GPT judge is the most critical and gives the lowest task success scores.

Limitations

- Automated evaluation using LLM-based judges is limited without human annotation to calibrate the judge.

Table 17: Recall and Task Success Scores across different generator, judge, retriever combinations.

Generator	Judge	Retriever	Recall	Task Success
Qwn	Qwn	Dense	0.52	4.76 ± 2.44
		Hybrid	0.50	4.56 ± 2.44
		Sparse	0.38	4.13 ± 2.66
	Qwn Reasoning	Dense	0.52	5.77 ± 2.47
		Hybrid	0.50	5.62 ± 2.39
		Sparse	0.38	4.97 ± 2.73
	GPT	Dense	0.52	4.01 ± 2.19
		Hybrid	0.50	3.77 ± 2.19
		Sparse	0.38	3.22 ± 2.24
Qwn Reasoning	Qwn	Dense	0.52	5.38 ± 2.56
		Hybrid	0.50	5.46 ± 2.62
		Sparse	0.38	4.85 ± 2.70
	Qwn Reasoning	Dense	0.52	7.07 ± 2.53
		Hybrid	0.50	7.11 ± 2.46
		Sparse	0.38	6.28 ± 2.61
	GPT	Dense	0.52	4.67 ± 2.63
		Hybrid	0.50	4.68 ± 2.53
		Sparse	0.38	4.01 ± 2.60
GPT	Qwn	Dense	0.52	6.30 ± 2.41
		Hybrid	0.50	6.32 ± 2.44
		Sparse	0.38	5.76 ± 2.69
	Qwn Reasoning	Dense	0.52	6.51 ± 2.62
		Hybrid	0.50	6.59 ± 2.49
		Sparse	0.38	5.87 ± 2.92
	GPT	Dense	0.52	5.40 ± 2.34
		Hybrid	0.50	5.49 ± 2.47
		Sparse	0.38	4.95 ± 2.66

3.13 Lexical Overlap

The experiments on the C4 datasets showed a clear difference between the dense and sparse retriever. In this experiment we want to find some attribute that is indicative of this difference. We assume that due to the retrieval nature of the sparse retriever it performs better on questions that have a high lexical overlap (see Section 2.5.4) with the relevant paragraphs.

We compare the computed lexical overlap to the retrieval recall@k and the task success score in a scatter plot. To fit a regression line we choose to use isotonic regression because we expect the relation to be monotonic but not necessarily linear.

Isotonic regression is defined as:

$$\min_{\hat{y}} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad \text{where} \quad \hat{y}_1 \leq \hat{y}_2 \leq \dots \leq \hat{y}_n \quad (14)$$

It fits a free-form, non-decreasing step function to the sequence of data points from the lexical overlap and recall@k or task success score while minimizing the mean squared error $(y_i - \hat{y}_i)^2$. The constraint $\hat{y}_1 \leq \hat{y}_2 \leq \dots \leq \hat{y}_n$ keeps the fitted function monotonic.

Datasets: C4, $N = 598$

Retriever: Dense, Sparse, Hybrid, $k = 10$

Retriever Success: Recall@k

Generator: Qwen 3.5 9B (Qwn)

Generator Success: LLM-based

Judge: Qwen 3.5 9B (Qwn)

Hypothesis: Questions that have a high lexical overlap with the reference paragraphs are found easier by the sparse retriever and yield a higher task success score.

Results

Figure 16 shows the lexical overlap between the query and the reference paragraphs compared to (left) the recall@10 and (right) the task success score. The metric pairs are plotted as a scatter plot and an isotonic regression line is fitted to them using repeated resampling to estimate the 95% confidence intervals. Additionally, the spearman ρ correlation is calculated for both of the retrievers.

The lexical overlap compared to the recall@10 shows that the sparse retriever has a moderate correlation of $\rho = 0.33$ but recall seems to increase only after a lexical overlap > 0.2 . The regression of the sparse retriever stays below the dense retriever throughout. The dense retriever shows a negligible correlation of $\rho = 0.1$.

On the comparison with the task success rate the correlation on the sparse retriever is weak $\rho = 0.19$ while the correlation of the dense retriever is negligible. The regression looks nearly flat on both retrieval strategies.

Discussion

The moderate correlation on the sparse retriever is to be expected due to the nature of the retrieval strategy. The comparison with the task success score indicates that there is not a big correlation between

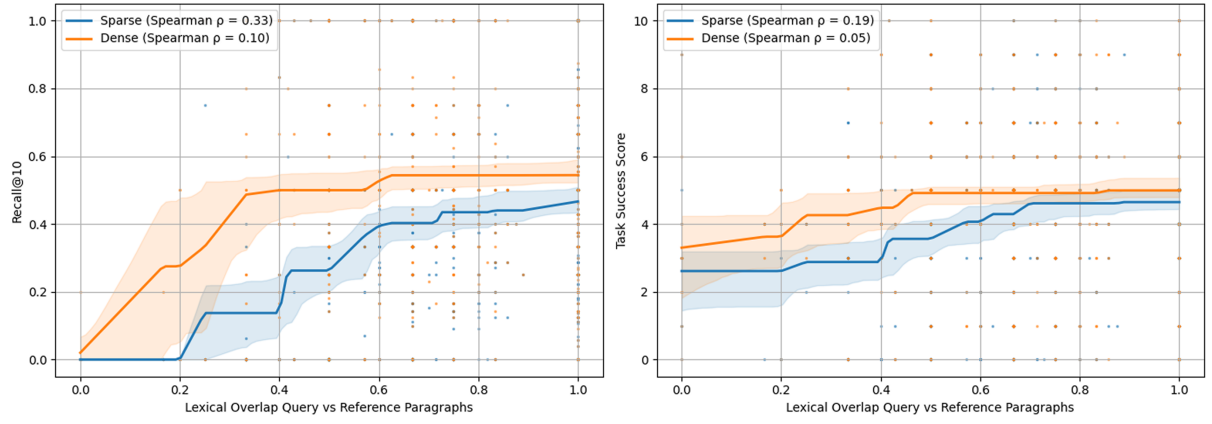


Figure 16: Relationship between the lexical overlap of the query and reference paragraphs and the Recall@10 (left) and the task success score (right). With a regression line fitted using isotonic regression.

the lexical overlap and the score. Overall it seems like that the lexical overlap does only partially explains the difference in the retrieval strategies.

Limitations

- The lexical overlap definition does not take into account the frequency of the individual words nor the position within the texts.

4 Discussion

The results in Section 3 can be summarized more formally. An evaluation is a triple (D, M, J) of a dataset D , a set of metrics M , and, where applicable, a judge J . A RAG system, in the terms of Section 2.2, is a configuration of the System or Behavior model, e.g. (Q, R, G) or (R, A, T, G) , instantiated with a concrete retriever and generator. Every result reported in this thesis is therefore the outcome of evaluating one RAG configuration with one evaluation setup, $(D, M, J) \times (R, G)$.

The experiments in Sections 3.3 through 3.9 show that findings obtained under one such pairing do not translate to another: neither across datasets, nor across retrievers, nor across generators, even when the retrieval success itself is held constant.

This has a practical consequence, which can be illustrated with a case drawn from Sections 2.2.3 and 3.2. Consider a RAG system evaluated with $(R = \text{Dense}, G = \text{Qwn})$ on $(D = \text{FEVER}, M = \text{Binary})$. Suppose the retriever is improved by switching to Hybrid, since Hybrid is found to achieve a higher retrieval success $P(R = 1)$ on this dataset. After deploying this change, the generator success $P(G = 1)$ drops, even though retrieval improved. Table 8 shows why: switching from Dense to Hybrid does increase $P(R = 1)$, but it also changes the distribution of queries on which retrieval fails. The generator abstains less often on these new retrieval-failure cases than it did on the ones produced by the Dense retriever, so $P(A = 1 \mid R = 0)$ drops, and with it $P(G = 1)$, despite the retriever itself being "better" by the metric used to select it. This is exactly the divergence between task success and generator success discussed above: an improvement measured on one component, in isolation, can degrade the behavior of the system as a whole once its interaction with the rest of the pipeline is taken into account.

The point of this example is not that Hybrid is a worse retriever than Dense, nor that retrieval improvements are undesirable. It is that a retriever cannot be evaluated, or compared to an alternative, independently of the generator, the abstention behavior it induces, and the dataset it is tested on. An evaluation result is only valid for the exact $(D, M, J) \times (R, G)$ tuple it was produced under. Extrapolating it to a new retriever, generator, or dataset is not supported by the results in this thesis, and, as the example shows, can be actively misleading.

Across the experiments in Section 3, a consistent pattern emerges: configurations that appear equivalent under a single marginal metric behave differently once the pipeline is examined more closely. Two retrievers with the same retrieval success $P(R = 1)$ do not necessarily yield the same task success (Section 3.3), the relationship between retrieval and task success differs across datasets (Sections 3.3 and 3.5), and even within a single dataset, retrieval success collapses questions with very different difficulty into the same binary outcome (Section 3.10). A single reported number, e.g. Recall@k or task success, is therefore not a sufficient description of a RAG system’s behavior. It is a projection of a higher-dimensional outcome that we attempted to model explicitly through the *System* and *Behavior* models.

What underlies this gap remains only partially explained. Non-relevant paragraphs, i.e. paragraphs not marked as reference but still retrieved, sometimes improve task success (Sections 3.3 and 3.5), yet directly manipulating them through the context switch experiment did not reproduce the expected effect (Section 3.7), and replacing them with random paragraphs instead improved generator success through higher abstention rather than through better answers (Section 3.9). Paragraph order, in turn, showed no measurable effect at $k = 5$ (Section 3.8). Taken together, these results narrow down but do not identify the confounding factor responsible for the divergence between retrievers with equal recall. The behavior of a RAG system is not fully explained by which paragraphs are retrieved, and simple interventions on the retrieved context are insufficient to isolate the cause.

The distinction between task success and generator success, introduced in the *Behavior* model (Section 2.2), is the most direct methodological payoff in our work. Section 3.2 shows that two generators can reach near-identical task success while differing substantially in generator success, driven entirely by how often they abstain under retrieval failure. Evaluating a RAG system by task success alone, as most of the frameworks discussed in Section 1.1 do, would treat these two generators as equivalent, even though only one of them behaves as a user would expect when retrieval fails.

The results obtained on C4 using an LLM-based judge (Sections 3.11 through 3.12) should be weighted differently than the binary results on FEVER, NQ, and HotpotQA. Section 3.12 shows that judges agree on which generator performs worst but disagree on which performs best, and that absolute task success scores shift depending on which model is used as judge. Since these scores were not calibrated against human annotation, any ranking of retrievers or generators derived from them carries an additional, unquantified source of uncertainty that the binary metric results do not.

This connects back to the frameworks discussed in Section 1.1. Prior work such as RAGAs, ARES, and CRUD-RAG evaluate retrieval and generation along multiple dimensions but treat these dimensions independently, without modeling the statistical dependency between them. The results in this thesis show that this dependency is not negligible: retrieval and generation performance are coupled in ways that a per-dimension evaluation does not surface, which is precisely the gap the *System* and *Behavior* models were designed to address.

In practice, this suggests that comparing RAG configurations through a single leaderboard-style metric is not helpful. A retriever with higher recall, or a generator with higher task success, is not necessarily the better choice once abstention behavior, dataset-specific confounds, and judge reliability are taken into account. A defensible evaluation practice would need to report conditional and behavioral metrics alongside marginal ones, and would need to treat any such comparison as specific to the exact combination of dataset, retriever, and generator under test, rather than as a general statement about which component is best.

4.1 Limitations

RAG evaluation is difficult and requires a lot of assumptions which leads to unavoidable limitations. Some limitations are specific to a single experiment and are already noted where they occur in Section 3.

Model

The *System* and *Behavior* models reduce the RAG pipeline to a few binary nodes (Q, R, G) and (R, A, T, G) . This is a simplification, modern implementations include rerankers, query rewriting, multi-turn context, routing, etc. Success and failure is modeled as binary throughout, despite some of the underlying metrics being continuous or ordinal, e.g. Recall@k and the LLM-judge score. Reducing these to binary success discards information and requires a threshold that is not part of the model itself. The model does not capture query difficulty as a variable. Every query is treated as exchangeable within a dataset, although Section 3.10 shows that this assumption does not hold even within a single dataset.

Dataset

The usefulness of the chosen benchmarks for RAG evaluation is questionable. This is not due to a lack of consideration but a lack of availability. Each QA dataset has its own disadvantages: some are too easy,

which leads to a high task success even on retrieval failure, and some are too difficult to judge the answer against.

FEVER: 3-class label prediction understates real generation difficulty, and generator success stays high even under retrieval failure.

HotpotQA: the multi-hop framing partially breaks down, as shown by splitting retrieval success into partial success (see Section 3.10). Many questions are answerable from only the first of the two required paragraphs, and the first paragraph is also systematically easier to retrieve than the second.

NQ: questions are frequently ambiguous or underspecified, and in some cases answerable without any retrieval at all, which undermines retrieval success as a meaningful predictor of task success on this dataset.

C4: higher quality, more abstractive questions that do not fit the binary retrieval and task success model, making evaluation on this dataset difficult to compare to the other three. The dataset is also small ($N = 598$), and its free-text answers require an LLM-based judge, introducing a new, unquantified source of uncertainty.

Metric

Binary retrieval success cannot express the partial success required to properly evaluate HotpotQA. Recall@ k ignores the ranking position of retrieved paragraphs, though the context shuffle experiment (see Section 3.8) suggests ordering has little effect on the final task success, at least at $k = 5$.

Free-text evaluation is fundamentally harder and requires a separate, less validated LLM-based judge that would need human annotations for calibration. Since the judge would need to be recalibrated after every change to the RAG system, this cost is incurred repeatedly rather than once. The generator/judge experiment (see Section 3.12) shows systematic disagreement between judges, consistent with this lack of calibration.

Experiments

The probabilistic retriever experiment places correct paragraphs at the start of the retrieved context, introducing a positional confound that is not disentangled from the retrieval success rate itself (see Section 3.3). The context size experiment cannot distinguish between retrieval saturation and the generator’s context window being exceeded as the cause of the observed plateau (see Section 3.6).

The context shuffle experiment is undersized: it was run at a single context size ($k = 5$) with a single generator, and should be repeated at larger k before concluding that paragraph order has no effect (see Section 3.8). The probabilistic retriever sweep is computationally expensive, requiring approximately 22 hours per retriever strategy, which restricted the experiment to a single dataset and generator pair (see Section 3.3).

General

The evaluated RAG system is single-turn only. It does not include a rerankers, query rewriting, multi-turn context, routing, or other elements common in modern RAG deployments.

4.2 Future Work

The limitations discussed above point to several directions for future work.

Extended Model

The *System* and *Behavior* models used in this project are deliberately minimal, reduced to a few binary nodes. An extended model that supports additional pipeline components, such as a reranker or query rewriting, and that represents success as an ordinal or continuous quantity rather than a binary outcome, so that metrics like Recall@k or an LLM-judge score can be incorporated directly rather than thresholded into a binary success. Such a model could also represent query difficulty as an explicit variable, rather than treating all queries within a dataset as exchangeable, addressing the partial-success cases identified in Section 3.11.

Clean QA Benchmark

No existing dataset used in this project was designed for the kind of evaluation this framework requires. A benchmark constructed specifically for RAG evaluation would need to annotate not only a reference answer, but also query difficulty, whether a question is extractive or abstractive, and which parts of the reference paragraphs are actually necessary to produce the answer.

Human Annotation for Calibration

The judge disagreement observed in Section 3.12 shows that LLM-as-a-judge evaluation cannot currently be trusted without calibration against human annotation. Future work should establish this calibration for a fixed evaluation setup and study how much it degrades as the underlying RAG configuration changes, which would give a direct, empirical estimate of how often recalibration is actually necessary in practice, rather than assuming it is required after every change.

Isolating the confounding factor

The interventions in Sections 3.4 to 3.9 narrow down but do not identify why retrievers with equal recall can yield different task success. Future work could isolate this more directly, for example by controlling for the semantic similarity between non-relevant and relevant paragraphs, or by tracking which specific retrieved paragraphs the generator attends to when producing an answer, rather than manipulating the retrieved set as a whole.

4.3 Conclusion

In this project we set out to evaluate RAG systems and, in doing so, to understand why this evaluation is difficult in the first place. The central finding is that evaluation results do not generalize: a finding obtained under one combination of dataset, metric, judge, retriever, and generator does not transfer to another, even when a metric such as retrieval success is held fixed. This is not merely a measurement artifact, it reflects a real coupling between retrieval and generation through abstention, as shown by the Dense-to-Hybrid example in Section 4. An improvement on one component can degrade the system as a whole, because it changes which queries fail and how the generator behaves under those new failures.

The distinction between task success and generator success, introduced in the *Behavior* model (Section 2.2), is the clearest expression of this coupling. Two generators can reach identical task success while differing substantially in how often they abstain under retrieval failure, meaning task success alone rewards generators that answer regardless of retrieval outcome. What exactly drives this coupling remains only partially explained. Several targeted interventions (Sections 3.3 to 3.9) narrow down but do not isolate a single confounding factor, and this open question is itself a finding rather than a gap to be dismissed. This uncertainty compounds with the reliability of the metrics used to measure it. Uncalibrated LLM judges disagree on ranking (Section 3.12), adding a further, unquantified source of error on top of the non-generalization result.

These results extend prior work such as RAGAs, ARES, and CRUD-RAG (Section 1.1), which evaluate retrieval and generation as independent dimensions. Our results show this independence assumption does not hold. The *System* and *Behavior* models proposed here are not a solution to this problem but a diagnostic tool that makes the dependency, and the resulting non-generalization, explicit and quantifiable.

Evaluating Retrieval Augmented Generation (RAG) systems is challenging and very tedious.

List of Figures

1	Overview of a RAG system including the two main components <i>Retriever</i> and <i>Generator</i> that coordinate the data flow within the system.	4
2	The <i>System</i> model captures how information flows through the system: the query Q is passed to the generator G together with the output of the retriever R	6
3	The <i>Behavior</i> model shows the possible behavioral states of the system, with R denoting retrieval success, A abstention, T task success, and G generator success.	7
4	Interactive tool for exploring error propagation in an example RAG pipeline consisting of Query, Retriever, and Generator components. Adjusting the conditional probabilities (left inputs of each node) automatically recomputes the marginal probabilities (bottom of each node), illustrating how estimation errors propagate through the system.	10
5	Class diagram of the RAG implementation.	20
6	Class diagram of the different retriever implementations.	22
7	Probabilistic Retriever (PR) with the three base retrieval strategies (Dense, Sparse, Hybrid) showing the relationship between retrieval success and task success. The changes in the PR are shown as a line plot with an error bar and the results from the base retriever are shown as a dot with error bars.	32
8	Approximated relation between retriever success $P(R = 1)$ and task success $P(T = 1)$ across multiple datasets (FEVER, HotpotQA, NQ) and multiple retriever strategies (Sparse, Dense, Hybrid, Oracle, Empty).	34
9	Example of the Oracle Intersection (OI) process on the dense and sparse retrieval strategies.	36
10	Approximated relation between retriever success $P(R = 1)$ and task success $P(T = 1)$ across multiple datasets (FEVER, HotpotQA, NQ) and multiple retriever strategies adjusted with the Oracle Intersection (OI) process (Sparse, Dense, Hybrid, Oracle, Empty).	37
11	Retriever success, abstention, and task success for $k \in [5, 10, 20, 100]$ across the three datasets FEVER, HotpotQA, and NQ on the three retrieval strategies (Sparse, Dense, Hybrid).	40
12	Example of the context switch process on the dense and sparse retrieval strategies.	41
13	Example of the context shuffle process on the dense and sparse retrieval strategies.	44
14	Example of the context randomize process on the dense and sparse retrieval strategies.	46
15	Density map of retriever recall@k and task success as well as the regression line in red fitted on the density map on the C4 dataset across the sparse, dense and hybrid retriever using the Qwen 3.5 9B (Qwn) as a judge model.	51

16	Relationship between the lexical overlap of the query and reference paragraphs and the Recall@10 (left) and the task success score (right). With a regression line fitted using isotonic regression.	56
----	--	----

List of Tables

1	Example from the Fact Extraction and VERification (FEVER) dataset.	12
2	Example from the Natural Questions (NQ) dataset.	13
3	Example from the HotpotQA (HotpotQA) dataset.	13
4	Example instance #2 from the Catechism of the Catholic Church (C4) dataset.	14
5	Evaluation Checklist	16
6	Retrieval and generator binary evaluation across 27 RAG configurations for 10'000 queries per dataset. The evaluation shows the marginal success probabilities for retrieval success $P(R = 1)$ and generator success $P(G = 1)$. It also shows the conditional generator success given retrieval success $P(G = 1 R = 1)$ and given retrieval failure $P(G = 1 R = 0)$	25
7	Retrieval and generator binary evaluation across 27 RAG configurations for 10'000 queries per dataset. Evaluation shows marginal success probabilities for retrieval success $P(R = 1)$, abstention $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$	28
8	Retrieval and generator binary evaluation across 27 RAG configurations for 10'000 queries per dataset. Evaluation shows the conditional success probabilities for abstention given retrieval failure $P(A = 1 R = 0)$ or retrieval success $P(A = 1 R = 1)$ as well as task success given non-abstention and retrieval failure $P(T = 1 R = 0, A = 0)$ and retrieval success $P(T = 1 R = 1, A = 0)$	29
9	Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are switched and when not across the three dataset FEVER, HotpotQA, and NQ.	42
10	Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.	43
11	Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when their retrieval order is shuffled and when not.	45
12	Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.	45

13	Marginal probabilities for retriever success $P(R = 1)$, abstain $P(A = 1)$, task success $P(T = 1)$, and generator success $P(G = 1)$ for the dense and sparse retriever when the "non-relevant" paragraphs are replaced by random paragraphs.	47
14	Conditional probabilities for abstention given retrieval success and retrieval failure, as well as the task success given retrieval success and retrieval failure when the generator did not abstain.	47
15	Comparing sparse and dense retriever on partial retrieval results.	49
16	Results when sparse and dense retriever agree	49
17	Recall and Task Success Scores across different generator, judge, retriever combinations. .	54

References

- [1] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (N. Aletras and O. De Clercq, eds.), (St. Julians, Malta), pp. 150–158, Association for Computational Linguistics, Mar. 2024.
- [2] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "ARES: An automated evaluation framework for retrieval-augmented generation systems," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (K. Duh, H. Gomez, and S. Bethard, eds.), (Mexico City, Mexico), pp. 338–354, Association for Computational Linguistics, June 2024.
- [3] D. Rau, H. Déjean, N. Chirkova, T. Formal, S. Wang, S. Clinchant, and V. Nikoulina, "BERGEN: A benchmarking library for retrieval-augmented generation," in *Findings of the Association for Computational Linguistics: EMNLP 2024* (Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, eds.), (Miami, Florida, USA), pp. 7640–7663, Association for Computational Linguistics, Nov. 2024.
- [4] S. Wang, E. Khramtsova, S. Zhuang, and G. Zuccon, "Feb4rag: Evaluating federated search in the context of retrieval augmented generation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 763–773, 2024.
- [5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 9459–9474, Curran Associates, Inc., 2020.
- [6] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. De Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240, PMLR, 17–23 Jul 2022.

- [7] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, “Evaluation of retrieval-augmented generation: A survey,” in *Big Data* (W. Zhu, H. Xiong, X. Cheng, L. Cui, Z. Dou, J. Dong, S. Pang, L. Wang, L. Kong, and Z. Chen, eds.), (Singapore), pp. 102–120, Springer Nature Singapore, 2025.
- [8] Y. Tang and Y. Yang, “Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries,” *arXiv preprint arXiv:2401.15391*, 2024.
- [9] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39. Cambridge University Press Cambridge, 2008.
- [10] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, “Judging llm-as-a-judge with mt-bench and chatbot arena,” in *Advances in Neural Information Processing Systems* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), vol. 36, pp. 46595–46623, Curran Associates, Inc., 2023.
- [11] J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking large language models in retrieval-augmented generation,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 17754–17762, Mar. 2024.
- [12] Y. Lyu, Z. Li, S. Niu, F. Xiong, B. Tang, W. Wang, H. Wu, H. Liu, T. Xu, and E. Chen, “Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models,” *ACM Trans. Inf. Syst.*, vol. 43, Jan. 2025.
- [13] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, V. Plachouras, T. Rocktäschel, and S. Riedel, “KILT: a benchmark for knowledge intensive language tasks,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, eds.), (Online), pp. 2523–2544, Association for Computational Linguistics, June 2021.
- [14] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “FEVER: a large-scale dataset for fact extraction and VERification,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (M. Walker, H. Ji, and A. Stent, eds.), (New Orleans, Louisiana), pp. 809–819, Association for Computational Linguistics, June 2018.
- [15] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, “Natural questions: A benchmark for question answering research,” *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 452–466, 2019.
- [16] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, “HotpotQA: A dataset for diverse, explainable multi-hop question answering,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, eds.), (Brussels, Belgium), pp. 2369–2380, Association for Computational Linguistics, Oct.-Nov. 2018.
- [17] P. v. Däniken, M. Cieliebak, and J. Deriu, “C4: A multilingual benchmark for retrieval-augmented generation based on the catechism of the catholic church and its compendium,” in *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)* (S. Piperidis, N. Bel, H. van den Heuvel, N. Ide, S. Krek, and A. Toral, eds.), (Palma, Mallorca, Spain), pp. 7446–7456, European Language Resources Association (ELRA), May 2026.

- [18] Y. Lee, J. Kim, J. Kim, H. Cho, J. Kang, P. Kang, and N. Kim, “CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing* (C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, eds.), (Suzhou, China), pp. 15771–15798, Association for Computational Linguistics, Nov. 2025.
- [19] S. Sturua, I. Mohr, M. K. Akram, M. Günther, B. Wang, M. Krimmel, F. Wang, G. Mastrapas, A. Koukounas, A. Koukounas, N. Wang, and H. Xiao, “jina-embeddings-v3: Multilingual embeddings with task lora,” 2024.
- [20] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, “The faiss library,” 2024.
- [21] A. Hernández-Cano, A. Hägele, A. H. Huang, A. Romanou, A.-J. Solergibert, B. Pasztor, B. Messmer, D. Garbaya, E. F. Ďurech, I. Hakimi, J. G. Giraldo, M. Ismayilzada, N. Foroutan, S. Moalla, T. Chen, V. Sabolčec, Y. Xu, M. Aerni, B. AlKhamissi, I. A. Marinas, M. H. Amani, M. Ansaripour, I. Badanin, H. Benoit, E. Boros, N. Browning, F. Bösch, M. Böther, N. Canova, C. Challier, C. Charmillot, J. Coles, J. Deriu, A. Devos, L. Drescher, D. Dzenhaliou, M. Ehrmann, D. Fan, S. Fan, S. Gao, M. Gila, M. Grandury, D. Hashemi, A. Hoyle, J. Jiang, M. Klein, A. Kucharavy, A. Kucherenko, F. Lübeck, R. Machacek, T. Manitaras, A. Marfurt, K. Matoba, S. Matrenok, H. Mendonça, F. R. Mohamed, S. Montariol, L. Mouchel, S. Najem-Meyer, J. Ni, G. Oliva, M. Pagliardini, E. Palme, A. Panferov, L. Paoletti, M. Passerini, I. Pavlov, A. Poiroux, K. Ponkshe, N. Ranchin, J. Rando, M. Sauser, J. Saydaliev, M. A. Sayfiddinov, M. Schneider, S. Schuppli, M. Scialanga, A. Semenov, K. Shridhar, R. Singhal, A. Sotnikova, A. Sternfeld, A. K. Tarun, P. Teiletche, J. Vamvas, X. Yao, H. Z. A. Ilic, A. Klimovic, A. Krause, C. Gulcehre, D. Rosenthal, E. Ash, F. Tramèr, J. VandeVondele, L. Veraldi, M. Rajman, T. Schulthess, T. Hoefler, A. Bosselut, M. Jaggi, and I. Schlag, “Apertus: Democratizing Open and Compliant LLMs for Global Language Environments.” <https://arxiv.org/abs/2509.14233>, 2025.
- [22] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. bastien Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehari, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Pöder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed,

- V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot, “Gemma 3 technical report,” 2025.
- [23] Qwen Team, “Qwen3.5: Towards native multimodal agents,” February 2026.