Scaling Reasoning and Ranking Feedback for Multi-Hop Question Answering

From Many Thoughts to One Truth

Master Thesis

Tushar Rajiv Kumar

Delft University of Technology

TU Delft

Cover image generated by ChatGPT

Scaling Reasoning and Ranking Feedback for Multi-Hop Question Answering

From Many Thoughts to One Truth

by

Tushar Rajiv Kumar

Master of Science in Computer Science

Committee Chair:	Dr. Avishek Anand
First Core Member:	Dr. Sicco Verwer
Daily Supervisor:	Dr. Venktesh Viswanathan
Project Duration:	December 2024 - August 2025
Faculty:	Faculty of Computer Science, TU Delft

TU Delft

Preface

This thesis marks the conclusion of my Master's in Computer Science at TU Delft and the start of a deeper interest in the intersection of artificial intelligence and real-world reasoning. Over the past year, I explored how large language models (LLMs) can address problems that demand understanding, reasoning, and synthesis of complex information. The project bridged core ideas in machine learning with broader questions about how intelligent systems interpret and respond to human language, making it both technically challenging and intellectually rewarding.

The work presented here is the product of months of research, experiments, setbacks, and moments of clarity. It taught me to ask better questions, think critically about system design, and manage a long-term project with focus and adaptability. I also gained a deeper appreciation for the complexity of natural language understanding and the challenges of aligning LLMs with human notions of truth and evidence.

I am deeply grateful to my supervisors, Dr. Avishek Anand and Dr. Venktesh Viswanathan, for their insightful feedback, open-mindedness, and steady encouragement, which gave me the confidence to explore bold ideas. I would also like to thank the Web Information Systems group at TU Delft for the fresh perspectives and collaborative spirit that helped shape this work.

A heartfelt thanks to my family for their unwavering support and to my friends for keeping me balanced and motivated throughout this journey. Finally, a quiet nod to the humble machine that ran every experiment and safeguarded every model checkpoint. Its reliability made more of a difference than I care to admit.

This thesis is the result of learning, persistence, and the support of many people. While this chapter of my academic life is closing, the questions and ideas explored here will continue to guide my research and thinking in the years ahead.

Tushar Rajiv Kumar
Delft, August 2025

Abstract

Open-domain question answering (ODQA) often requires integrating evidence from multiple sources and reasoning across several steps. While recent work has made progress on retrieval and reasoning independently, their combined optimization remains challenging. Standard retrieval methods may fail to surface all relevant documents, while reasoning models can generate confident but incorrect answers when evidence is incomplete, noisy, or inconsistent. This limits both accuracy and the reliability of model predictions, particularly for multi-hop and compositional questions.

This thesis explores strategies to jointly enhance retrieval and reasoning for ODQA. On the retrieval side, we introduce a dynamic answer frontier mechanism that prioritizes candidate documents based on semantic consistency across multiple generated answers, guiding iterative document expansion over a retrieval graph. This consistency-driven approach improves recall by promoting documents aligned with the most reliable reasoning traces. On the reasoning side, we apply test-time scaling (TTS), generating multiple candidate answers per question and training a verifier model to select the most trustworthy one. The verifier evaluates both semantic correctness and grounding in retrieved evidence, mitigating the effects of misleading or irrelevant documents.

We evaluate the proposed approach on two challenging ODQA benchmarks, MuSiQue and 2WikiMultiHopQA, which require complex multi-hop reasoning and resist shortcut solutions. Experimental results show that our method improves evidence recall and downstream answer accuracy over strong baselines, including standard retrieval pipelines and semantic uncertainty-based re-ranking methods. Qualitative analysis reveals better handling of compositional queries, including temporal comparisons and multi-hop relational reasoning, along with improved resilience to noisy retrievals and reduced divergence from relevant evidence. The study also identifies remaining challenges, such as reliance on agreement as a proxy for correctness and the computational cost of TTS, pointing to future directions involving principled uncertainty measures, end-to-end feedback integration, and efficiency improvements for exploring larger reasoning spaces. Together, these findings underscore the value of integrating semantic consistency-driven retrieval with verifier-guided reasoning selection to advance robustness and trustworthiness in complex ODQA systems.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Background and Motivation

1.1.1. Open-Domain Question Answering and Its Challenges

Open-domain question answering (ODQA) systems answer natural-language questions by searching a large, unstructured corpus (e.g., the web or Wikipedia) for evidence and then generating an answer. A typical pipeline retrieves candidate passages and reasons over them to produce a final answer. Such traditional QA systems have predominantly focused on answering simple, factoid-based questions, where answers typically reside within a single document [75, 76, 54]. In recent years, large language models (LLMs) have pushed ODQA performance forward by improving both retrieval and generation. However, as real-world information needs become increasingly complex, users pose queries requiring the integration and synthesis of evidence from multiple documents—a task known as Complex QA or multi-hop QA. This paradigm shift is illustrated by recent benchmarks such as HotpotQA [89] and WikiHop [81], which explicitly challenge systems to aggregate and reason across multiple documents.

Complex QA necessitates multi-step reasoning processes to aggregate and combine pieces of evidence from different sources systematically, as demonstrated by models evaluated on datasets like ComplexWebQuestions [66] and MultiRC [28]. This challenge is amplified by ambiguities, inconsistencies, and incomplete information prevalent in natural language texts [4], highlighting the need for robust reasoning capabilities.

Consider the seemingly straightforward question in Figure 1.1: *“Which city hosted the*

first Summer Olympics after World War II?" Answering it correctly involves several reasoning steps:

1. Identifying when World War II ended.
2. Determining which Olympics took place immediately after that date.
3. Extracting the host city for that edition.

No single document may contain all of these facts, and the required information may be scattered across different sources. The system must retrieve relevant evidence for each step and integrate it to produce the final answer. If retrieval misses one crucial document, such as the year of the first post-war Olympics, then the reasoning process can fail regardless of model capacity.

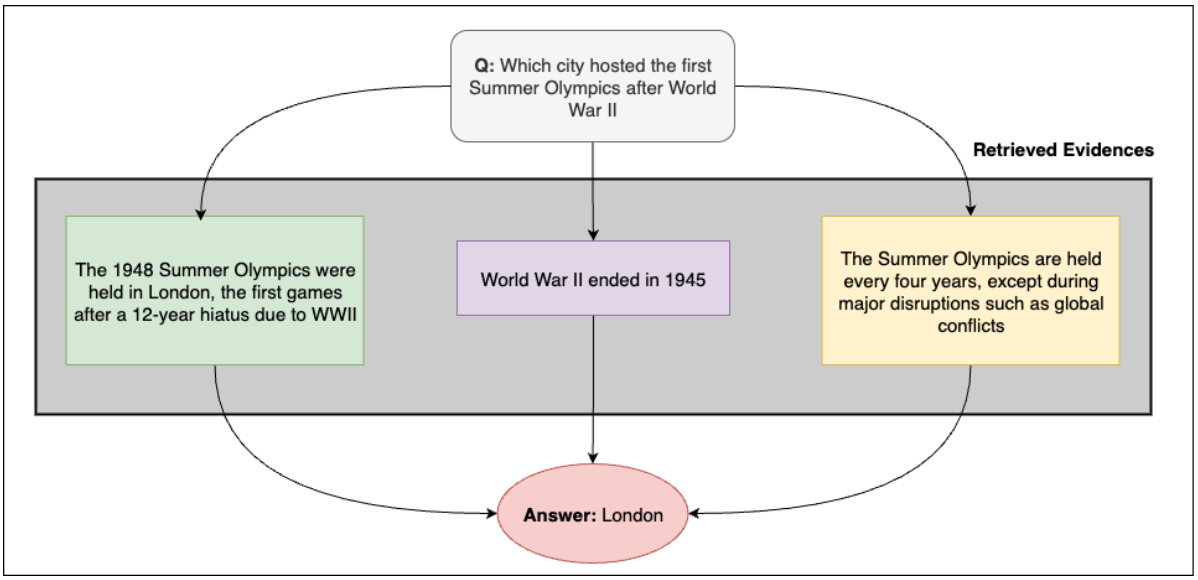

Figure 1.1: A complex ODQA example. Answering requires integrating facts from multiple retrieved documents. Missing even one key piece can lead to an incorrect answer.

This points to two intertwined challenges. First, complex ODQA faces a *bounded-recall* problem: retrieval typically admits only the top-k results, so useful evidence outside that window is invisible to the reader module [72]. Second, even with the right evidence present, LLMs can make reasoning errors—misinterpreting numbers, taking spurious paths ("overthinking"), or producing a fluent but incorrect top-1 output.

1.1.2. Retrieval Challenges: Bounded Recall and Evidence Quality

Modern ODQA systems typically follow a retrieve–then–read pipeline: a retriever surfaces candidate passages from a large corpus, and a reader (often an LLM) synthesizes an answer [36]. Retrieval has improved dramatically with dense encoders and

late interaction models, yet complex questions expose old weaknesses in new ways.

The bounded-recall problem:

In practice, readers only see the top-k retrieved passages. Everything below that cutoff is effectively invisible, regardless of how relevant it might be. This top-k gate induces a *bounded-recall* problem. If the right evidence is not present in the final ranked documents, the downstream reasoning step has to generate an answer from incomplete context. For multi-hop questions—where one missed passage can break the chain—this often determines success or failure. An illustrative example is shown in Fig 1.2.

Why Bounded Recall Affects Complex Questions:

- **Query mismatch.** A single surface query often underspecifies multi-faceted information needs, so essential passages never get scored highly enough.
- **Redundancy vs. diversity.** High-scoring passages can be semantically redundant (near-duplicates), crowding out complementary pieces needed to complete the reasoning chain.
- **Cascade effects.** Small retrieval misses lead to larger reasoning errors: with incomplete context, the reader hallucinates links, overfits to partial cues, or defers to priors [21].

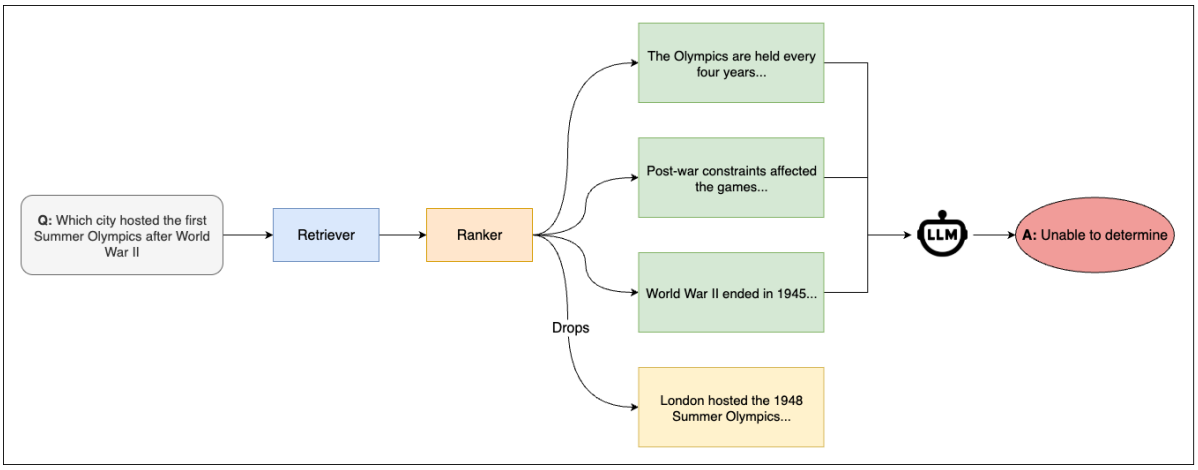

Figure 1.2: Bounded recall in ODQA: a top-k gate hides potentially critical evidence from the reader. Even strong rankers can miss complementary documents needed to complete multi-step reasoning.

1.1.3. Evidence quality under a fixed budget.

Even when relevant passages enter the top-k, their *quality* matters. Passages that are long, noisy, or only tangentially relevant waste the reader's limited attention window; overly similar passages reduce marginal utility; and numerically or temporally

ambiguous snippets increase the chance of incorrect inferences. Dense retrieval [26] and late-interaction architectures [29] improve ranking signals, but they do not directly reason about what *set* of documents best supports the whole answer.

Why iterative retrieval helps, but is not enough:

Iterative or multi-step retrieval attempts to refine queries as partial answers emerge, widening recall and improving coverage. This direction is promising as the system forms a hypothesis, it can seek missing pieces. Yet without an explicit signal of what is still needed, iterations can drift, add redundant context, or amplify early mistakes. In short, better retrieval reduces (but does not remove) the risk that the reader must reason from incomplete or noisy evidence.

1.1.4. Improving Reasoning Quality Beyond Retrieval

Even with high-quality evidence in hand, the reasoning stage can still fail. Large Language Models (LLMs) can produce coherent explanations, yet for complex questions the reasoning often contains gaps, unsupported steps, or subtle numerical mistakes. This makes the top-1 output fragile and unreliable for tasks that require multi-step logical processing.

Limitations of Relying on a Single Output

Relying on a single chain of thought (CoT) is risky for multi-hop or reasoning-intensive questions. Small variations in decoding or the model's focus can lead to different intermediate steps, sometimes altering the conclusion entirely. In some cases, the answer might be correct but the reasoning flawed, which can make the result hard to trust. In others, most steps might be valid but a final calculation error leads to the wrong answer. This brittleness means that one-off generations cannot reliably capture the full reasoning potential of an LLM [79, 78].

Expanding the Solution Space via Test-Time Scaling

The shortcoming of relying on the top-1 output has motivated a more robust approach: to expand the solution space at inference by sampling multiple reasoning paths for the same question [64]. This process, known as test-time scaling, trades additional inference compute for improved reliability (see Fig 1.3). The core idea is that generating many independent CoTs increases the likelihood that at least one of them will be both logically sound and factually correct. This shifts the challenge from "finding the perfect answer in one try" to "generating several plausible answers and choosing the best one." The latter is often easier because error elimination is generally more reliable than error-free generation [7].

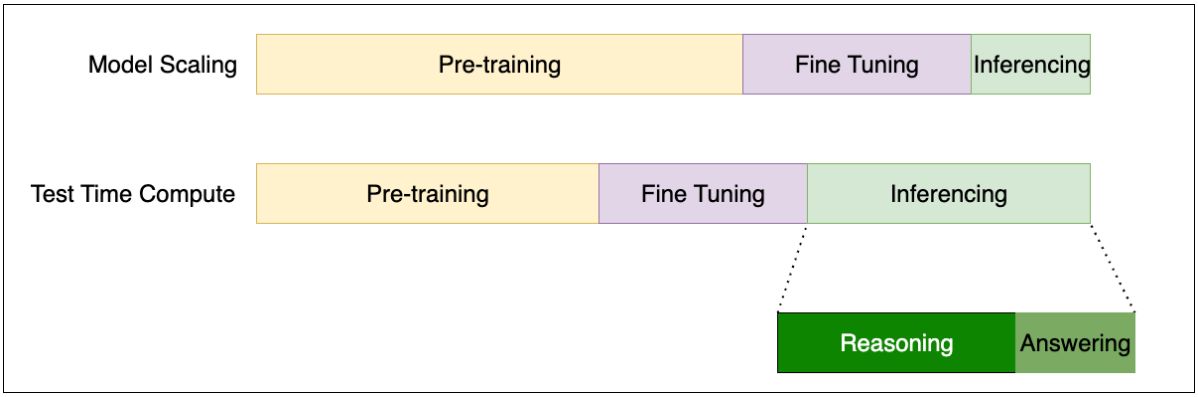

Figure 1.3: Test-time scaling shifts the focus from increasing model size to increasing inference-time compute, enabling expanded reasoning and better answer selection without modifying the model's parameters.

Verifier-Guided Selection of Reasoning Chains

Once multiple reasoning chains are available, a separate verifier model can be used to select the most credible one [71, 7, 40]. Unlike simple majority voting, which favors the most frequent answer, a verifier can assess each chain on factors such as logical coherence, factual grounding in the provided evidence, and internal consistency. By explicitly evaluating reasoning quality rather than frequency, verifier-guided selection reduces the risk of selecting a flawed but popular chain. When combined with improved retrieval, this approach strengthens both the evidence-gathering and reasoning stages, leading to more trustworthy final answers. This hypothesis is supported by our experiments. Generating 10 reasoning paths per claim frequently yields at least one path with correct reasoning, as reflected in the upper bound results in Table 5.5.

1.2. Research Questions

Building upon these insights, our approach aims to systematically investigate the following research questions (RQs):

- **RQ1:** How can answer-level feedback be incorporated into iterative retrieval to improve the robustness and recall of multi-hop question answering systems, compared to uncertainty-based feedback mechanisms?
- **RQ2:** How does test-time expansion of the solution space via multiple reasoning paths affect the quality and consistency of final answers in complex question answering pipelines?
- **RQ3:** How does the choice of verifier architecture impact the effectiveness of chain selection in a test-time reasoning framework?

By systematically exploring these research questions, this verification-centric methodology leverages iterative feedback loops and test-time scaling to significantly improve complex QA systems, driving greater reliability and robustness in automated answering systems.

1.3. Scientific Contributions

This thesis makes the following contributions to the field of complex ODQA:

1. **Dynamic answer-level feedback for iterative retrieval:** We propose a retrieval framework that incorporates answer-level consistency as feedback for document selection. Instead of relying solely on uncertainty-based heuristics, the method dynamically adjusts the retrieval frontier based on semantic agreement in generated answers, leading to higher recall and robustness in multi-hop scenarios.
2. **Integration of test-time scaling for reasoning enhancement:** We adapt the test-time scaling paradigm to ODQA by generating multiple diverse reasoning chains for each question and selecting the most credible one. This expands the solution space during inference without requiring retraining, improving the reliability of final answers in reasoning-intensive cases.
3. **Verifier-guided reasoning chain selection:** We evaluate the use of verifiers for selecting the most accurate reasoning chain from multiple candidates. Our study compares fine-tuned smaller models with larger off-the-shelf LLMs, showing that lightweight fine-tuned models can match or even outperform larger models in chain selection, offering a more computationally and financially efficient alternative.

1.4. Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews prior research relevant to the challenges of retrieval and reasoning in complex open-domain question answering. Chapter 3 presents the proposed approach in detail, including the iterative retrieval framework with answer-level feedback, the application of test-time scaling, and the use of verifiers for reasoning chain selection. Chapter 4 describes the experimental setup, datasets, and implementation details used to address the research questions. Chapter 5 reports the findings, analyzes their implications, and relates them to existing literature. Finally, Chapter 6 summarizes the main contributions, highlights limitations, and outlines potential directions for future work.

2

Related Work

2.1. Complex Question Answering

Complex question answering (QA) systems are designed to address queries that cannot be resolved by retrieving a single passage. Instead, these systems require multi-hop reasoning. This means chaining information across multiple documents or sentences to reach a correct answer. This represents a significant shift from traditional single-hop QA benchmarks like SQuAD [54], where a single context span suffices, to more demanding tasks like HotpotQA [89], WikiHop [80], and ComplexWebQuestions [66], where answers depend on integrating scattered information. HotpotQA, for instance, provides not only the answer but sentence-level supporting facts across two or more documents, thus enabling explainable multi-hop reasoning.

The difficulty in complex QA stems from several intertwined factors. First, relevant information is often *distributed across multiple sources*, requiring systems to locate and connect disparate pieces of evidence. Second, these datasets often include *distractor documents*. These are passages that are topically similar but irrelevant to the reasoning chain thus making the retrieval process more error-prone. Third, the reasoning process may require combining heterogeneous types of information, such as numerical values, temporal facts, and relational knowledge, which challenges even the most capable models. Lastly, multi-hop reasoning typically increases the risk of error compounding: a single missing or incorrect piece of evidence in the early stages can derail the entire reasoning chain.

Recent benchmarks have been specifically designed to evaluate these abilities. While HotpotQA remains one of the most foundational datasets—with 113K Wikipedia-based

questions requiring sentence-level supporting facts, it has been shown to suffer from shortcut opportunities that undermine its multi-hop reasoning goals [46, 70]. As a result, newer datasets were introduced to better evaluate complex reasoning capabilities and compositional generalization.

Among these, 2WikiMultiHopQA [16] was constructed by pairing Wikidata triples with Wikipedia passages to ensure genuine multi-hop reasoning across both structured and unstructured knowledge sources. It contains over 119K examples and enforces reasoning paths explicitly, thereby requiring systems to traverse factual chains involving multiple entities and relations. Similarly, MuSiQue [69] was designed to challenge models with compositional multi-hop questions derived from single-hop QA pairs. With both ”single-hop hard” and ”multi-hop” versions and built-in adversarial distractors, MuSiQue has emerged as one of the most rigorous benchmarks for evaluating cross-sentence and cross-paragraph reasoning. These datasets present higher reasoning difficulty and significantly reduce the effectiveness of shallow heuristics, forcing systems to perform genuine multi-step reasoning.

Beyond standard Wikipedia-based QA, researchers have begun incorporating cross-format reasoning into benchmarks such as narratives, emails, and semi-structured data. The MuSiQue-2Wiki combined setup [70] enables evaluation of models' generalization across formats by mixing questions from both datasets, which results in a performance variance of up to $\sim 15\%$ depending on domain shifts.

Overall, complex QA is challenging not merely because it requires identifying the correct answer, but because it demands *integrating multiple, dispersed, and potentially heterogeneous pieces of evidence into a coherent reasoning process*. Addressing these challenges requires not only sophisticated reasoning models but also retrieval pipelines that can reliably surface all necessary evidence which we explore in detail in the next section.

2.2. Improving Retrieval for Complex Question Answering

In complex question answering (CQA), the primary bottleneck often lies not in the reasoning capabilities of large language models (LLMs), but in the initial retrieval phase, where essential evidence may be missed early on. This bounded-recall issue significantly hampers downstream reasoning, especially when the final answer depends on assembling multiple, independently retrieved facts. Most question answering (QA) systems today rely on a retrieve-then-read pipeline, where a retriever fetches relevant

documents and a reader or language model extracts or generates an answer based on them.

From a retrieval perspective, early methods can be broadly divided into sparse and dense approaches. Sparse techniques such as TF-IDF [65], BM25 [59, 58], and SPLADE [11] rely on lexical overlap between the query and candidate documents, making them effective for single-hop queries where relevant terms appear explicitly. However, these methods often fail when questions require semantic understanding beyond exact keyword matching. Dense retrievers, such as the Dense Passage Retriever (DPR) [26] and ANCE [86], address this limitation by encoding queries and passages into a shared vector space using neural encoders, typically BERT-based, and ranking them via dot-product or cosine similarity. By capturing semantic similarity, dense methods substantially improve recall. DPR, for example, often yields a 9-19 point gain over BM25 in open-domain settings. Nonetheless, early dense retrievers were primarily designed for single-hop retrieval, limiting their effectiveness in truly multi-hop question answering scenarios.

To overcome these limitations, multi-hop dense retrieval models were developed. Xiong et al. [87] introduced a two-stage framework that retrieves passages with DPR in sequential hops conditioned on previously fetched evidence. Their approach achieved strong performance on HotpotQA and the multi-evidence FEVER dataset [67], matching earlier graph-based systems in accuracy while drastically reducing inference cost. They demonstrated that multi-hop dense retrieval can avoid reliance on corpus-specific signals such as hyperlinks or entity annotations, eventually making it more generalizable across corpora.

In parallel, graph-based reasoning methods emerged to explicitly model relationships between entities and passages. Models like QA-GNN [90] and Hierarchical Graph Networks (HGN) [10] build entity or passage-level graphs, over which Graph Neural Networks (GNNs) [61] propagate information. Attention mechanisms like GAT [74] help these models weigh the importance of different neighbors. These architectures select evidence paths and jointly predict answers and supporting facts. This helps in improving interpretability and robustness. Other graph-augmented innovations like PATHFID [91] and SeqGraph [55] convert reasoning chains into sequence prediction or path-following problems over graph-structured evidence, attaining strong results on HotpotQA and IIRC.

However, despite advances in retriever quality, even top dense retrievers still struggle to capture all relevant evidence needed for complex, multi-hop questions. This issue is especially critical when a single missing document can break the entire rea-

soning chain. Researchers have explored Pseudo-Relevance Feedback (PRF) [37] to address this. In PRF, the system uses documents from an initial retrieval round to reformulate the query and retrieve again, thereby bridging vocabulary or representation mismatches. While this helps recall, PRF approaches often suffer from semantic drift, where the reformulated query starts retrieving irrelevant or misleading content. This makes them risky for QA tasks, where precision is just as important as recall.

To tackle this, more adaptive retrieval strategies have been introduced. These methods typically involve a two-stage setup: an initial retrieval followed by a re-ranking step guided by signals from an external model or feedback loop. MacAvaney et al. [44], for instance, proposed using retrieval graphs to support iterative re-ranking based on document relationships. This approach expands the candidate pool beyond the initial retrieval results by leveraging a corpus graph that encodes document relationships. Their approach builds on the clustering hypothesis, which suggests that related documents are likely relevant to the same query. Instead of relying solely on the top retrieved documents, Gar iteratively identifies and re-ranks the neighbors of high-scoring documents based on their pre-computed similarity in the corpus graph. This feedback loop allows the system to surface relevant documents that might have been overlooked in the initial retrieval stage, leading to improvements in both precision and recall. Their experiments on the MS MARCO [3] dataset demonstrated that Gar significantly enhances re-ranking effectiveness, particularly when combined with neural ranking models such as monoT5 [49].

A similar strategy was proposed by Kulkarni et al. in their Lexically-Accelerated Dense Retrieval (LADR) [31] framework which is a lightweight re-ranking system that reuses ranking signals to limit cost. Unlike Gar, which focuses on leveraging document-to-document relationships, LADR integrates lexical and dense retrieval signals to enhance re-ranking efficiency. The approach begins with a lexical retrieval step (e.g., BM25) to seed an initial document pool, followed by a dense retrieval exploration where neighboring documents—determined through vector similarity—are included for re-ranking. This hybrid approach bridges the gap between lexical and dense retrieval, allowing the system to capture relevant documents that might not share exact lexical matches with the query. The authors introduced two variants: Proactive LADR, where all neighboring documents are retrieved at once and re-ranked in a single step, and Adaptive LADR, which iteratively refines the ranking by incorporating new documents in multiple stages. Their results showed that LADR achieves performance on par with exhaustive dense retrieval while maintaining low latency (8ms per query), making it suitable for real-world applications.

Building upon these prior approaches, Rathee et al. introduced Quam (Query Affinity Modelling) [56], an adaptive re-ranking method that enhances retrieval by incorporating a learned document affinity model. Unlike traditional methods that rely on fixed similarity metrics, Quam leverages historical query-document relevance data to predict how likely two documents are to be co-relevant to the same query. By using this learned affinity, the system constructs a more refined corpus graph, allowing for a more targeted selection of candidate documents during the re-ranking process. Experimental results indicate that Quam achieves up to 26% improvement in recall, particularly in settings where the re-ranking budget is constrained. Furthermore, when Quam was integrated into Gar's pipeline, it led to additional gains in retrieval effectiveness, demonstrating how learned document representations can significantly enhance adaptive retrieval performance.

Retrieval-augmented generation (RAG) [36, 12, 14] represents another line of work, merging a parametric transformer (often BART [35]) with a dense retrieval mechanism, and jointly optimizing retriever and generator end-to-end. While initially designed for single-hop tasks, RAG variants such as RAG-Token and RAG-Sequence extend the approach to multi-hop QA. More recent generative retrieval approaches move beyond static retrieval pipelines toward autoregressive, LM-driven retrieval. Generative Multi-hop Retrieval [34], for example, alternates between generating intermediate queries and retrieving corresponding passages, dynamically deciding when to stop. It is a prominent example where an encoder-decoder model generates the next query string (or document) based on prior context by avoiding rigid vector-space reformulation and yielding improvements across five QA datasets. Moving further, Self-Critique Guided Iterative Reasoning (SiGIR) [6] models adopt chain-of-thought-like branching combined with self-evaluation scores to steer retrieval. While SiGIR is still emerging, the broader paradigm of LM-based planner + retriever has been shown to improve over rigid pipelines by 5-10 EM points on HotpotQA and IIRC, though SiGIR as described remains future research needing formal publication.

An orthogonal effort integrates graph-based structures into retrieval, such as relation-aware RAG models that fetch passages based on knowledge-graph cues and adaptively expand retrieval based on token-level uncertainty (e.g., "stop early" decisions) [23, 88]. These methods have shown consistent gains on HotpotQA's explainability scores (2-4 points higher EM) and improved recall on IIRC [39] by $\sim 7\%$.

Yet, most of these approaches still operate with a fixed retrieval budget and only use relevance signals tied to the query, not the answers or the semantic content of the documents themselves. This is where SUNAR (Semantic Uncertainty based Neigh-

bourhood Aware Retrieval) [72] brings a different perspective. It targets the weakness of standard retrievers by introducing an iterative, LLM-guided document exploration method based on semantic uncertainty estimation and graph-based document neighbourhoods.

SUNAR basically builds upon the clustering hypothesis. Rather than solely relying on query-document similarity, it creates a neighbourhood graph (typically built with ColBERTv2 embeddings) and explores this graph during retrieval. The innovation lies in incorporating Answer Semantic Uncertainty (ASU). It is a mechanism where multiple answer candidates are generated for a sub-question, clustered semantically (using bidirectional entailment), and the number of unique semantic sets is used as a proxy for answer consistency. Higher uncertainty signals greater ambiguity, prompting the model to penalize the associated evidence documents. This strategy contrasts with Pseudo-Relevance Feedback (PRF) methods, which are prone to semantic drift by reinforcing initially retrieved (but potentially irrelevant) documents.

SUNAR is especially effective in multi-hop QA setups like 2WikiMultiHopQA [16] and MuSiQue [69], where missing or contradictory evidence is common. By using the model's own generations as a feedback signal, SUNAR avoids some of the key pitfalls of PRF such as drifting too far from the original intent and thus it significantly improves recall without sacrificing quality. In fact, SUNAR was shown to outperform strong baselines like SELF-ASK and ReAct by over 30% EM on 2WikiMultiHopQA. SUNAR's approach, using LLM-internal signals as retrieval feedback, takes a more semantic and context-aware path. This aligns with recent findings in IR studies [43, 57], which indicate that current retrievers fail to adapt when question phrasing drifts semantically or when the evidence is lexically distant but semantically critical. SUNAR is part of a growing trend of retrieval-enhancing strategies that treat the retrieval stage not as a static, one-shot process, but as an iterative, feedback-driven loop. Related efforts such as Adaptive Re-ranking with Corpus Graphs [44], Lexically-Accelerated Dense Retrieval [31], and self-reflective retrieval agents like Self-RAG [2] share this goal. However, SUNAR's use of semantic answer variability as a feedback mechanism and its principled integration of graph-based retrieval set it apart. As LLMs grow more capable, methods like SUNAR which exploit their internal reasoning for better retrieval, offer a promising direction for robust, interpretable, and high-recall complex QA.

2.3. Test-Time Scaling and Verifier-Guided Reasoning

Test-time scaling refers to using extra compute at inference time to improve reasoning outcomes in large language models (LLMs). Here, "extra compute" means increas-

ing the number of reasoning paths explored for a given question, either by generating them independently or by adaptively refining them, and then selecting the best outcome. Broadly, this can be done in two ways: **parallel scaling**, where multiple outputs are generated in one batch and ranked or voted on [79, 7], and **sequential scaling**, where outputs are generated in stages, with each step informed by feedback from earlier generations [64, 25]. Parallel scaling is faster but may waste compute on low-quality candidates, while sequential scaling can allocate compute more efficiently but at the cost of latency. The simplest form of test-time scaling involves generating multiple answers and using majority voting (also popularly known as self-consistency). The main aim of majority voting is to pick the most common answer. Wang et al. [79] introduced Self-Consistency, showing that sampling several chain-of-thought (CoT) outputs and voting improves multi-step reasoning accuracy significantly. However, this method comes with diminishing returns. The improvements plateau as more samples are generated, and cost scales linearly with sample size. Another issue that Majority Voting can face is that the method can consistently pick the wrong answer just because it's the most occuring one.

Recent advances in test-time scaling methods generally fall into two main categories. The first line of work focuses on modifying the proposal distribution, i.e., how the model generates its outputs [8, 33, 92, 63, 45, 52, 25]. These methods aim to guide the sampling process to be more diverse, consistent, or likely to yield correct answers. For example, through techniques like temperature tuning, beam search, or decoding strategies that prioritize consistency. The second approach centers around training reward models or verifiers that evaluate and select the best answer from a set of generated candidates [78, 64, 7, 40]. Instead of changing how outputs are sampled, these methods improve what is ultimately chosen by learning to recognize better answers, either based on final outcomes or the reasoning steps that lead to them. These fall into two types:

- **Outcome-based Reward Models (ORMs)** score final answers directly, often trained on human preferences or correctness labels [7].
- **Process-based Reward Models (PRMs)** evaluate each reasoning step in a chain-of-thought, offering finer-grained feedback [40, 71]. PRMs have shown better performance in math reasoning benchmarks like MATH, GSM8K, and ProcessBench, as they identify logical errors early rather than just judging final correctness [62, 40]. For instance, ThinkPRM [27] uses only 1% of data yet matches or surpasses other verifiers on core reasoning tasks.

Training verifiers typically requires annotated chains-of-thought. PRMs need each

step labeled as correct or incorrect, and ORMs need final responses scored or ranked. Researchers have used crowd-sourced labeling (e.g., the PRM800K dataset [40]) or synthetic data from expert LLMs to bootstrap training [27]. Once trained, verifiers are used at test time in a *search + verify* pipeline. The base model samples multiple CoT candidates. Each is scored by the verifier, either as a whole (ORM) or step-wise (PRM). Out of these two, the top-scoring answer is selected. Such pipelines (e.g., ModelSwitch [5]) not only improve accuracy but also allocate compute more wisely, sometimes outperforming much larger models with the same budget.

Recent work has formalized test-time scaling under a theoretical lens. They show that a compute-optimal mixture of Best-of-N and verifier-assisted strategies can yield much better performance than merely increasing model size, especially under a fixed compute budget [64, 83, 95, 18, 96]. These mechanisms enable smaller or mid-sized LLMs to match or exceed larger models like GPT-4 in complex reasoning tasks while managing inference cost. The future likely lies in better verifiers (e.g., generative PRMs like GenPRM [98]) and adaptive systems that balance sampling and verification per query.

3

Methodology

3.1. System Overview

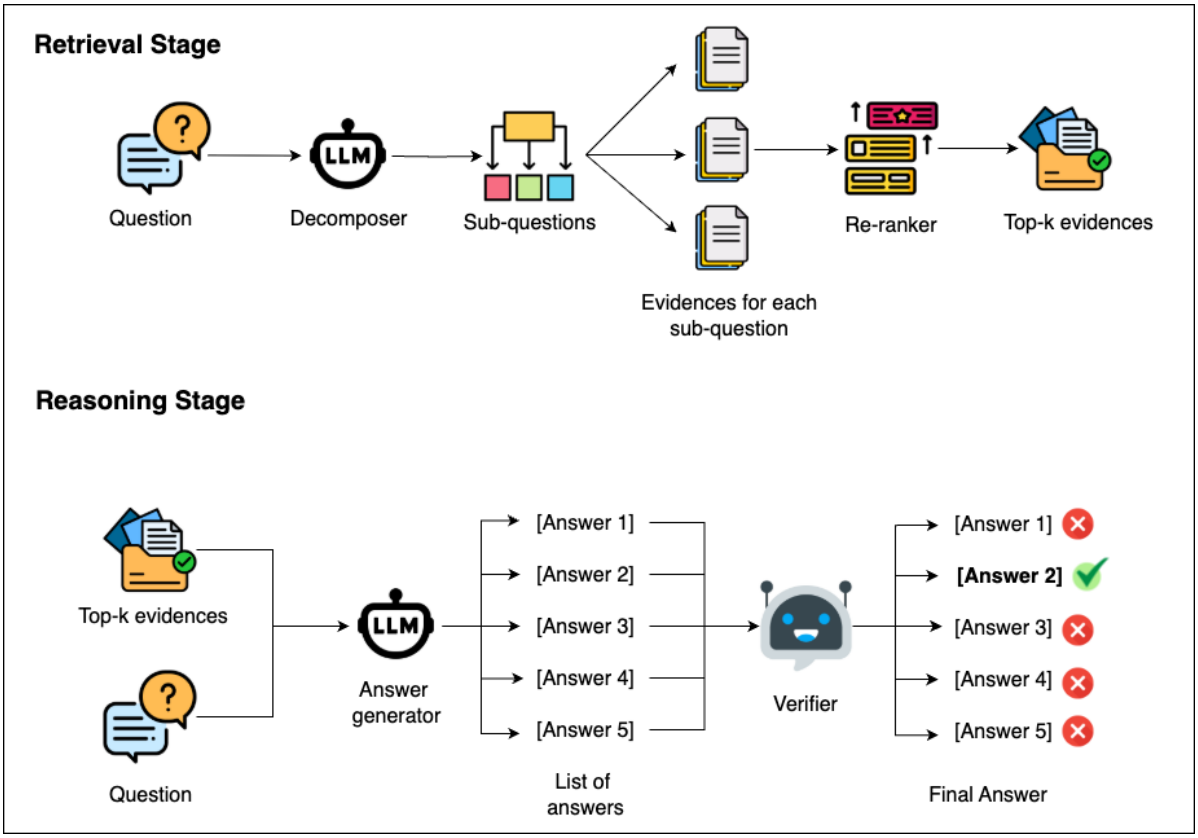

Figure 3.1: Overview of the two-stage pipeline consisting of retrieval and reasoning components.

To tackle complex open-domain questions, our system adopts a two-stage pipeline consisting of a retrieval stage followed by a reasoning stage. As illustrated in Figure 3.1, the retrieval stage begins with decomposing the input question into simpler sub-questions. For each sub-question, relevant documents are retrieved and subsequently re-ranked using a cross-encoder to obtain the top-k evidences. These curated evidences form the basis for the reasoning stage, wherein the original question is paired with the retrieved context to generate multiple candidate answers. A verifier model is then employed to score these candidates, ultimately selecting the most reliable answer. This modular design enables the system to both navigate multi-hop reasoning and verify answer consistency using retrieved evidence.

3.2. Neighbourhood Aware Retrieval (NAR)

Most existing retrieval pipelines in open-domain QA rely on a static set of top-k documents retrieved in the first stage. However, this can limit performance when the initial retrieval fails to surface useful or diverse evidence. To address this, we adopt the Neighbourhood Aware Retrieval (NAR) framework [44], which iteratively re-ranks and expands the document pool by leveraging a neighbourhood graph over the corpus. The key idea is that documents close in the semantic space are more likely to provide complementary evidence for answering a complex question [22]. As shown in Figure 3.2, the NAR pipeline alternates between scoring document batches (using re-ranking), and expanding the candidate pool via nearest neighbors from a precomputed graph. This allows the retrieval process to progressively move toward better evidence.

In our implementation, we apply NAR at the sub-question level, where the LLM first decomposes the original question and generates follow-up sub-questions. Each sub-question is processed independently through the NAR loop to gather high-quality supporting documents. These sub-questions and their corresponding batches of retrieved documents form the basis for downstream answer generation and consistency checking.

Building upon this framework, we introduce Frontier Aware Iterative Retrieval (FAIR) — a new re-ranking strategy designed to reward answer consistency and confidence across iterations.

3.3. Frontier Aware Iterative Retrieval (FAIR)

While NAR helps overcome the limitations of first-stage retrieval by incorporating local document neighbourhoods for expansion and re-ranking, it does not account for the

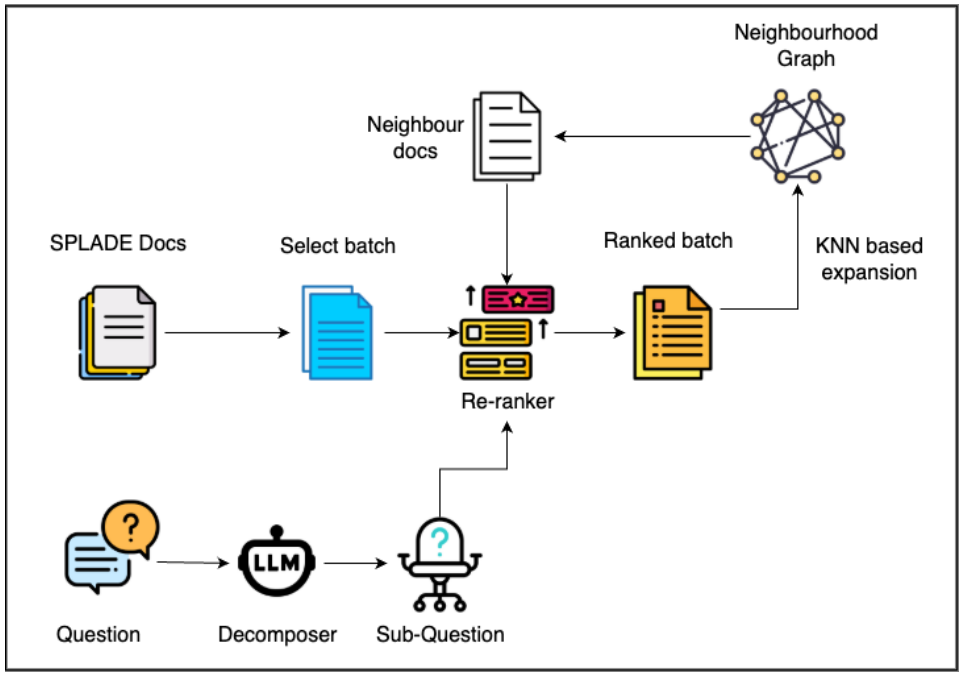

Figure 3.2: Neighbourhood Aware Retrieval (NAR) pipeline.

consistency of answers across different batches of documents in the FAIR setup. So, NAR lacks a mechanism to estimate and reinforce confidence in answers that repeatedly emerge from different evidence contexts. As a result, even if multiple retrieved batches independently support the same answer, NAR does not explicitly leverage this agreement to guide subsequent retrieval steps.

To improve the quality of evidence used for complex question answering, we introduce FAIR (Frontier Aware Iterative Retrieval). It is a re-ranking strategy that collects confidence for answers in the frontier across varying batches of documents and uses this information to rescore the candidate evidence. FAIR retains and promotes documents that reinforce the current answer frontier, thereby increasing the likelihood that later iterations retrieve complementary and corroborative evidence. FAIR is built on the same underlying infrastructure as SUNAR but diverges in its scoring logic. While SUNAR penalizes documents that contribute to semantic uncertainty, FAIR avoids penalization altogether and instead selectively rewards evidence that drives convergence towards a consistent answer frontier. So basically, it selectively boosts the scores of documents that strengthen the current answer frontier. This type of distinction is especially important when working with LLMs, which are often stochastic and prone to minor lexical variations in output. Penalizing documents because of slight inconsistencies across generations may result in unfair suppression of genuinely helpful content. Thus, FAIR's design makes it more robust to LLM generation variance and avoids unfairly downgrading helpful documents that may have received inconsistent

answers due to model randomness.

FAIR implements the NAR pipeline, as shown in Figure 3.2. Given a complex question, it decomposes the problem into simpler sub-questions and retrieves a batch of documents using a sparse retriever such as SPLADE. These are passed to a re-ranker that selects the top candidates. The corresponding neighbourhood in the document graph is then explored via a KNN expansion module, allowing the retrieval of semantically related documents. As this cycle repeats, FAIR updates document scores based on their contribution to consistent LLM-generated answers. Figure 3.3 represents the FAIR pipeline.

It is a novel method that integrates document re-ranking with a dynamic answer frontier, which is maintained and updated over time. This frontier consists of one or more highly supported answers and serves as a running estimate of the system's current belief about the correct response to a sub-question. At each iteration, FAIR rewards those documents whose answers align with the current frontier and ignores others, allowing it to build answer confidence through iterative consistency checks.

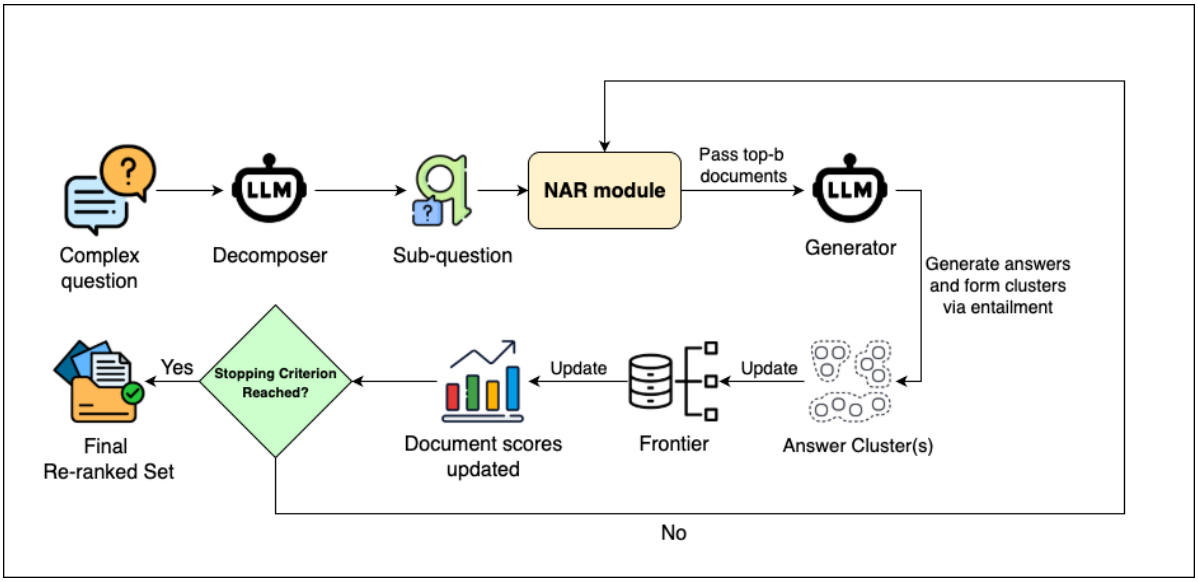

Figure 3.3: Frontier Aware Retrieval (FAIR) pipeline.

3.3.1. Working of FAIR

Algorithm 1 FAIR: Frontier Aware Iterative Retrieval

Input: Question Q, sub-question sq, initial document pool D, batch size b, document graph G, cross-encoder ψ_{CE}, language model ϕ_{LLM}
Output: Final re-ranked document set R^+

$R^+ \leftarrow \emptyset$ ▷ Final re-ranked result set
$C \leftarrow D$ ▷ Candidate pool
$N \leftarrow \emptyset$ ▷ Neighbor pool
$\mathcal{F} \leftarrow \emptyset$ ▷ Initial answer frontier
while $|R^+| < k$ **do**
  $B \leftarrow$ top-b documents from C
  $\{a_1, \ldots, a_k\} \leftarrow \phi_{LLM}(sq, B)$ ▷ Generate k answers using LLM
  $\mathcal{C} \leftarrow \sigma(\{a_1, \ldots, a_k\})$ ▷ Cluster answers via entailment
  TopK $\leftarrow$ top-k clusters in $\mathcal{C}$ by frequency
  Update $\mathcal{F}$ using TopK ▷ Replace weakest entry if needed
  if $|\mathcal{F}| = 1$ **then**
    $multiplier \leftarrow 2$
  else
    $multiplier \leftarrow 1$
  end if
  for all $d \in B$ **do**
    $score \leftarrow \psi_{CE}(sq, d) \times multiplier$
    Store $score$
  end for
  $R^+ \leftarrow R^+ \cup B$
  $C \leftarrow C \setminus B$
  $N \leftarrow N \cup (\text{Neighbours}(B, G) \setminus R^+)$
  $C \leftarrow \begin{cases} D & \text{if } C = N \\ N & \text{if } C = D \end{cases}$
end while
return R^+

Notation: We denote the language model generation function as ϕ_{LLM}, the answer clustering operation as σ, and document scoring using a cross-encoder as ψ_{CE}. The answer frontier $\mathcal{F}$ is updated iteratively based on the most frequent semantic clusters to promote consistency in the re-ranked result set R^+.

As shown in Algorithm 1, the retrieval takes place in the following manner. We begin with an initial sub-question sq derived from a complex query Q, and an initial document pool D retrieved via a sparse retriever. The re-ranking loop operates over batches of size b, using a cross-encoder model ψ_{CE} for scoring and a language model ϕ_{LLM} for generating candidate answers. The algorithm maintains three working sets: a can-

didate pool C, a neighbor pool N, and a final re-ranked result set R^+, all initialized accordingly. Additionally, a dynamic set $\mathcal{F}$ is maintained to represent the current *answer frontier*, which reflects the most consistent answers observed so far.

At each iteration, the algorithm selects the top-b documents $B \subseteq C$ and passes them along with sq to the LLM. This results in a set of k generated answers:

$$\{a_1, a_2, \ldots, a_k\} \leftarrow \phi_{LLM}(sq, B) \tag{3.1}$$

These answers are then semantically clustered into groups of equivalent meaning. Clustering is carried out using bidirectional entailment checks to group paraphrases into consistent semantic clusters. Formally, we denote this operation as:

$$\mathcal{C} \leftarrow \sigma(\{a_1, a_2, \ldots, a_k\}) \tag{3.2}$$

where σ is the clustering function that merges semantically equivalent answers.

From the resulting clusters $\mathcal{C}$, the $top-k$ most frequent clusters are selected and used to update the answer frontier $\mathcal{F}$. If a newly observed answer cluster is not already in the frontier and is supported more strongly than the weakest frontier entry, it replaces the weaker entry. The update logic ensures that $\mathcal{F}$ always contains the up to k most dominant answers observed across document batches, effectively forming a bounded confidence set. This evolving frontier allows the system to track convergence of answers over time.

The consistency of the answer frontier is then used to modulate document scores. If the frontier contains only a single answer (i.e., $|\mathcal{F}| = 1$), this indicates high agreement among model generations. In such cases, documents in the current batch are rewarded by doubling their score:

$$multiplier = \begin{cases} 2 & \text{if } |\mathcal{F}| = 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.3}$$

Each document $d \in B$ is then scored by the cross-encoder, conditioned on sq, and adjusted by the multiplier:

$$score_d = \psi_{CE}(sq, d) \times multiplier \tag{3.4}$$

This reward mechanism encourages the inclusion of documents that promote answer consistency without explicitly penalizing documents that do not, avoiding potential

over-pruning of relevant but underrepresented evidence.

After scoring, the current batch B is added to the final re-ranked set R^+, and removed from the candidate pool C. The graph G, constructed offline using dense retrieval similarity (e.g., via ColBERT or SPLADE embeddings), is then used to expand the document pool. Specifically, the neighbors of documents in B that have not yet been added to R^+ are inserted into the neighbor pool N:

$$N \leftarrow N \cup \big(\text{Neighbours}(B, G) \setminus R^+\big) \tag{3.5}$$

To ensure broader exploration of the document graph and mitigate potential stagnation, the candidate pool C is alternated between the original document pool D and the neighbor pool N. When the current candidate pool is exhausted, the update is performed as follows:

$$C \leftarrow \begin{cases} D & \text{if } C = N \\ N & \text{if } C = D \end{cases} \tag{3.6}$$

This alternation mirrors the expansion mechanism used in SUNAR [72], enabling the retrieval system to incrementally discover new but semantically relevant documents not surfaced by the initial sparse retriever.

The process continues until the re-ranked set R^+ contains *k* documents. These documents, now scored with consistency-aware feedback, are passed to the reasoning module of the QA system for final answer synthesis.

FAIR thus introduces a principled method for promoting documents that reinforce consistent model behavior, without prematurely excluding evidence that may become relevant in later stages. By dynamically maintaining a small frontier of supported answers and adjusting scores based on global consistency rather than local uncertainty, FAIR offers a robust and interpretable alternative to purely uncertainty-based document reranking.

3.3.2. Answer Generation and Final Reasoning

Once the final re-ranked document set R^+ has been constructed using the FAIR procedure, the system proceeds to the answer generation stage. For each sub-question sq_i derived during the decomposition phase, we select the top-l documents (where $l = 10$) from R^+ and prompt the language model to generate an answer a_i. This is repeated for all sub-questions in the decomposition chain.

While this sequential reasoning approach enables step-by-step synthesis of interme-

diate answers, we observed that it may suffer from cascading errors. Since the final answer is typically generated using only the result of the last sub-question and its corresponding evidence, any mistakes in earlier steps can propagate and lead to incorrect conclusions.

To mitigate this issue, we incorporate a post-hoc correction mechanism inspired by the Meta Evidence Reasoner (MER) proposed in SUNAR [72]. The idea behind MER is to revisit the entire reasoning trajectory and combine it with top-ranked evidence from across all sub-questions to re-evaluate the original query. Specifically, once we have obtained the full sequence of intermediate reasoning pairs:

$$(sq_1, a_1),\ (sq_2, a_2),\ \ldots,\ (sq_n, a_n)$$

we aggregate the top-l evidences retrieved during each step and concatenate them. This combined evidence set, along with the original complex question Q and the reasoning path, is passed once more to the LLM. The final answer is generated in a single pass, allowing the model to revise earlier steps and resolve contradictions, if any.

This simple correction mechanism significantly improves answer quality in practice, especially on complex multi-hop questions where factual coherence across reasoning steps is essential. As also noted in SUNAR, MER serves as an effective fallback strategy that allows the model to reconsider all available evidence before committing to a final output. In our pipeline, we find that combining FAIR with MER provides a robust and interpretable method for final answer generation that is both consistent and context-aware. The prompt used is shown in Figure A.1 in the Appendix.

3.4. Answer Selection via Test-Time Scaling and Verifier Models

Once the top-k evidences are retrieved for a given question, the reasoning stage is tasked with synthesizing them into a correct and well-justified answer. However, even with strong evidence, the first reasoning attempt from an LLM is not guaranteed to be correct. Factors such as stochastic decoding, incomplete utilization of provided evidence, and subtle reasoning errors can cause a top-1 output to fail. For instance, self-evaluation guided beam search reveals how uncertainty propagates across intermediate steps and undermines chain consistency [84]. Other studies show that even when a model correctly answers a question, its accompanying chain-of-thought may be logically incoherent or partially unsupported [47]. Furthermore, errors introduced late in a reasoning chain can disproportionately affect the final answer—a vulnerabil-

ity termed “late-stage fragility” [93]. Together, these findings indicate that relying on a single chain is brittle, especially for multi-hop or reasoning-intensive tasks.

To mitigate this risk, we adopt *test-time scaling* (TTS). It is a strategy that shifts the problem from “finding the perfect answer in one shot” to “generating multiple plausible answers and selecting the best one.” Instead of relying on a single chain of thought, we sample multiple reasoning paths for the same question, thereby expanding the space of candidate answers. This increases the likelihood that at least one candidate is both logically valid and factually correct [79, 64, 7]. Test-time scaling operationalizes this idea by trading additional compute for greater answer reliability. The selection step then becomes a matter of identifying the most trustworthy candidate, which can be done via majority voting or, as we explore later, with a trained verifier.

3.4.1. Majority Voting

In majority voting, after performing test-time scaling, we aggregate its outputs. More concretely, for each question q, the LLM is prompted k times (typically $k = 10$) using the same set of retrieved documents. This produces a set of candidate answers:

$$A_q = \{a_1, a_2, \ldots, a_k\} \tag{3.7}$$

We then perform majority voting over these k outputs to determine the final answer. This involves grouping semantically equivalent answers and identifying the most frequently occurring one. Formally, we define a function σ that clusters equivalent answers into sets:

$$\mathcal{C} = \sigma(A_q) = \{C_1, C_2, \ldots, C_m\} \tag{3.8}$$

where each cluster $C_i \subseteq A_q$ contains answers that are semantically similar (typically judged by exact match or normalized string equivalence).

The final answer a_q^* is then chosen as the representative of the largest cluster:

$$a_q^* = \arg\max_{a \in A_q} |\{a_i \in A_q \mid a_i \sim a\}| \tag{3.9}$$

where $\sim$ denotes semantic equivalence (e.g., ignoring casing, punctuation, or minor rewordings).

This procedure is illustrated in Figure 3.4, where multiple LLM outputs are grouped and the most common answer is selected. While simple, this approach leverages the natural variability of LLM generations to improve stability at inference time.

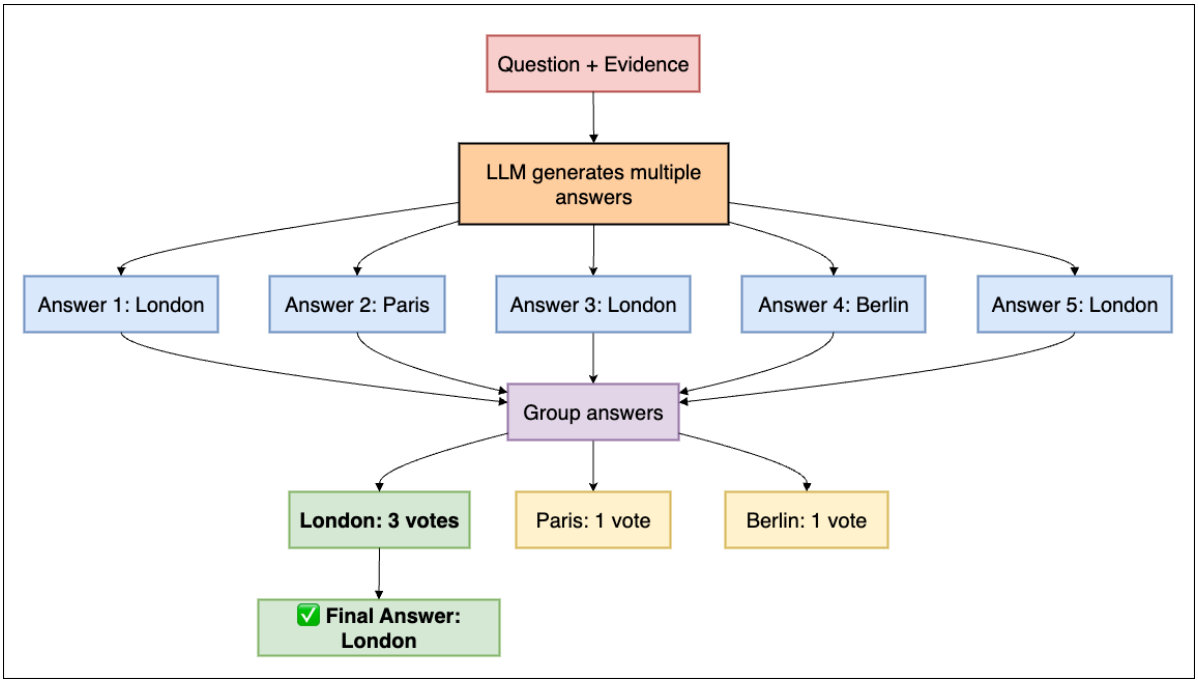

Figure 3.4: Illustration of the majority voting approach. Given a question and supporting evidence, the LLM generates multiple candidate answers. These are grouped by textual similarity, and the most frequently occurring answer is selected as the final prediction.

3.4.2. Training Verifier Models for Answer Selection

Having discussed majority-vote and self-consistency based mechanisms for combining multiple reasoning chains, we now turn to a more sophisticated and reliable method: verifier-based test-time scaling.

In the classic self-consistency (majority-vote) approach [79], multiple reasoning paths are sampled and their final answers are aggregated by frequency. Although widely adopted, this strategy has notable limitations:

- It assumes that the most commonly generated answer is always correct, which can fail when multiple chains converge on the same incorrect reasoning.
- It ignores the quality of individual chains and often overlooks minority but correct chains.
- As highlighted in Mirror-Consistency [19], majority voting may miss subtle uncertainties that alternative chains reveal.

Researchers have thus proposed augmenting aggregation with verification-based methods that evaluate the correctness of chains rather than just trusting their consensus. For instance, Li et al., propose DiVeRSe, a verifier-driven weighted voting approach that significantly improves accuracy across multiple reasoning benchmarks by dis-

carding faulty chains before aggregation [38]. Similarly Vacareanu et al., introduced a general-purpose verifier that weights chains based on confidence scores, outperforming majority-vote self-consistency on tasks like GSM8K and CommonsenseQA [73]. In V-STaR, the research demonstrates that verifiers substantially outperform both standard self-consistency and iterative self-improvement strategies in math reasoning and code generation settings [17].

Given these findings, we adopt verifier-based TTS in our retrieval-augmented pipeline. Rather than relying on majority voting alone, we fine-tune verifiers to assign a confidence score to each reasoning chain with respect to the question. At inference time, the answer with the highest verifier score is selected. This approach is more robust to noisy or spurious consensus and promotes reasoning chains that are both internally consistent and grounded in evidence.

The remainder of this section details how training data for verifiers is constructed and how fine-tuning is conducted to enable high-quality answer selection.

Training Data Preparation. We begin by sampling a subset of questions from the training set, denoted by $\{q_1, q_2, \ldots, q_n\}$. For each question q_i, we use its corresponding gold evidences to prompt a language model multiple times, generating a set of reasoning chains $\{c_{i1}, c_{i2}, \ldots, c_{ik}\}$. Each chain c_{ij} concludes with a predicted answer $\hat{a}_{ij}$.

Let a_i be the gold answer to question q_i. We define a normalization function $\eta(\cdot)$ to account for minor lexical variations. A binary label y_{ij} is assigned to each chain c_{ij} as follows:

$$y_{ij} = \begin{cases} 1 & \text{if } \eta(\hat{a}_{ij}) = \eta(a_i) \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

This results in a labeled dataset $\mathcal{D} = \{(q_i, c_{ij}, y_{ij})\}$, which contains both correct and incorrect reasoning chains for each question. Each training example consists of the input string formed by concatenating the question and the reasoning chain:

$$\texttt{[Question]: } q_i \quad \texttt{[Chain]: } c_{ij}$$

The corresponding label y_{ij} supervises the model's prediction for this input.

Importantly, evidences are not included in the input to the verifier during training. This design choice is made to avoid excessive context length and reduce the risk of introducing noise, as the reasoning chain itself typically contains the core logical path

leading to the predicted answer.

Model Training. Each input (q_i, c_{ij}) is tokenized and passed through a pretrained encoder. We experiment with two architectures: a RoBERTa-based model and a LLaMA-based model.

Rationale for model choice: RoBERTa [42] is a strong bidirectional encoder that has consistently shown competitive performance on natural language understanding tasks, making it well-suited for discriminative scoring of reasoning chains. In contrast, LLaMA [68] is a generative, decoder-only architecture capable of capturing broader contextual and reasoning patterns learned from large-scale pretraining. By including both architectures, we aim to study whether reasoning-chain verification benefits more from a bidirectional understanding model or a generative LLM fine-tuned for discriminative tasks. This comparison has also been explored in prior verifier work [7, 78].

Architecture design: In both cases, we append a lightweight classification head—a single linear layer—on top of the encoder's pooled output representation. This design is common in verifier models [7, 40], as it allows the base encoder to focus on extracting semantically rich features while the classification head maps these to a scalar confidence score. The simplicity of this head also prevents overfitting, especially when fine-tuning on relatively small verifier datasets, and has been shown to be effective in tasks such as fact verification [48] and math reasoning verification [71].

Let z_{ij} denote the scalar logit output for example (q_i, c_{ij}). This value is passed through a sigmoid activation to obtain the probability that the reasoning chain is valid:

$$\hat{y}_{ij} = \sigma(z_{ij}) = \frac{1}{1 + e^{-z_{ij}}}$$

The model is trained using the binary cross-entropy loss:

$$\mathcal{L} = -\left[y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij})\right] \tag{3.11}$$

Here, y_{ij} is the ground-truth label, and $\hat{y}_{ij}$ is the model's predicted probability. This objective encourages the model to output high confidence scores for correct chains and low scores for incorrect ones.

Over time, the verifier learns to internalize answer correctness patterns directly from the natural language content of the reasoning chains, allowing it to act as an effective reward model during inference. The overall data preparation process is shown in Figure 3.5.

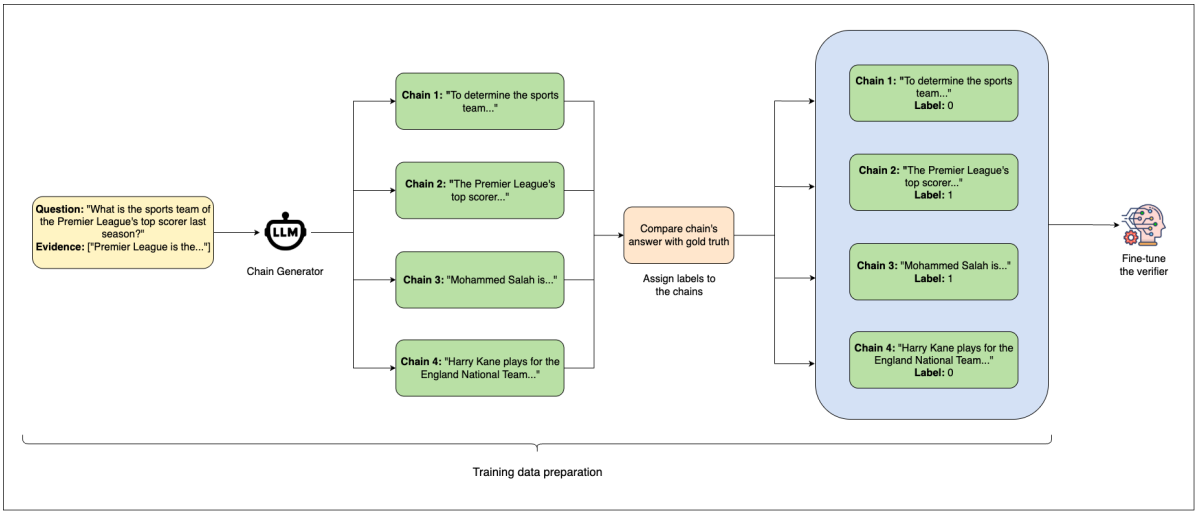

Figure 3.5: Training data preparation pipeline for fine-tuning the verifier. For each question from the training set, multiple reasoning chains are generated using gold evidences. Chains are compared against the ground truth answer and assigned binary labels based on correctness. The labeled examples are then used to fine-tune the verifier model.

3.4.3. Inference using Trained Verifiers

At test time, the trained verifier models are used to select the most reliable reasoning chain for each question. Given a test question q and its corresponding top-k evidences $\{e_1, e_2, \ldots, e_k\}$ retrieved via the pipeline described earlier, we prompt the language model k times to generate k different reasoning chains $\{c_1, c_2, \ldots, c_k\}$. Each chain attempts to answer the original question using the given evidences.

To select the best among these chains, we use a verifier model ψ trained as described in the previous subsection. For each chain c_i, we form an input string by concatenating the original question q and the chain:

$$x_i = \texttt{[Question]:}\ \ q \quad \texttt{[Chain]:}\ \ c_i$$

This input x_i is passed through the verifier ψ to produce a scalar score:

$$z_i = \psi(x_i)$$

which is interpreted as the model's confidence in the correctness of the chain. The chain with the highest score is selected as the final answer:

$$c^* = \arg\max_i \ \psi(x_i) \tag{3.12}$$

Unlike the majority voting baseline, which selects the most frequently occurring an-

swer across generated chains, this verifier-based selection relies on deeper semantic understanding captured during fine-tuning. The verifier can implicitly reason about factual consistency and logical coherence, rather than relying purely on surface-level frequency.

The overall inference process is illustrated in Figure 3.6, where the highest-scoring chain is selected as the best reasoning path. This chain's concluding statement is extracted as the final answer to the original question.

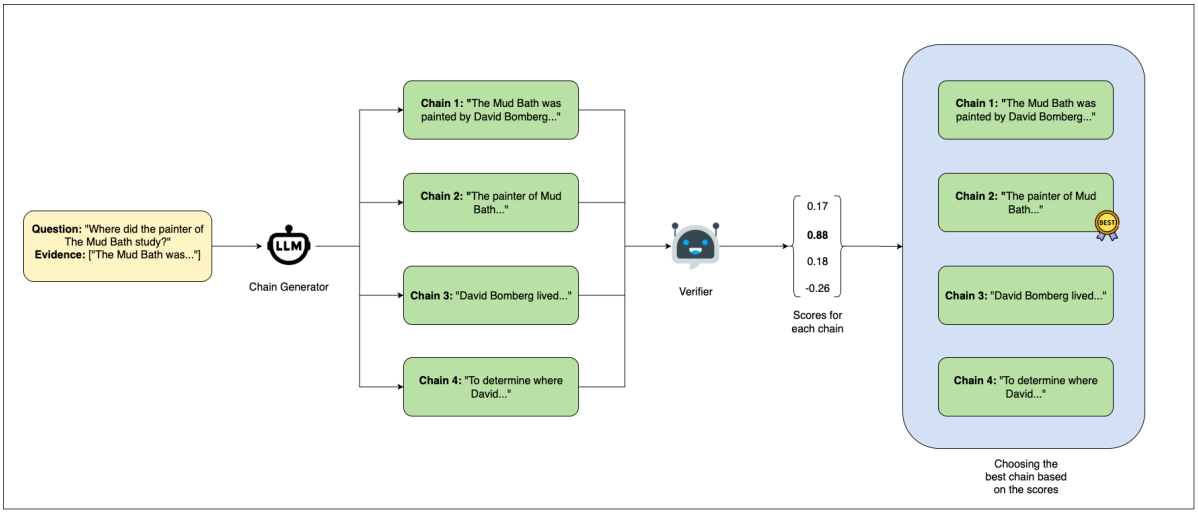

Figure 3.6: Inference-time pipeline using a trained verifier. Given a test question and retrieved evidences, multiple reasoning chains are generated using an LLM. Each chain is scored independently by the verifier, and the chain with the highest score is selected as the final answer.

4

Implementation Details and Experiments

4.1. Datasets

To evaluate the retrieval and reasoning strategies proposed in this work, we use two benchmark datasets that explicitly require multi-hop, compositional reasoning: **MuSiQue (MQA)** [69] and **2WikiMultiHopQA (WQA)** [16]. These datasets have emerged as reliable alternatives to earlier multi-hop QA benchmarks, particularly due to their resistance to shortcut strategies and their enforcement of deeper reasoning. Examples illustrating the structure and type of questions in these datasets are provided in Appendix A.1.

4.1.1. MuSiQue (MQA)

MuSiQue, short for *Multi-hop Questions via Single-hop Question Composition*, was introduced by Trivedi et al. (2022) to address critical flaws in prior multi-hop datasets. Datasets like HotpotQA [89] and WikiHop [81] have been shown to allow shallow heuristics and incomplete evidence chains to suffice for correct answer prediction [46]. In contrast, MuSiQue constructs each question by combining two or more independently sourced single-hop QA pairs that cannot be answered correctly unless the model integrates evidence from both. This compositional nature ensures that models cannot rely on isolated facts or entity co-occurrence.

Each question is accompanied by a small set of gold passages that support the answer, along with a larger pool of challenging distractors. These distractors are topically

similar but do not contain answer-relevant information, thereby increasing the retrieval difficulty. In line with prior work [51], we restrict our evaluation to the 2-hop subset of MuSiQue, which consists of 1,252 questions that require exactly two reasoning steps. This filtered subset presents a clean and focused setting for studying multi-step retrieval and inference.

4.1.2. 2WikiMultiHopQA (WQA)

2WikiMultiHopQA [16] is another multi-hop QA benchmark, constructed by pairing Wikidata triples with corresponding Wikipedia passages. Questions in WQA are explicitly designed to require reasoning across multiple Wikipedia articles, typically involving entities that are connected through intermediate facts or relations. The task setup simulates real-world information-seeking scenarios where users ask comparison questions (e.g., “Who has more Olympic gold medals, X or Y?”), temporal questions (e.g., “Which event happened first?”), or complex fact-synthesis queries.

Each WQA question is paired with sentence-level supporting facts extracted from different documents, making it well-suited for evaluating document retrieval strategies that must not only identify relevant passages but also preserve their contextual alignment. For our experiments, we use a randomly sampled set of 1,200 questions from the development split, covering a broad range of question types and reasoning patterns.

4.2. Hardware Configuration

The experiments were conducted across a combination of compute platforms, each selected to meet the specific requirements of different components in the pipeline.

The bulk of our development and retrieval experiments were carried out on a dedicated remote server located in Germany. The server runs on a Linux environment with an AMD EPYC 7302P processor (32 cores, 3.0 GHz base clock) and is equipped with two NVIDIA RTX 3090 GPUs. This machine offers ample memory capacity (256 GB RAM) and high parallel throughput, making it suitable for both GPU-intensive tasks and large-batch processing during document scoring and re-ranking

To train and fine-tune the verifier models, we used Kaggle’s GPU-accelerated notebooks [1], which provide access to dual NVIDIA T4 GPUs. Each T4 is based on the Turing architecture and features 2,560 CUDA cores, 320 Tensor cores, and 16 GB of GDDR6 memory. With a combined 32 GB of GPU memory, the dual T4 setup en-

[1] https://www.kaggle.com/

ables larger batch sizes and supports deeper model architectures during fine-tuning. Although T4s are not as powerful as A100 or V100 GPUs, they are well-optimized for deep learning workloads and offer a good trade-off between cost-efficiency and computational capability. In our experiments, both the RoBERTa-based verifier and the LLaMA-3.2-3B model were trained on this setup using PyTorch, with mixed-precision training enabled where applicable. All training and evaluation tasks were carried out in interactive Kaggle notebooks, with environment management handled through Conda and pip.

For scaling up the Frontier-Aware Iterative Retrieval (FAIR) experiments, particularly for large-batch inference with LLaMA models, we utilized the DelftBlue supercomputer at TU Delft [1]. We ran jobs on compute nodes equipped with NVIDIA A100 GPUs (40 GB), leveraging SLURM job scheduling and local scratch storage for high-throughput document retrieval and answer tracking. FAIR iterations were parallelized across question batches to optimize GPU utilization under constrained wall-time limits.

4.3. Implementation Details

4.3.1. FAIR

Corpus Construction

To simulate open-domain QA, we construct a shared retrieval corpus by combining all gold contexts from both datasets, along with their associated distractor passages. The distractors are drawn using dense retrievers such as DPR [26] or SPLADE [11] and are intentionally crafted to resemble relevant documents while lacking critical evidence. This setup mirrors the protocol outlined in recent work by Khot et al. [30] and ensures that the retrieval component of the pipeline must operate under noisy and challenging conditions.

In total, the resulting corpus comprises 569,461 Wikipedia passages, serving as the evidence base for both MQA and WQA evaluations. This corpus is used to assess the robustness of retrieval strategies like SUNAR and FAIR under realistic and high-entropy environments.

Retriever Setup

The first stage of the QA pipeline begins with document retrieval. It is a critical component in any open-domain question answering system, particularly in multi-hop settings where the reasoning chain is easily disrupted by missing or noisy evidence. In this work, we adopt the same foundational retrieval design as SUNAR [72], which has

been shown to outperform other dense and sparse retrievers in complex QA scenarios.

For the initial retrieval stage, we use SPLADEv2, a sparsity-aware lexical expansion retriever that has demonstrated strong recall in open-domain settings. SPLADE expands both the query and the documents using transformer-based contextual embeddings, enabling it to capture semantically relevant matches that go beyond exact lexical overlap. SUNAR empirically found SPLADEv2 to outperform other off-the-shelf dense retrievers commonly used in retrieval-augmented generation (RAG) pipelines (see Table 8 in their Appendix), which motivates its use as the retrieval backbone in this work as well.

To model document-document relationships and support neighbourhood-based re-ranking, we construct a document graph following the methodology outlined in SUNAR. Specifically, we use ColBERTv2 to generate document embeddings. ColBERTv2 is a late-interaction retriever, which makes it particularly effective for capturing fine-grained semantic relevance between long-form documents. This property is especially useful when modeling the relationships among multi-paragraph Wikipedia passages.

A K-nearest neighbors (KNN) graph is constructed by performing a similarity search over all document embeddings in the corpus. For each document, the top-100 nearest neighbors are identified based on cosine similarity, and these links are used to build the static document graph. At query time, the system retrieves the top-100 documents using SPLADEv2 and applies neighbourhood-aware re-ranking by exploring the local graph structure. During re-ranking iterations, only 10 neighbors per document are explored to ensure computational efficiency and fair comparison with other baselines that operate on a re-ranking budget of 1,000 documents.

For FAIR, we set the answer frontier size parameter k to 2, meaning that at each iteration, only the two most frequent semantic clusters from the LLM-generated answers are retained in the frontier. This choice allows the system to capture a small set of dominant, semantically consistent answers while still accommodating alternative candidates if agreement is low. The value of k was chosen empirically to balance robustness against noise with the ability to track multiple plausible answers.To score document relevance during re-ranking, we use a cross-encoder model: `nreimers/mmarco-mMiniLMv2-L12-H384-v1` , which is also employed in SUNAR. This model computes pairwise relevance scores between the query and each candidate document, providing higher-fidelity judgments than simple dot-product similarity.

Following SUNAR, we limit the final ranked list to the top-10 documents, which are

passed as context to the language model for answer generation and reasoning. This decision is grounded in their empirical findings: SUNAR compared different top-K cutoffs and found that using more than 10 documents did not yield significant improvements, while increasing inference cost (see Figure 3 in their Appendix). By adhering to the same cutoff, this work maintains consistency with prior evaluations and ensures fair comparability.

Relevance of Decomposed Prompting

The decision to use datasets like MuSiQue and WQA is further motivated by the growing body of research on modular reasoning, most notably the Decomposed Prompting framework proposed by Khot et al. [30]. That work demonstrated that breaking down complex questions into simpler sub-questions, each of which can be tackled independently by a language model, leads to more interpretable and accurate solutions. Importantly, they highlight that datasets like MuSiQue are particularly well-suited for such decomposition-based methods, since each reasoning step in a question corresponds naturally to a discrete inference module.

Our approach builds directly upon this philosophy of compositionality. We use follow-up decomposition to isolate and address intermediate reasoning steps, which then inform both document retrieval (as in SUNAR and FAIR) and inference-time answer selection. In doing so, we move beyond monolithic generation and instead promote multi-stage, verifiable reasoning that better aligns with human-like problem solving. Thus, both our dataset selection and retrieval design are shaped by the modularity principles advocated in decomposition-based QA.

To perform decomposition, we use an instruction-style prompting setup where the model is encouraged to explicitly ask and answer sub-questions before arriving at the final answer. The prompt is constructed using a series of few-shot examples that demonstrate how a complex question can be broken down into follow-up questions, each producing an intermediate answer. The final output is expected only after all intermediate reasoning steps are completed. A snippet of our prompt is shown in Figure A.2 in the Appendix.

The decomposition prompt is used to guide the language model in breaking down a complex question into a sequence of follow-up questions and intermediate answers. This format encourages step-by-step reasoning and culminates in a final answer, aligning with the compositional structure of datasets like MuSiQue and WQA. This format is applied across a diverse set of compositional question types—temporal comparisons, causal reasoning, nested entity relations, and nationality comparisons thus allowing

the model to learn how to reason step-by-step across a variety of patterns. At runtime, the actual test question is appended at the end of this prompt structure, and the model completes the chain of reasoning in a similar fashion. By standardizing decomposition in this way, we ensure consistency and interpretability across both training and evaluation stages.

Model Configuration and Reasoning Setup

The FAIR retrieval framework relies on LLMs not only for final answer generation, but also as an integral part of the iterative retrieval loop. We experiment with two LLMs for meta-reasoning : `LLaMA-3.1-8B-Instruct` [2] and `GPT-4o-mini` [3]. These models are evaluated across both the MQA and WQA datasets under identical retrieval and reasoning conditions.

For the iterative retrieval steps in FAIR, we use `LLaMA-3.1-8B-Instruct`, deployed using the `vLLM` [32] inference engine. The model is configured with `dtype="half"` to enable mixed-precision computation and reduce memory overhead. To support large batch processing during document re-ranking, we set `max_model_len` and `max_num_batched_tokens` to 50,000 tokens and allocate up to 80% of GPU memory via `gpu_memory_utilization=0.80`. We also enable tensor parallelism across two GPUs (`tensor_parallel_size=2`) and disable chunked prefill to optimize latency during re-ranking. The model is run in eager mode for compatibility with `vLLM`'s batching backend.

In each retrieval iteration, the model receives a conversational prompt consisting of the original or follow-up question along with the currently retrieved document batch. Sampling is performed with a temperature of 0.3 and `top_p` = 1.0 (nucleus sampling), while restricting generation to a maximum of 256 tokens. A stopping condition is defined based on the presence of a specific intermediate tag, ensuring that partial answers or unsupported hallucinations are filtered before contributing to the frontier update.

To evaluate the robustness of FAIR under different reasoning substrates, we also perform parallel experiments using OpenAI's `GPT-4o-mini` model. This model is accessed via the OpenAI API and configured with similar decoding parameters: `temperature=0.3`, `top_p=1.0`, and `max_tokens=2048`. For each question, the prompt combines structured information from the table, textual evidence, and the question itself, with a system instruction that enforces answer derivation through multi-step rationale generation. We set the frequency penalty to 0.8 and the presence penalty to 0.6 respectively, discouraging repetitive or overconfident generations. Multiple answer

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
[3] https://platform.openai.com/docs/models/gpt-4o-mini

candidates are sampled per query (`n=self.top_n`) to enable downstream selection using verifier models.

The use of two distinct LLMs allows us to evaluate FAIR's retrieval stability across generation styles—one open-source and instruction-tuned (`LLaMA-3.1-8B-Instruct`), the other closed-source and commercially hosted (`GPT-4o-mini`). The underlying infrastructure for `LLaMA`-based experiments runs on multi-GPU servers and the DelftBlue A100 cluster, while `GPT-4o` queries are executed remotely via API.

4.3.2. Test-Time Scaling with Verifier Models

Multi-Chain Generation

As discussed in 3.4.3, for each question in the test set, we generate 10 distinct reasoning chains, each consisting of step-by-step rationale followed by a final answer. These chains are constructed using OpenAI's `GPT-4o-mini` model with a temperature of 0.3, ensuring diversity while maintaining factual coherence. The prompting format closely follows that of the FAIR meta-reasoning framework, where the model is given both the question and the previously constructed reasoning path, along with the top-k evidences (retrieved using either SUNAR or FAIR). Few-shot examples are prepended to the prompt to demonstrate the required format, using tags like `[Answer]:` and `[Final Answer]:` to clearly delineate reasoning from conclusion.

Chain generation is performed for both the MuSiQue (MQA) and 2WikiMultiHopQA (WQA) datasets, under two evidence settings: one using SUNAR's retrieved evidence as context, and the other using FAIR's retrieved evidence as context. This setup enables a controlled comparison of how different retrieval strategies interact with generation diversity and answer selection.

Chain Selection Using Prompt-Based Verifiers

Once the reasoning chains are generated, we use a verifier model to identify the most trustworthy one. The verifier is prompted with the question and all ten candidate chains, formatted as a numbered list. It is then asked to evaluate each chain in terms of logical consistency, factual accuracy, and directness, and to select the best among them. The instruction is explicit: if multiple chains are valid, the verifier should favor the most concise and question-aligned one.

We conduct this evaluation using two different verifier models:

- `GPT-4o-mini` (via OpenAI API)
- `DeepSeek R1-7B`, an instruction-tuned open-source model optimized for step-by-

step reasoning

Both models are queried using the same prompt structure, with decoding temperature set to 0.3 and `top_p` to 1.0, and the output is parsed to extract the selected chain index. The final answer is then extracted from the chain using a regex pattern that targets the `[Final Answer]:` tag.

This verifier-based re-ranking is applied to all generated chains across both datasets and evidence settings. As we show in the results section, this slightly boosts cover-EM scores compared to relying on the default chain or a majority-vote heuristic.

Verifier Training: RoBERTa and LLaMA

In addition to evaluating open-source and prompt-based language models for verifier-based answer selection, we also trained two dedicated verifier models: one based on `RoBERTa` and the other on `LLaMA-3.2-3B`. These models were trained to judge the quality of reasoning chains by classifying whether a given chain led to the correct final answer. The training dataset for both models was created by generating multiple chains per question from the training split of the MuSiQue and 2WikiMultiHopQA datasets, using `GPT-4o-mini` as the generator.

Training Data Construction

As discussed in 3.4.2, to construct the training data for verifier models, we generated 5 distinct reasoning chains for each question in the training split of the dataset, using `GPT-4o-mini` with few-shot prompting. This process was repeated for 5,000 questions, yielding a total of 25,000 chains.

However, this initial dataset was imbalanced. Only 6,532 of the 25,000 chains were labeled as incorrect, while the remainder were correct. To avoid bias during training, we constructed a balanced dataset by sampling an equal number of correct chains. This resulted in 13,064 chains in total (half correct and half incorrect), ensuring that the verifier learned to distinguish strong reasoning paths from faulty ones rather than defaulting to the majority class.

To prepare the data for training, we applied stratified sampling to split the balanced dataset into training and validation sets while preserving label proportions. Specifically, 80% of the examples were used for training and the remaining 20% for validation. Each training instance was formatted as a prompt combining the question and its associated reasoning chain, with the label appended at the end. Validation instances were formatted similarly, but with the label left blank, allowing the model to predict it.

Split	# Chains	Correct	Incorrect
Initial Generation	25,000	18,468	6,532
Balanced Dataset	13,064	6,532	6,532
Train (80%)	10,451	5,225	5,226
Validation (20%)	2,613	1,307	1,306

Table 4.1: Data composition and splits for training the verifier model. The *Initial Generation* set contains all generated reasoning chains, labeled correct or incorrect based on gold answers. A *Balanced Dataset* is created by downsampling the majority class, and then split into 80% *Train* and 20% *Validation* sets.

RoBERTa-Based Verifier

To train the RoBERTa-based verifier, we used Hugging Face's `Trainer` API with `roberta-base` as the underlying model. The input format for each instance consisted of the question and the corresponding reasoning chain, followed by a classification label (`correct` or `incorrect`). During training, the model was optimized using AdamW with a learning rate of 2×10^{-5} and a batch size of 16. Evaluation was performed every 500 steps, and the best model checkpoint was selected based on validation accuracy. We also enabled early stopping with a patience of two evaluation intervals to prevent overfitting. This training setup allowed the model to focus on distinguishing logically consistent and semantically aligned chains from those that were either incorrect or incoherent.

LLaMA-Based Verifier

The LLaMA-based verifier was trained using a similar dataset, but implemented in a custom training loop outside the Hugging Face framework. We used the `meta-llama/Llama-3.2-3B-Instruct` model as the base, and added a lightweight binary classification head on top. Tokenization was handled using the corresponding LLaMA tokenizer, and input sequences were capped at 512 tokens. The training loop was implemented in PyTorch, with a `BCEWithLogitsLoss` objective and a cosine learning rate scheduler with 5% warmup. We used AdamW for optimization with a learning rate of 1×10^{-3} and a batch size of 16. During training, the model's performance on the validation set was tracked, and early stopping was applied using validation loss as the criterion. Unlike the RoBERTa setup, this model was trained outside of the Transformers library and made use of a manually managed training loop to allow greater flexibility in customizing components like the learning rate scheduler and model saving routines.

4.4. Evaluation Metrics

4.4.1. Quantitative Metrics

For quantitative evaluation, we use the *cover-EM* metric to measure answer correctness. This metric, proposed by Rosset et al. [60], considers an answer correct if the gold (ground-truth) answer is a substring of the model's predicted answer. This offers a relaxed and more robust evaluation compared to exact match, particularly in multi-hop settings where answer phrasing can vary despite being semantically correct. We follow the same cover-EM formulation adopted by recent work such as Press et al. [51], allowing for a fair comparison to systems like SUNAR.

To evaluate retrieval quality, we report *Recall@k*, where a prediction is considered successful if at least one gold evidence passage appears among the top-k retrieved documents. This is calculated separately for both the SUNAR-based and FAIR-based pipelines, giving us insight into how each method performs in isolating relevant context. We report Recall@1 and Recall@10 for both the experiments, aligning with the standard used in multi-hop open-domain QA benchmarks.

4.4.2. Qualitative Analysis

Alongside quantitative evaluation, we also conduct a qualitative analysis of the reasoning chains generated during inference. This is particularly important in multi-step question answering, where accuracy metrics alone may fail to capture the nuances of model behaviour. For example, a correct final answer may mask flaws in intermediate reasoning steps, such as unsupported claims, overlooked evidence, or incorrect intermediate computations that would undermine the robustness of the model in a real-world setting. Conversely, a logically sound and well-supported reasoning chain may be penalized under strict evaluation criteria for superficial lexical mismatches or differences in abstraction level between the model's output and the gold answer.

To systematically capture these nuances, we adopt a grounded theory-inspired approach [82], applying the constant comparison method [15] to iteratively examine and group individual reasoning instances into coherent categories. Here, each instance refers to a full reasoning chain produced by the model for a given question, including intermediate steps and the final prediction. By continuously comparing instances, we identify recurring reasoning strategies, common failure modes, and qualitative improvements achieved by specific inference settings.

In some cases, models produce partially correct reasoning that fails only at the final aggregation step, suggesting opportunities for targeted post-processing or inter-

mediate verification. In others, the reasoning may drift away from the retrieved evidence, highlighting limitations in retrieval relevance or in the model's ability to stay grounded. Such analyses also help detect cases where dataset annotation choices such as overly strict string matching or ambiguous gold answers introduce apparent errors that are not genuine reasoning failures.

Our qualitative analysis therefore complements the aggregate scores by revealing where performance bottlenecks truly lie: whether in retrieval coverage, reasoning robustness, answer abstraction mismatches, or annotation inconsistencies. These insights are critical for guiding future improvements, as they point to failure modes that may not be apparent from quantitative metrics alone.

5

Results

5.1. Evaluating FAIR Retrieval

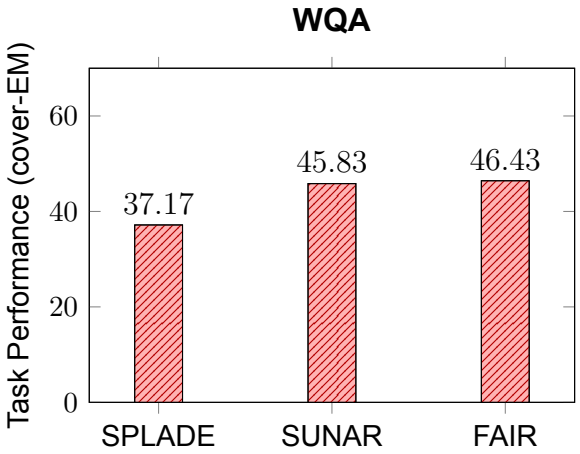

Figure 5.1: Task performance of different retrieval methods on the WQA dataset in a zero-shot setting performed using `gpt-4o-mini`.

Before diving into complex reasoning or verifier-based answer selection, it is important to first assess the quality of retrieval alone (without any downstream re-ranking or reasoning). To that end, we compare the effectiveness of FAIR against existing methods like SPLADE and SUNAR in a simple zero-shot setup.

The idea is straightforward: given a question, we retrieve documents using each method and ask a language model (`GPT-4o-mini`) to generate a direct answer without any chain-of-thought reasoning or verification. This isolates the contribution of retrieval quality in the overall QA pipeline.

As shown in Figure 5.1, **FAIR outperforms both SPLADE and SUNAR** on the WQA benchmark. In particular, FAIR achieves 46.43% cover exact match (cover-EM) accuracy in this zero-shot setup, compared to 45.83% with SUNAR and 37.17% with SPLADE. While the absolute gains over SUNAR may seem modest in this specific configuration, they are consistent–and more importantly, they are backed by deeper improvements in evidence quality, as we will explore shortly.

To further quantify FAIR's ability to retrieve relevant information, we also evaluate Recall@k across both the MQA and WQA benchmarks. As shown in Table 5.1, FAIR achieves the highest recall at the top-10 cutoffs across both datasets, even outperforming SUNAR. This demonstrates that FAIR does a better job at keeping gold evidence passages "in play," which in turn provides a stronger foundation for subsequent reasoning modules.

Retriever	**MQA**		**WQA**	
	R@1	**R@10**	**R@1**	**R@10**
SELF-ASK(Re-Rank)	0.157	0.309	0.230	0.402
SELF-ASK(Retrieval)	0.152	0.240	0.205	0.399
SUNAR	**0.284**	0.459	0.287	0.606
Frontier Based Retrieval (FAIR)	0.254	**0.490**	**0.288**	**0.629**

Table 5.1: Retrieval performances on MQA and WQA.

5.1.1. Qualitative Analysis

While quantitative metrics such as Recall@k provide a broad view of retrieval effectiveness, they do not always capture the qualitative differences in the types of evidence retrieved. To further illustrate the strengths of **FAIR** over **SUNAR**, we present a comparative case study (Table 5.2) involving the following question:

> *Which film has the director who died first — Crimen A Las Tres or The Working Class Goes To Heaven?*

To answer this, the system must identify both directors and compare their dates of death. **FAIR** succeeds in this instance by retrieving all the necessary facts: that *Crimen A Las Tres* was directed by Luis Saslavsky, who died in 1995, and that *The Working Class Goes To Heaven* was directed by Elio Petri, who died earlier, in 1982. With this information clearly present in the evidence set, the model confidently returns the correct answer: *The Working Class Goes To Heaven*.

SUNAR, on the other hand, fails despite retrieving some relevant passages. While it does include multiple documents about Luis Saslavsky and accurately notes his death

date, it fails to retrieve any document explicitly stating when Elio Petri died. Instead, it retrieves indirect mentions of Petri through the lens of his filmography and collaborations with actor Gian Maria Volonté, including a paragraph that confusingly focuses on Volonté's death in 1994. Lacking concrete evidence about Petri's death, the model incorrectly assumes that Luis Saslavsky died first, resulting in a wrong answer.

This example highlights a key advantage of FAIR's approach: by maintaining an evolving *answer frontier* and rewarding evidence batches that produce consistent answer candidates, FAIR encourages the retrieval of complementary information — in this case, details about *both* directors. SUNAR's uncertainty-penalization strategy, on the other hand, does not explicitly reward such complementary coverage, leading it to fixate on one entity and overlook the other. As a result, even in seemingly straightforward comparative questions, SUNAR can falter if the retrieval process lacks global consistency.

Method	**Evidences**
Question: Which film has the director died first, *Crimen A Las Tres* or *The Working Class Goes To Heaven*? [Dataset: **WQA**]	
FAIR	**[Evidence 1]** The Working Class Goes to Heaven is a 1971 political drama film directed by Elio Petri. **[Evidence 2]** Crimen a las tres is an Argentine crime film directed by Luis Saslavsky. **[Evidence 3]** Luis Saslavsky was born in 1903 and died on March 20, 1995. **[Evidence 4]** Elio Petri died in 1982. **[Final Answer]:** The Working Class Goes To Heaven
SUNAR	**[Evidence 1]** Crimen a las tres is an Argentine crime film directed by Luis Saslavsky. **[Evidence 2]** Luis Saslavsky was born in 1903 and died in 1995. **[Evidence 3]** The Working Class Goes to Heaven is a 1971 film directed by Elio Petri. **[Evidence 4]** Gian Maria Volonté, who acted in Petri's films, died in 1994. **[Final Answer]:** Crimen A Las Tres

Table 5.2: Qualitative comparison of evidences retrieved by **FAIR** and **SUNAR** for a comparative temporal reasoning question. FAIR successfully retrieves and connects both death dates, while SUNAR fails to find Petri's death information and makes an incorrect prediction.

The previous example demonstrated FAIR's advantage in handling comparative temporal reasoning questions, where success depends on retrieving complementary facts about multiple entities. FAIR's frontier-based retrieval encouraged balanced cover-

age of both directors' death dates, enabling it to correctly answer the question, while SUNAR's uncertainty-penalization approach failed to capture the complete picture.

We now turn to an example from the **MQA** dataset, which further highlights FAIR's ability to gather all the critical facts needed for reasoning. In this case, the question asks:

> *"What is the name of the food safety system of the federal agency that regulates prescription drugs?"*

To answer this question, a system must first identify the relevant federal agency — the *U.S. Food and Drug Administration* (FDA) — and then retrieve the specific name of its food safety system. This requires connecting two pieces of information: the agency responsible for prescription drug regulation and the food safety system it administers.

FAIR handles this process effectively. It begins by confirming that the FDA regulates prescription drugs, as seen in multiple supporting evidences. Crucially, it then retrieves a passage explicitly naming the *Food Safety Modernization Act (FSMA)* as the FDA's food safety system. This direct linkage between the FDA and FSMA makes the reasoning step straightforward, allowing FAIR to confidently produce the correct answer: *Food Safety Modernization Act (FSMA)*.

SUNAR, in contrast, retrieves incomplete and somewhat unfocused evidence. While it does correctly identify the FDA as the relevant agency, it fails to retrieve any document explicitly mentioning FSMA. Instead, its evidence set contains tangential details about prescription drug importation and regulatory practices, with no direct connection to the FDA's food safety framework. Lacking this crucial link, SUNAR defaults to producing an incomplete and incorrect answer: FDA regulations.

This example illustrates how FAIR's *answer frontier* mechanism rewards retrieval batches that contribute complementary information, ensuring that all reasoning steps are fully supported. SUNAR's approach, while effective in some contexts, can result in fixation on partial information (as seen in Table 5.3), leading to missing key facts and ultimately producing an incorrect answer.

Method	Evidences
Question: What is the name of the food safety system of the federal agency that regulates prescription drugs? [Dataset: **MQA**]	
FAIR	**[Evidence 1]** …FDA or U.S. Food and Drug Administration, is an agency responsible for the control and safety of food and drugs. **[Evidence 2]** The United States Federal Food, Drug, and Cosmetic Act …gives authority to the FDA to oversee the safety of food, drugs, medical devices, and cosmetics. **[Evidence 3]** The Food Safety Modernization Act (FSMA) …grants the FDA new authorities to regulate the way foods are grown, harvested and processed. **[Final Answer]:** Food Safety Modernization Act (FSMA)
SUNAR	**[Evidence 1]** In the United States, there has been a push to legalize importation of medications …while in most cases importation of prescription medications violates FDA regulations. **[Evidence 2]** Ben Goldacre has argued that regulators …such as the FDA in the United States …advance the interests of the drug companies rather than the public. **[Final Answer]:** FDA regulations

Table 5.3: Qualitative comparison of evidences retrieved by **FAIR** and **SUNAR** for an MQA example. FAIR successfully retrieves both the agency (FDA) and the food safety system (FSMA), while SUNAR retrieves only partial information and misses the crucial FSMA mention.

Key Insight

Enhancing retrieval quality has a direct and measurable impact on end-to-end QA performance. Even without additional reasoning or re-ranking steps, stronger retrieval ensures that more relevant evidence is available to the model, leading to higher answer accuracy.

5.2. Impact of Evidence Quality on Downstream Reasoning

Model	MQA	WQA
gpt-4o-mini		
SELF-ASK	26.76	37.33
SUNAR	32.19	48.16
FAIR	**33.06**	**48.53**
LLaMA 3.1 (8B)		
SELF-ASK	5.43	25.83
SUNAR	13.82	39.52
FAIR	**16.30**	**42.30**
DeepSeek-R1 (7B)		
SPLADE only	15.10	42.13

Table 5.4: Single-shot chain-of-thought meta-reasoning results across different LLM substrates. FAIR and SUNAR use their respective retrieval methods, while DeepSeek operates in a zero-shot setting with SPLADE's top-10 evidences and performs its own internal test-time scaling. In both cross-model and same-model setups, FAIR's retrieval consistently leads to stronger downstream reasoning performance compared to SUNAR, and improves over DeepSeek despite the latter's built-in TTS mechanism.

Beyond direct retrieval metrics, it is also important to assess whether the improvements in evidence quality offered by FAIR translate into gains for downstream reasoning tasks. Table 5.4 presents results for a single-shot chain-of-thought meta-reasoning setup across the **MQA** and **WQA** datasets, comparing FAIR, SUNAR, and SELF-ASK under different LLM substrates.

In the first setting, evidence retrieval and reasoning are decoupled: FAIR's evidences are retrieved using `LLaMA-3.1-8B-Instruct` but the reasoning step is carried out with the stronger `GPT-4o-mini` model. Despite operating with retrieval from a smaller model, FAIR still slightly outperforms SUNAR on both datasets. This is particularly notable given that SUNAR's retrieval is powered by `GPT-3.5-turbo`[1], a larger model than the retriever used for FAIR in this setup. These results suggest that FAIR's retrieval strategy—by virtue of producing more complementary and targeted evidence—can provide a quality advantage that compensates for the smaller retriever size, resulting in better final reasoning accuracy.

In the second setting, both retrieval and reasoning are performed with the same `LLaMA-3.1-8B-Instruct` model. Here too, FAIR maintains its advantage over SUNAR,

[1] https://platform.openai.com/docs/models/gpt-3.5-turbo

achieving 42.30% on WQA compared to SUNAR's 39.52%. This indicates that the gains from FAIR's frontier-aware retrieval mechanism are not confined to cross-model pipelines, but also hold when reasoning is done with the same model used for retrieval. A plausible explanation for this trend is that FAIR's frontier consistency mechanism increases the likelihood of retrieving complementary pieces of information that address all aspects of the question, rather than over-concentrating on a subset of entities or facts. This broader yet targeted coverage appears to reduce reasoning gaps downstream, even when the reasoning model itself is relatively small. Conversely, SUNAR's uncertainty-penalization approach does not explicitly encourage such coverage, making it more susceptible to partial retrievals that limit reasoning effectiveness.

We also benchmark a third setting using `DeepSeek-R1-7B`, a reasoning model with a built-in zero-shot test-time scaling mechanism. DeepSeek internally generates and aggregates multiple reasoning chains before producing its answer. Supplied with SPLADE-retrieved evidences, DeepSeek achieves 42.13% cover-EM on WQA surpassing `LLaMA-3.1-8B-Instruct` with SUNAR retrieval. However, FAIR still outperforms DeepSeek in the same dataset, despite DeepSeek's internal TTS. This highlights a critical point: **reasoning quality is fundamentally constrained by evidence quality**. Even a strong reasoning model cannot compensate for incomplete, redundant, or tangential retrievals. FAIR mitigates this by promoting documents that reinforce consistent answers, ensuring that any reasoning process (be it single-shot, verifier-assisted, or internally aggregated) operates over high-value, diverse, and relevant context.

These results reinforce the central hypothesis of this work: *improvements in retrieval quality propagate through the QA pipeline, enhancing final answer accuracy even when reasoning is performed by a stronger model or aided by built-in TTS mechanisms.* Retrieval optimization and reasoning optimization are not competing strategies but complementary ones.

Having established that FAIR's retrieval quality can translate into measurable reasoning improvements, we now turn to scenarios where reasoning performance is pushed further through *test-time scaling* and the use of *verifiers* to select the best answers from multiple reasoning paths.

5.3. Verifier-Driven Gains in Multi-Chain Reasoning

Method	MQA		WQA	
	SUNAR	FAIR	SUNAR	FAIR
Upperbound (Golden Evidences)				
gpt-4o-mini	**80.67**		**88.75**	
Upperbound (Ideal Verifier)				
gpt-4o-mini	38.42	40.85	58.75	60.98
Single Shot Prompting (No TTS)				
gpt-4o-mini	32.19	33.06	48.16	48.53
Prompt-based Verifier				
gpt-4o-mini	35.75 (+11.06%)	35.81 (+8.32%)	53.67 (+11.44%)	54.84 (+13.00%)
DeepSeek	35.46 (+10.16%)	34.69 (+4.93%)	51.92 (+7.80%)	52.40 (+7.97%)
Majority Voting				
gpt-4o-mini	34.03 (+5.72%)	34.69 (+4.93%)	50.50 (+4.86%)	50.63 (+4.33%)
Verifier (Best-of-N)				
RoBERTa Verifier	35.22 (+9.41%)	35.41 (+7.10%)	53.67 (+11.44%)	54.67 (+12.65%)
LLaMA-based Verifier	34.37 (+6.77%)	34.90 (+5.57%)	54.25 (+12.65%)	54.50 (+12.30%)

Table 5.5: Cover-Exact Match (Cover-EM) performance on the MQA and WQA datasets under different retrieval settings (**SUNAR** and **FAIR**) and answer selection methods. The table also reports the **Upperbound (Golden Evidences)** score, representing the maximum attainable performance if gold-standard evidences were provided, and the **Upperbound (Ideal Verifier)** score, which assumes perfect chain selection given the retrieved evidences. Percentage gains (in green) are computed relative to the *Single Shot Prompting (No TTS)* baseline.

In this part of the evaluation, the focus shifts from retrieval to what happens after the relevant evidence has been gathered. Here we see how multiple reasoning chains can be leveraged at inference time, and how the choice of a final answer from these chains impacts performance. The experimental setup for this has already been discussed in 4.3.2.

We begin this analysis by examining the performance of DeepSeek that incorporates its own internal mechanism for test-time scaling in a zero-shot setting (as discussed in the previous section). In our experiments, DeepSeek is paired with the top-10 SPLADE retrieved evidences, relying entirely on its built-in approach to generate and select answers from multiple reasoning paths without the use of an external verifier. While such implicit test-time scaling can provide a level of robustness, its effectiveness is constrained by the model's ability to accurately judge the quality of its own outputs, a challenge that decoder-only models often face [85].

In contrast, explicit test-time scaling with a dedicated verifier separates the generation and selection stages. Multiple candidate reasoning chains are first produced by the reasoning model, after which a verifier, trained specifically to detect subtle factual or logical inconsistencies, scores and ranks these candidates. This separation allows

even smaller reasoning models to achieve higher final accuracy when supported by a strong verifier and high-quality evidences.

The benefits of this explicit approach become more apparent when combined with FAIR retrieval. FAIR not only increases the coverage of essential facts but also promotes the retrieval of complementary evidences that collectively address all aspects of the question. This leads to a more diverse and complete set of candidate chains, which in turn increases the likelihood that a capable verifier will identify and select the correct one. Consequently, the combination of FAIR retrieval with verifier-based test-time scaling surpasses the performance of DeepSeek's implicit scaling with SPLADE retrieval, illustrating the advantage of decoupling reasoning from answer selection and of supplying the verifier with stronger, more targeted evidences.

The **Upperbound (Golden Evidences)** in Table 5.5 captures the most optimistic scenario possible. Here, the model is fed gold-standard evidences (human-annotated supporting documents) rather than retrieved ones. The evaluation criterion is generous: if *any* of the ten generated answers is correct, the question is counted as correct. This means we are isolating the *generation ceiling* — the maximum attainable performance if retrieval were flawless and the verifier could, in principle, choose correctly every time.

The **Upperbound (Ideal Verifier)** is conceptually similar but uses actual evidences retrieved by SUNAR or FAIR. The same generous "at least one correct chain" rule is applied, but now the context is constrained to whatever the retrieval stage produced. This upper bound therefore reflects the combined effect of retrieval coverage and reasoning diversity, assuming an oracle verifier that never makes a wrong selection.

An interesting pattern emerges here. For WQA, the gap between the two upper bounds is relatively small, suggesting that when using high-quality retrieval (as in FAIR), most of the essential information is already present, which perhaps helps in better reasoning. In MQA, however, the gap is significantly larger. Our interpretation is that MQA's multi-hop structure, with its longer and often noisier reasoning chains, makes retrieval quality a much stronger bottleneck. Even an ideal verifier cannot compensate if the retrieved set is missing key bridging facts or if the evidences are fragmented across multiple documents that never appear together in a single chain. We return to this point later in Section 5.4, where we break down specific MQA error cases.

When we move from these theoretical limits to actual verifiers, the trade-offs become more pronounced. The RoBERTa-based verifier stands out: across both datasets, it

achieves results that are nearly on par with GPT-4o-mini, despite being far smaller and faster. This is a critical insight. GPT-4o-mini, while powerful, is computationally expensive for large-scale deployment. A compact discriminative model like RoBERTa offers a practical, cost-efficient, and scalable alternative without sacrificing much accuracy. In operational terms, this means that once a verifier of this quality is trained, we could run large inference batches at a fraction of the computational and financial cost.

Q: *Where was the director of Ronnie Rocket born?*

Single-shot

Pred: United States
Gold: Missoula, Montana

Test-time scaling (10 chains)

Answer	Count
Missoula / Montana	8
United States	2

Verifier pick: Missoula, Montana

Single-shot predicted *United States* (wrong). Multiple chains produced *Missoula, Montana*, which the verifier selected correctly.

Figure 5.2: Test-time scaling improves accuracy by sampling multiple reasoning chains and letting the verifier pick the best one.

The LLaMA-based verifier performs respectably but consistently trails RoBERTa. One plausible explanation lies in the nature of the task itself. Selecting the best reasoning chain from a set of candidates is a discriminative classification problem rather than a generative one. Architectures such as RoBERTa are pretrained using masked language modeling (MLM), which has been shown to produce contextualized representations that are highly effective for downstream classification and ranking tasks [9, 42]. This pretraining paradigm directly encourages the model to capture bidirectional dependencies and to make fine-grained token-level distinctions, both of which are useful when identifying subtle factual or logical inconsistencies between candidate answers.

By contrast, LLaMA is an autoregressive decoder-only model optimized for left-to-right generation. While such models excel at producing coherent and contextually plausible text, their representations are not inherently tuned for discriminative decision-making [53, 77]. Without substantial task-specific adaptation, generative models may be less sensitive to small but critical factual mismatches or reasoning errors that distinguish a “good” chain from a superficially coherent yet incorrect one. Similar observations

have been made in verifier-oriented studies, where discriminative encoders consistently outperform decoder-only models of similar size when applied to answer selection or ranking tasks [7, 40]. This architectural bias likely explains the performance gap observed here, despite both models being fine-tuned on the same verification dataset.

The performance of DeepSeek as a verifier is another point worth discussing. While DeepSeek is marketed as a reasoning-optimized model, its results here lag behind both prompt-based and discriminative verifiers. There are several factors that could explain this. First, DeepSeek is primarily optimized for producing reasoning, not for evaluating and ranking existing reasoning. Second, as discussed earlier in this section, decoder-only generative models often lack calibration when judging their own or others' outputs — a well-known issue in verifier tasks [85]. Third, the prompts we used were adapted from generative settings and may not fully exploit DeepSeek's strengths for discriminative ranking. Without specialized tuning for this exact decision-making role, its ability to catch subtle logical flaws or factual mismatches may be limited.

5.3.1. Qualitative Analysis

Temporal Ordering Resolution (Date Comparison)

Following the methodology in Section 4.4.2, we analyzed 30 samples to define a taxonomy of improvements in verifier-selected output sequences. The most common category, **Temporal Ordering Resolution**, accounted for 60% of cases. These questions are solved by extracting two or more dates from the evidence and comparing them. Single-shot chains often miss one date or skip the comparison, leading to errors, especially when distractor documents contain similar names or irrelevant temporal information.

Unlike single-shot prompting, TTS does not rely on a single reasoning path; instead, it generates multiple candidate chains, increasing the chance that at least one captures all necessary details. The verifier then favours chains that explicitly extract all relevant dates and perform the correct chronological comparison. In the example shown in Figure 5.3, the single-shot approach fails to locate Whitaker's death date due to confusion from distractors, whereas TTS with verifier correctly extracts both death dates (1698 and 1916) and concludes Fleetwood Sheppard died first. This pattern illustrates how verifier-driven selection can systematically reduce reasoning failures caused by partial or incomplete evidence usage.

[Claim]: Who died first, Fleetwood Sheppard or George William Whitaker?

[Evidence]:

1. Fleetwood Sheppard (1 January 1634 – 25 August 1698).
2. George William Whitaker (25 September 1840 – 6 March 1916).
3. Frederick Arthur Whitaker (1893 – 1968), British civil engineer.
4. Curtis Sheppard, American boxer, no recorded death date.
5. William Whitaker (1629 – 1672), English ejected minister.
6. William Whitaker (1923 – 1995), English footballer.

Reasoning without TTS (Single-shot Prompting)
[Prediction]: Uncertain
[Reasoning]: Misses Whitaker's death date due to distractors with similar surnames. Unable to compare both dates, so fails to determine who died first.

Reasoning with TTS + Verifier
[Prediction]: Fleetwood Sheppard
[Reasoning]: Extracts both death dates (1698 and 1916) from relevant evidence. Compares them and concludes Fleetwood Sheppard died first.

Figure 5.3: Example of **Temporal Ordering Resolution** (Date Comparison). Relevant evidence is highlighted in blue. Single-shot prompting fails due to distractors, whereas TTS + Verifier correctly extracts and compares dates to identify Fleetwood Sheppard as the earlier death.

Familial Relationship Reasoning

Another frequent improvement pattern, observed in 8 out of the 30 analyzed cases (26.6%), involves reasoning over genealogical links between entities. We refer to this as **Familial Relationship Reasoning**. These questions typically require identifying an entity's relative (e.g., paternal grandmother) by traversing a short chain of family connections. For instance, answering "Who is Blanche of Portugal's (1259–1321) paternal grandmother?" requires (i) identifying her father, King Afonso III of Portugal, and (ii) determining that Afonso III's mother was Urraca of Castile.

Single-shot prompting often fails in such cases due to the presence of distractor entities in the retrieved evidence (see Fig 5.4). When only one reasoning chain is generated, any incorrect early step such as associating the subject with the wrong family branch propagates to the final answer without correction.

Test-time scaling (TTS) improves robustness here through two mechanisms. First, generating multiple reasoning chains increases diversity in entity linking paths: some chains may follow distractor passages, but others correctly trace the lineage. Second, the verifier filters for logically complete and internally consistent chains, often favoring

those that explicitly identify the intermediate relative before producing the final answer. As a result, TTS greatly increases the likelihood that at least one chain avoids distractors and executes the correct multi-step reasoning, making it more resilient than single-shot prompting for this class of questions.

[Claim]: Who is Blanche Of Portugal (1259–1321)'s paternal grandmother?

[Evidence]:

1. Blanche of Portugal (1259–1321) was the daughter of King Afonso III of Portugal and his second wife Beatrice of Castile.
2. Afonso III of Portugal was the son of King Afonso II of Portugal and Urraca of Castile.
3. Urraca of Castile (1148–1211) was queen consort of León.
4. Other biographies of unrelated historical figures.

Reasoning without TTS (Single-shot Prompting)
[Prediction]: Queen Maud of Savoy
[Reasoning]: Correctly identifies Blanche's father but incorrectly links to an unrelated grandmother, likely due to confusion from distractor biographies.

Reasoning with TTS + Verifier
[Prediction]: Urraca of Castile
[Reasoning]: Explicitly resolves both hops — identifies Blanche's father (Afonso III), then finds his mother (Urraca of Castile) — yielding the correct answer despite distractors.

Figure 5.4: Example of **Familial Relationship Reasoning**. Relevant evidence is highlighted in blue. Single-shot prompting fails due to an incorrect link in the family chain, whereas TTS + Verifier correctly traces the paternal line to Urraca of Castile.

The majority voting baseline provides a useful contrast. It is appealing in its simplicity — no extra model, just count and choose — but its inherent weakness becomes clear in multi-hop QA. If the majority of chains converge on an incorrect conclusion (often with high internal consistency), majority voting will confidently amplify the wrong answer. In fact, in several qualitative cases we examined (an example shown in Fig 5.5), the majority answer was wrong because the reasoning model made the same early incorrect assumption in multiple chains, leading to a consistent but incorrect final answer. This limitation makes majority voting brittle in domains where plausible distractors are common and nuanced reasoning steps are critical.

Question: *Where did the director of Border (2018 film) graduate from?*

Answer	Count	Tag
Unknown	4	Majority vote (wrong)
Stockholm University	2	Wrong
National Film School of Denmark	**1**	Gold / correct
Denmark and Sweden	1	Wrong
National Film School in Denmark	2	Wrong

Takeaway: Majority voting selected *Unknown* (4), which is incorrect, while the gold answer *National Film School of Denmark* appears only once.

Figure 5.5: Majority voting can fail when the most frequent answer is wrong.

Overall, these results highlight a key finding: the quality of the verifier directly impacts the effectiveness of test-time scaling. Even small improvements in verifier accuracy can meaningfully close the gap to the upper bound, particularly when retrieval is already strong. FAIR's advantage in retrieval indirectly benefits this stage as well — higher-quality evidence leads to more high-quality chains, which in turn increases the likelihood that a competent verifier, whether GPT-4o-mini or RoBERTa, will pick the correct one.

Key Insights

1. Verifier-based test-time scaling (TTS) outperforms single-shot prompting and majority voting by filtering out chains with subtle factual or logical errors, and selecting those that are complete and internally consistent.
2. High-quality retrieval (e.g., FAIR) amplifies TTS gains by providing more diverse and complementary evidence, increasing the likelihood that a capable verifier will select the correct reasoning chain.
3. Lightweight BERT-based verifiers (e.g., RoBERTa) can match or even surpass large reasoning-oriented models when trained on high-quality data, as their strong discriminative representations are well-suited for detecting subtle reasoning flaws.

5.4. Analysis of Error Sources in MQA dataset

5.4.1. Answer Granularity Mismatch: Broad vs. Specific Labels

Question: Which film has the director died first, *Crimen A Las Tres* or *The Working Class Goes To Heaven*?

Gold Answer: sports league

Verifier Prediction: National Football League (NFL)

Figure 5.6: Example of abstraction-level mismatch between gold answer and verifier prediction.

One notable source of error contributing to the large gap between the upper bound (ideal verifier) and the upper bound with golden evidences in the MQA dataset is not necessarily a failure in retrieval or reasoning, but rather a mismatch in the expected level of abstraction for the final answer.

For instance, consider the example as shown in Fig 5.6. The question is: *"What is the league of the Carolina Panthers an example of?"* The gold answer here is *sports league*, which is in fact a broad category. The verifier, however, predicts *National Football League (NFL)*, which is a perfectly correct and highly specific instantiation of the gold label. From a human perspective, the prediction is accurate. The NFL is indeed a sports league. However, under the dataset's strict evaluation criteria, the two strings are not considered equivalent, and the prediction is marked as incorrect.

This mismatch illustrates a subtle challenge in open-domain QA evaluation: the model can retrieve high-quality evidence and perform correct reasoning, yet still fail according to the scoring metric because it returns an entity-level answer rather than a category-level one. In this example, the FAIR retrieval pipeline had already surfaced all the relevant details — that the Carolina Panthers are a professional American football team in the NFL — and the reasoning model built a logically sound chain to arrive at NFL. The "error" is purely one of abstraction granularity.

Such cases inflate the gap between the ideal verifier's performance and the golden evidence upper bound, particularly in datasets like MQA that mix factoid and conceptual questions. This suggests that part of closing the gap may require integrating controlled answer type generation or post-processing mechanisms that map specific entities to their broader categories when appropriate.

5.4.2. Misaligned Gold Annotations: Temporal Granularity Errors

Question: What year did the Japanese get to the city where Wang Yue died and the rest of Guangdong province?
Gold Answer: November 5
Verifier Prediction: 1938

Figure 5.7: Example of annotation misalignment in the MQA dataset, where the gold answer format does not match the explicit information requested in the question.

Another contributing factor to the observed gap between the upper bound (ideal verifier) and the golden evidence upper bound in the MQA dataset stems from inconsistencies or misalignments in the gold annotations themselves. A representative example is illustrated in the Figure 5.7. In this case, the question is: *"What year did the Japanese get to the city where Wang Yue died and the rest of Guangdong province?"* The phrasing of the question explicitly requests a *year*. The retrieved evidence and the reasoning chains produced by the verifier correctly identify **1938** as the year in which Japanese forces entered Guangdong province and reached Foshan, the city where Wang Yue later died.

However, the gold answer provided in the dataset is *November 5*, which specifies a month and day rather than a year. From a semantic standpoint, this does not align with the intent of the question and creates an artificial mismatch. The model's answer is factually correct with respect to the question, but it is nevertheless marked as incorrect under the dataset's cover-EM evaluation criterion.

This type of error differs from the "answer abstraction mismatch" discussed in the previous subsection. Here, the failure is not due to the model producing an answer at an incorrect level of generality, but rather because the reference label is itself ill-suited to the question being asked. Such misalignments introduce noise into evaluation results, potentially inflating the perceived gap between system performance and the golden evidence upper bound. In cases like this, no amount of improved retrieval or reasoning will yield a match unless additional answer normalization or gold-label correction is performed.

5.4.3. The Role of Annotator Intent in Evaluation Outcomes

Question: Which team is the highest goal scorer in EPL this season a member of?
Gold Answer: Egypt national football team
Verifier Prediction: Liverpool FC

Figure 5.8: Example of team-affiliation ambiguity between gold answer and verifier prediction.

As illustrated in Figure 5.8, an example of annotation ambiguity arises with the question: *Which team is the highest goal scorer in EPL this season a member of?*

In this case, the gold answer provided here is **Egypt national football team**, whereas the verifier predicts **Liverpool FC**. On the surface, the verifier's prediction aligns closely with what most readers would reasonably infer — the question explicitly refers to the English Premier League (EPL), which is a club competition, so "team" would naturally be interpreted as the player's club. Mohamed Salah, the top scorer in the referenced 2018-19 EPL season, played for Liverpool FC at the time, making the verifier's output contextually correct.

However, the gold label instead refers to Salah's national team. This discrepancy likely arises from a mismatch between the annotator's intent and the question wording. It is possible that the annotator expected the answer to reflect "any team" the player is a member of, with a preference for national representation, or that they were attempting to capture a broader fact about the player's affiliations. In another interpretation, both the club and the national team could be considered valid, but the dataset records only one canonical answer (here, the national team) leading to the club-based prediction being marked as incorrect under strict string matching.

This illustrates a subtle but important source of error in open-domain QA evaluation: even when the reasoning process is sound and the retrieved evidence is directly relevant, the answer may be judged wrong due to unclear or underspecified annotation guidelines. For questions like this, without explicit disambiguation between club and national teams, model predictions can diverge from the gold purely because of differing assumptions about the intended entity type.

6

Conclusion

6.1. Conclusion

This thesis set out to improve the robustness and effectiveness of complex open-domain question answering (ODQA) systems by addressing two critical challenges: the retrieval of relevant evidence, and the selection of reliable reasoning chains at inference time. While recent work has made progress on each of these fronts in isolation, this study explored their integration, examining how retrieval and reasoning can be jointly optimized to produce more accurate and trustworthy answers.

The retrieval improvements were realized through the development of *Frontier Aware Iterative Retrieval* (FAIR), an iterative re-ranking framework that rewards answer-level consistency across document batches. By dynamically promoting documents aligned with the most frequent semantic clusters of generated answers, FAIR increased the diversity and complementarity of retrieved evidence while avoiding premature exclusion of potentially useful documents. Experimental results showed consistent gains in recall@k over SPLADE and SUNAR baselines, with downstream accuracy improvements across multiple reasoning models. These findings confirm that encouraging convergence in answer space during retrieval can surface evidence that might otherwise remain undiscovered.

The second strategy focused on improving reasoning reliability at inference time. Here, the concept of *test-time scaling* (TTS) was applied, generating multiple reasoning chains for each question and using verifier-guided selection to choose the most plausible answer. Both prompt-based verifiers and smaller fine-tuned models were evaluated. Across MuSiQue and 2WikiMultiHopQA datasets, verifier selection consistently

outperformed self-consistency baselines, particularly in scenarios where distractor documents or noisy intermediate reasoning steps would otherwise mislead majority voting. While larger prompt-based verifiers achieved the highest absolute accuracy, smaller trained verifiers offered quite a favorable accuracy-cost trade-off.

Empirical results further revealed that evidence quality remains a significant bottleneck. Even the strongest reasoning strategies could not recover from critical missing information in the retrieved set. Oracle experiments with golden evidence demonstrated substantial headroom, underscoring the importance of continued advances in recall-oriented retrieval methods.

Finally, a grounded theory-inspired qualitative analysis of reasoning chains identified recurring error types and improvement patterns. This analysis highlighted both where improvements occurred and why, from successful handling of distractors to robust multi-hop reasoning in temporal and relational contexts, providing insights that quantitative metrics alone could not reveal.

Taken together, the contributions of this thesis demonstrate that coordinated improvements in retrieval consistency and reasoning selection lead to tangible performance gains in complex ODQA. By explicitly linking retrieval scoring to answer stability and refining chain selection through verifiers, the methods presented here address both the coverage and reliability challenges inherent in multi-hop question answering, while also offering a framework for understanding and diagnosing model behaviour in greater depth.

6.2. Limitations and Assumptions

This section outlines the main limitations and assumptions considered while designing and conducting the experiments.

Reliance on Consistency as a Proxy for Relevance

The retrieval method assumes that high agreement among generated answers is a reliable indicator of evidence quality. This is based on the intuition that when different reasoning chains converge to the same answer, they are likely grounded in relevant and supportive evidence. While this generally holds in practice, there may be cases where the model consistently produces the same incorrect answer, especially if the retrieved pool contains mutually reinforcing but factually wrong information. In such cases, the consistency signal could unintentionally promote misleading documents. Although this effect was not dominant in our experiments, it remains an underlying assumption that could be examined in future work with additional safeguards.

Computational Overhead of Test-Time Scaling

Generating multiple reasoning chains per query and applying verifier selection introduces additional computational cost compared to single-shot inference. This includes both the cost of producing multiple candidate outputs and the processing required by verifiers to evaluate them. While smaller trained verifiers help to reduce the overhead, the approach naturally trades off computation for improved robustness and accuracy. In settings where computational resources are limited, this trade-off can be tuned by adjusting the number of generated chains or selecting more efficient verification models.

Dataset Scope and Generalization

All experiments were conducted on MuSiQue and 2WikiMultiHopQA, two multi-hop QA datasets specifically designed to require compositional reasoning and to be resistant to shortcut strategies. While these datasets provide a rigorous testbed for evaluating the proposed methods, they may not capture the full variety of challenges present in real-world ODQA scenarios, such as domain-specific terminology, noisy user queries, or evolving knowledge bases. Consequently, further validation on a broader range of datasets would help to confirm the generality of the findings.

Hyperparameter Choices

Several design parameters, such as the frontier size ($k = 2$) and the multiplier value applied during consistency scoring, were selected empirically based on preliminary experimentation. Although these choices yielded positive results in our experiments, they were not exhaustively tuned through systematic ablation studies. It is possible that alternative values or adaptive strategies could lead to further improvements, particularly when adapting the approach to different datasets or retrieval-reasoning configurations.

6.3. Future Work

While the proposed methods demonstrate notable improvements in retrieval and reasoning for complex question answering, several promising directions remain for future exploration.

Principled Uncertainty Measures for Retrieval and Re-Ranking

The current feedback mechanism for document re-ranking is based on answer consistency, which serves as a practical proxy for evidence quality. However, more principled uncertainty estimation methods could provide richer and more reliable feed-

back signals. Approaches such as Bayesian inference, entropy-based confidence measures, or calibration-aware scoring could be incorporated to better distinguish between truly uncertain predictions and confidently incorrect ones. Prior work has shown that calibrated uncertainty metrics can better predict answer correctness and retrieval quality in both LLMs and retrieval-augmented settings [41, 50, 20]. Such measures could help mitigate cases where high consistency arises from systematic model bias or homogeneous but misleading evidence.

End-to-End Feedback Integration Across the Pipeline

At present, the verifier and test-time scaling (TTS) stages operate independently from earlier stages such as query decomposition, adaptive retrieval, and re-ranking. An interesting avenue for future work is to integrate these components into an end-to-end feedback loop, where verifier outputs and uncertainty signals are used to adapt earlier stages dynamically. For example, verifier feedback could guide query decomposition strategies, influence the selection or expansion of retrieval frontiers, or adjust re-ranking priorities in real time. This closed-loop optimization could allow the entire pipeline to operate more coherently, continuously refining both retrieval and reasoning stages based on downstream performance. Such end-to-end optimization has been explored in multi-hop QA and retrieval-generation systems, where retrieval and reasoning are trained jointly for better downstream accuracy [94, 24].

Improving the Efficiency of Test-Time Scaling

While TTS improves robustness by exploring multiple reasoning paths, its computational cost grows quickly with the number of candidate solutions. In this work, the number of explored paths was limited to a relatively small set to maintain feasibility. Future work could investigate more efficient search and selection strategies to scale TTS to larger solution spaces without prohibitive cost. Potential approaches include adaptive path pruning, fast explore-exploit algorithms, or lightweight approximations of scoring functions. Although methods such as Monte Carlo Tree Search have been explored in related contexts [95, 13, 97], they often remain computationally expensive; thus, developing efficient yet effective exploration strategies remains an open challenge.

6.4. Disclosure

In accordance with TU Delft's publication guidelines [1], I acknowledge the use of AI-assisted tools, including ChatGPT, to support the refinement of certain sections of

[1] https://www.tudelft.nl/library/actuele-themas/open-publishing/about/policies

this thesis. The visual elements and icons used in figures were sourced from various contributors on www.flaticon.com. All AI-assisted contributions have been critically reviewed and revised to ensure accuracy, clarity, and alignment with the intended meaning.

References

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[2] Akari Asai et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. arXiv: 2310.11511 [cs.CL]. url: https://arxiv.org/abs/2310.11511.

[3] Payal Bajaj et al. *MS MARCO: A Human Generated MAchine Reading COmprehension Dataset*. 2018. arXiv: 1611.09268 [cs.CL]. url: https://arxiv.org/abs/1611.09268.

[4] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1870–1879. doi: 10.18653/v1/P17-1171. url: https://aclanthology.org/P17-1171/.

[5] Jianhao Chen et al. *Do We Truly Need So Many Samples? Multi-LLM Repeated Sampling Efficiently Scales Test-Time Compute*. 2025. arXiv: 2504.00762 [cs.AI]. url: https://arxiv.org/abs/2504.00762.

[6] Zheng Chu et al. *Self-Critique Guided Iterative Reasoning for Multi-hop Question Answering*. 2025. arXiv: 2505.19112 [cs.CL]. url: https://arxiv.org/abs/2505.19112.

[7] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. url: https://arxiv.org/abs/2110.14168.

[8] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. url: https://arxiv.org/abs/2501.12948.

[9] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423. url: https://aclanthology.org/N19-1423.

[10] Yuwei Fang et al. “Hierarchical Graph Network for Multi-hop Question Answering”. In: *CoRR* abs/1911.03631 (2019). arXiv: 1911.03631. url: http://arxiv.org/abs/1911.03631.

[11] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. “SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 2288–2292. isbn: 9781450380379. doi: 10.1145/3404835.3463098. url: https://doi.org/10.1145/3404835.3463098.

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

[13] Zitian Gao et al. *Interpretable Contrastive Monte Carlo Tree Search Reasoning*. 2024. arXiv: 2410.01707 [cs.CL]. url: https://arxiv.org/abs/2410.01707.

[14] Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions*. 2024. arXiv: 2410.12837 [cs.CL]. url: https://arxiv.org/abs/2410.12837.

[15] Lillemor R-M. Hallberg. “The “core category” of grounded theory: Making constant comparisons”. In: *International Journal of Qualitative Studies on Health and Well-being* 1.3 (2006), pp. 141–148. doi: 10.1080/17482620600858399. eprint: https://doi.org/10.1080/17482620600858399. url: https://doi.org/10.1080/17482620600858399.

[16] Xanh Ho et al. “Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6609–6625. doi: 10.18653/v1/2020.coling-main.580. url: https://aclanthology.org/2020.coling-main.580/.

[17] Arian Hosseini et al. *V-STaR: Training Verifiers for Self-Taught Reasoners*. 2024. arXiv: 2402.06457 [cs.LG]. url: https://arxiv.org/abs/2402.06457.

[18] Audrey Huang et al. *Is Best-of-N the Best of Them? Coverage, Scaling, and Optimality in Inference-Time Alignment*. 2025. arXiv: 2503.21878 [cs.AI]. url: https://arxiv.org/abs/2503.21878.

[19] Siyuan Huang et al. *Mirror-Consistency: Harnessing Inconsistency in Majority Voting*. 2024. arXiv: 2410.10857 [cs.CL]. url: https://arxiv.org/abs/2410.10857.

[20] Xinmeng Huang et al. "Uncertainty in Language Models: Assessment through Rank-Calibration". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 284–312. doi: 10.18653/v1/2024.emnlp-main.18. url: https://aclanthology.org/2024.emnlp-main.18/.

[21] Gautier Izacard and Edouard Grave. "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Ed. by Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty. Online: Association for Computational Linguistics, Apr. 2021, pp. 874–880. doi: 10.18653/v1/2021.eacl-main.74. url: https://aclanthology.org/2021.eacl-main.74/.

[22] N. Jardine and C.J. van Rijsbergen. "The use of hierarchic clustering in information retrieval". In: *Information Storage and Retrieval* 7.5 (1971), pp. 217–240. issn: 0020-0271. doi: https://doi.org/10.1016/0020-0271(71)90051-9. url: https://www.sciencedirect.com/science/article/pii/0020027171900519.

[23] Yuelyu Ji et al. *Memory-Aware and Uncertainty-Guided Retrieval for Multi-Hop Question Answering*. 2025. arXiv: 2503.23095 [cs.CL]. url: https://arxiv.org/abs/2503.23095.

[24] Zhengbao Jiang et al. "Retrieval as Attention: End-to-end Learning of Retrieval and Reading within a Single Transformer". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2336–2349. doi: 10.18653/v1/2022.emnlp-main.149. url: https://aclanthology.org/2022.emnlp-main.149/.

[25] Bowen Jin et al. *Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning*. 2025. arXiv: 2503.09516 [cs.CL]. url: https://arxiv.org/abs/2503.09516.

[26] Vladimir Karpukhin et al. “Dense Passage Retrieval for Open-Domain Question Answering”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. doi: 10.18653/v1/2020.emnlp-main.550. url: https://aclanthology.org/2020.emnlp-main.550/.

[27] Muhammad Khalifa et al. *Process Reward Models That Think*. 2025. arXiv: 2504.16828 [cs.LG]. url: https://arxiv.org/abs/2504.16828.

[28] Daniel Khashabi et al. “Looking Beyond the Surface: A Challenge Set for Reading Comprehension over Multiple Sentences”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 252–262. doi: 10.18653/v1/N18-1023. url: https://aclanthology.org/N18-1023/.

[29] Omar Khattab and Matei Zaharia. “ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’20. Virtual Event, China: Association for Computing Machinery, 2020, pp. 39–48. isbn: 9781450380164. doi: 10.1145/3397271.3401075. url: https://doi-org.tudelft.idm.oclc.org/10.1145/3397271.3401075.

[30] Tushar Khot et al. *Decomposed Prompting: A Modular Approach for Solving Complex Tasks*. 2023. arXiv: 2210.02406 [cs.CL]. url: https://arxiv.org/abs/2210.02406.

[31] Hrishikesh Kulkarni et al. “Lexically-Accelerated Dense Retrieval”. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’23. Taipei, Taiwan: Association for Computing Machinery, 2023, pp. 152–162. isbn: 9781450394086. doi: 10.1145/3539618.3591715. url: https://doi-org.tudelft.idm.oclc.org/10.1145/3539618.3591715.

[32] Woosuk Kwon et al. *Efficient Memory Management for Large Language Model Serving with PagedAttention*. 2023. arXiv: 2309.06180 [cs.LG]. url: https://arxiv.org/abs/2309.06180.

[33] *Learning to reason with LLMs — openai.com*. https://openai.com/index/learning-to-reason-with-llms/. [Accessed 04-07-2025].

[34] Hyunji Lee et al. "Generative Multi-hop Retrieval". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 1417–1436. doi: 10.18653/v1/2022.emnlp-main.92. url: https://aclanthology.org/2022.emnlp-main.92/.

[35] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. doi: 10.18653/v1/2020.acl-main.703. url: https://aclanthology.org/2020.acl-main.703/.

[36] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. isbn: 9781713829546.

[37] Hang Li et al. "Improving Query Representations for Dense Retrieval with Pseudo Relevance Feedback: A Reproducibility Study". In: *Advances in Information Retrieval*. Ed. by Matthias Hagen et al. Cham: Springer International Publishing, 2022, pp. 599–612. isbn: 978-3-030-99736-6.

[38] Yifei Li et al. *Making Large Language Models Better Reasoners with Step-Aware Verifier*. 2023. arXiv: 2206.02336 [cs.CL]. url: https://arxiv.org/abs/2206.02336.

[39] Zijian Li et al. *Graph Neural Network Enhanced Retrieval for Question Answering of LLMs*. 2024. arXiv: 2406.06572 [cs.CL]. url: https://arxiv.org/abs/2406.06572.

[40] Hunter Lightman et al. *Let's Verify Step by Step*. 2023. arXiv: 2305.20050 [cs.LG]. url: https://arxiv.org/abs/2305.20050.

[41] Linyu Liu et al. *Uncertainty Estimation and Quantification for LLMs: A Simple Supervised Approach*. 2024. arXiv: 2404.15993 [cs.LG]. url: https://arxiv.org/abs/2404.15993.

[42] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. url: https://arxiv.org/abs/1907.11692.

[43] Xueguang Ma et al. *A Replication Study of Dense Passage Retriever*. 2021. arXiv: 2104.05740 [cs.CL]. url: https://arxiv.org/abs/2104.05740.

[44] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. "Adaptive Re-Ranking with a Corpus Graph". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. CIKM '22. Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 1491–1500. isbn: 9781450392365. doi: 10.1145/3511808.3557231. url: https://doi-org.tudelft.idm.oclc.org/10.1145/3511808.3557231.

[45] Aman Madaan et al. "SELF-REFINE: iterative refinement with self-feedback". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS '23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[46] Sewon Min et al. "Compositional Questions Do Not Necessitate Multi-hop Reasoning". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4249–4257. doi: 10.18653/v1/P19-1416. url: https://aclanthology.org/P19-1416/.

[47] Minh-Vuong Nguyen et al. *Direct Evaluation of Chain-of-Thought in Multi-hop Reasoning with Knowledge Graphs*. 2024. arXiv: 2402.11199 [cs.CL]. url: https://arxiv.org/abs/2402.11199.

[48] Yixin Nie, Haonan Chen, and Mohit Bansal. *Combining Fact Extraction and Verification with Neural Semantic Matching Networks*. 2018. arXiv: 1811.07039 [cs.CL]. url: https://arxiv.org/abs/1811.07039.

[49] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. *Document Ranking with a Pre-trained Sequence-to-Sequence Model*. 2020. arXiv: 2003.06713 [cs.IR]. url: https://arxiv.org/abs/2003.06713.

[50] Laura Perez-Beltrachini and Mirella Lapata. *Uncertainty Quantification in Retrieval Augmented Question Answering*. 2025. arXiv: 2502.18108 [cs.CL]. url: https://arxiv.org/abs/2502.18108.

[51] Ofir Press et al. “Measuring and Narrowing the Compositionality Gap in Language Models”. In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5687–5711. doi: 10.18653/v1/2023.findings-emnlp.378. url: https://aclanthology.org/2023.findings-emnlp.378/.

[52] Yuxiao Qu et al. “Recursive Introspection: Teaching Language Model Agents How to Self-Improve”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 55249–55285. url: https://proceedings.neurips.cc/paper_files/paper/2024/file/639d992f819c2b40387d4d5170b8ffd7-Paper-Conference.pdf.

[53] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. url: https://arxiv.org/abs/1910.10683.

[54] Pranav Rajpurkar et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by Jian Su, Kevin Duh, and Xavier Carreras. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. doi: 10.18653/v1/D16-1264. url: https://aclanthology.org/D16-1264/.

[55] Gowtham Ramesh, Makesh Sreedhar, and Junjie Hu. *Single Sequence Prediction over Reasoning Graphs for Multi-hop QA*. 2023. arXiv: 2307.00335 [cs.CL]. url: https://arxiv.org/abs/2307.00335.

[56] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. *Quam: Adaptive Retrieval through Query Affinity Modelling*. 2024. arXiv: 2410.20286 [cs.IR]. url: https://arxiv.org/abs/2410.20286.

[57] Benjamin Reichman and Larry Heck. “Dense Passage Retrieval: Is it Retrieving?” In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 13540–13553. doi: 10.18653/v1/2024.findings-emnlp.791. url: https://aclanthology.org/2024.findings-emnlp.791/.

[58] Stephen Robertson and Hugo Zaragoza. “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Found. Trends Inf. Retr.* 3.4 (Apr. 2009), pp. 333–389. issn: 1554-0669. doi: 10.1561/1500000019. url: https://doi-org.tudelft.idm.oclc.org/10.1561/1500000019.

[59] Stephen E Robertson et al. “Okapi at TREC-3”. In: *Nist Special Publication Sp* 109 (1995), p. 109.

[60] Corby Rosset et al. *Knowledge-Aware Language Model Pretraining*. 2021. arXiv: 2007.00655 [cs.CL]. url: https://arxiv.org/abs/2007.00655.

[61] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. doi: 10.1109/TNN.2008.2005605.

[62] Shuaijie She et al. *R-PRM: Reasoning-Driven Process Reward Modeling*. 2025. arXiv: 2503.21295 [cs.CL]. url: https://arxiv.org/abs/2503.21295.

[63] Avi Singh et al. *Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models*. 2024. arXiv: 2312.06585 [cs.LG]. url: https://arxiv.org/abs/2312.06585.

[64] Charlie Snell et al. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. 2024. arXiv: 2408.03314 [cs.LG]. url: https://arxiv.org/abs/2408.03314.

[65] Karen Sparck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[66] Alon Talmor and Jonathan Berant. “The Web as a Knowledge-Base for Answering Complex Questions”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 641–651. doi: 10.18653/v1/N18-1059. url: https://aclanthology.org/N18-1059/.

[67] James Thorne et al. “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)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 809–819. doi: 10.18653/v1/N18-1074. url: https://aclanthology.org/N18-1074/.

[68] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. url: https://arxiv.org/abs/2302.13971.

[69] Harsh Trivedi et al. "♫ MuSiQue: Multihop Questions via Single-hop Question Composition". In: *Transactions of the Association for Computational Linguistics* 10 (May 2022), pp. 539–554. issn: 2307-387X. doi: 10.1162/tacl_a_00475. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00475/2020694/tacl_a_00475.pdf. url: https://doi.org/10.1162/tacl%5C_a%5C_00475.

[70] Harsh Trivedi et al. "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 10014–10037. doi: 10.18653/v1/2023.acl-long.557. url: https://aclanthology.org/2023.acl-long.557/.

[71] Jonathan Uesato et al. *Solving math word problems with process- and outcome-based feedback*. 2022. arXiv: 2211.14275 [cs.LG]. url: https://arxiv.org/abs/2211.14275.

[72] Venktesh V, Mandeep Rathee, and Avishek Anand. "SUNAR: Semantic Uncertainty based Neighborhood Aware Retrieval for Complex QA". In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Luis Chiruzzo, Alan Ritter, and Lu Wang. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 5818–5835. isbn: 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.300. url: https://aclanthology.org/2025.naacl-long.300/.

[73] Robert Vacareanu et al. *General Purpose Verification for Chain of Thought Prompting*. 2024. arXiv: 2405.00204 [cs.CL]. url: https://arxiv.org/abs/2405.00204.

[74] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. url: https://arxiv.org/abs/1710.10903.

[75] Ellen M. Voorhees and Dawn M. Tice. "Building a question answering test collection". In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '00. Athens, Greece: Association for Computing Machinery, 2000, pp. 200–207. isbn: 1581132263. doi: 10.1145/345508.345577. url: https://doi-org.tudelft.idm.oclc.org/10.1145/345508.345577.

[76] Ellen M. Voorhees and Dawn M. Tice. “The TREC-8 Question Answering Track”. In: *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC‘00)*. Ed. by M. Gavrilidou et al. Athens, Greece: European Language Resources Association (ELRA), May 2000. url: https://aclanthology.org/L00-1018/.

[77] Liang Wang et al. *SimLM: Pre-training with Representation Bottleneck for Dense Passage Retrieval*. 2023. arXiv: 2207.02578 [cs.IR]. url: https://arxiv.org/abs/2207.02578.

[78] Peiyi Wang et al. “Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations”. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 9426–9439. doi: 10.18653/v1/2024.acl-long.510. url: https://aclanthology.org/2024.acl-long.510/.

[79] Xuezhi Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. arXiv: 2203.11171 [cs.CL]. url: https://arxiv.org/abs/2203.11171.

[80] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. “Constructing Datasets for Multi-hop Reading Comprehension Across Documents”. In: *CoRR* abs/1710.06481 (2017). arXiv: 1710.06481. url: http://arxiv.org/abs/1710.06481.

[81] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. “Constructing Datasets for Multi-hop Reading Comprehension Across Documents”. In: *Transactions of the Association for Computational Linguistics* 6 (2018). Ed. by Lillian Lee et al., pp. 287–302. doi: 10.1162/tacl_a_00021. url: https://aclanthology.org/Q18-1021/.

[82] Robert E. White and Karyn Cooper. “Grounded Theory”. In: *Qualitative Research in the Post-Modern Era: Critical Approaches and Selected Methodologies*. Cham: Springer International Publishing, 2022, pp. 339–385. isbn: 978-3-030-85124-8. doi: 10.1007/978-3-030-85124-8_9. url: https://doi.org/10.1007/978-3-030-85124-8_9.

[83] Yangzhen Wu et al. *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for Problem-Solving with Language Models*. 2025. arXiv: 2408.00724 [cs.AI]. url: https://arxiv.org/abs/2408.00724.

[84] Yuxi Xie et al. *Self-Evaluation Guided Beam Search for Reasoning*. 2023. arXiv: 2305.00633 [cs.CL]. url: https://arxiv.org/abs/2305.00633.

[85] Zhihui Xie et al. *Calibrating Reasoning in Language Models with Internal Consistency*. 2024. arXiv: 2405.18711 [cs.AI]. url: https://arxiv.org/abs/2405.18711.

[86] Lee Xiong et al. “Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval”. In: *CoRR* abs/2007.00808 (2020). arXiv: 2007.00808. url: https://arxiv.org/abs/2007.00808.

[87] Wenhan Xiong et al. *Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval*. 2021. arXiv: 2009.12756 [cs.CL]. url: https://arxiv.org/abs/2009.12756.

[88] Kehan Xu et al. “Crp-rag: A retrieval-augmented generation framework for supporting complex logical reasoning and knowledge planning”. In: *Electronics* 14.1 (2024), p. 47.

[89] Zhilin Yang et al. “HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff et al. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2369–2380. doi: 10.18653/v1/D18-1259. url: https://aclanthology.org/D18-1259/.

[90] Michihiro Yasunaga et al. “QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kristina Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 535–546. doi: 10.18653/v1/2021.naacl-main.45. url: https://aclanthology.org/2021.naacl-main.45/.

[91] Semih Yavuz et al. *Modeling Multi-hop Question Answering as Single Sequence Prediction*. 2022. arXiv: 2205.09226 [cs.CL]. url: https://arxiv.org/abs/2205.09226.

[92] Eric Zelikman et al. *Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking*. 2024. arXiv: 2403.09629 [cs.CL]. url: https://arxiv.org/abs/2403.09629.

[93] Dongxu Zhang et al. *ASCoT: An Adaptive Self-Correction Chain-of-Thought Method for Late-Stage Fragility in LLMs*. 2025. arXiv: 2508.05282 [cs.CL]. url: https://arxiv.org/abs/2508.05282.

[94] Jiahao Zhang et al. “End-to-End Beam Retrieval for Multi-Hop Question Answering”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 1718–1731. doi: 10.18653/v1/2024.naacl-long.96. url: https://aclanthology.org/2024.naacl-long.96/.

[95] Kexun Zhang et al. “Scaling LLM Inference Efficiently with Optimized Sample Compute Allocation”. In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Luis Chiruzzo, Alan Ritter, and Lu Wang. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 7959–7973. isbn: 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.404. url: https://aclanthology.org/2025.naacl-long.404/.

[96] Lunjun Zhang et al. *Generative Verifiers: Reward Modeling as Next-Token Prediction*. 2025. arXiv: 2408.15240 [cs.LG]. url: https://arxiv.org/abs/2408.15240.

[97] Yize Zhang et al. “ARise: Towards Knowledge-Augmented Reasoning via Risk-Adaptive Search”. In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Wanxiang Che et al. Vienna, Austria: Association for Computational Linguistics, July 2025, pp. 10978–10995. isbn: 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.538. url: https://aclanthology.org/2025.acl-long.538/.

[98] Jian Zhao et al. *GenPRM: Scaling Test-Time Compute of Process Reward Models via Generative Reasoning*. 2025. arXiv: 2504.00891 [cs.CL]. url: https://arxiv.org/abs/2504.00891.

A

Appendix

A.1. Datasets

A.1.1. WQA Dataset

Table A.1: Representative Question Types and Examples from the WQA Dataset

Question
People & Relations
Who is Wisigard's father-in-law?
Who is the maternal grandfather of Antiochus X Eusebes?
Attributes & Facts
What is the date of death of Alexander Baring, 4th Baron Ashburton's father?
What is the place of birth of the performer of the song *If You Want To Be My Woman*?
Temporal (Dates/Times)
When did John V, Prince of Anhalt-Zerbst's father die?
When was Britannicus's father born?
Locations & Places
Where did the director of the film *Vestire Gli Ignudi* die?
Where was the place of death of the director of the film *Mole Men Against the Son of Hercules*?

Question
Comparisons & Selections
Which film has the director died earlier, *Condemned Women* or *Faces in the Dark*?
Which country is Audofleda's husband from?
Causal (Why)
Why did Charis Wilson's husband die?
Why did the director of the film *Normande* die?
Boolean
Are both *Kaufland* and *Otrag* located in the same country?
Are the directors of both films *Der Blindgänger* and *Hotel Desire* from the same country?

A.1.2. MQA Dataset

Table A.2: Representative Question Types and Examples from the MQA Dataset

Question
People & Relations
Who did the screenwriter for *Good Will Hunting* play in *Dazed and Confused*?
Who played the girlfriend of Alex P. Keaton's actor on *Family Ties* in *Back to the Future*?
Who sings *Home Alone Tonight* with the singer of *You Can Crash My Party Anytime*?
Who wrote "Turn Me On" by the performer of "Happy Pills"?
Attributes & Facts
What is the record label for the band which performed *Pythons*?
The group representing at least 70% of Jews worldwide and Italians may be genetically similar due to what two factors?
Selections
Which county does Lloyd Dane's birthplace belong to?
Numerical (Amounts)
How much did the Black Death reduce the population of the region offered aid by the Marshall Plan?

Question
How far is Hod Lisenbee's place of death in TN from Nashville? How far from Nashville in miles is the place of death of Felix Ives Batson?
Duration/Extent
How long lasting was the pact between Kravchuk and the person organizers wanted to arrest?

A.2. Prompts

A.2.1. Meta-Reasoner Prompt

Meta-Reasoner Prompt

Instruction: Follow the given examples and, given the question and context, reasoning path, think step by step, extract key segments from given evidence relevant to the question and give rationale by forming your own reasoning path preceded by `[Answer]:`, and output the final answer for the question using information from the given evidences and give a concise precise answer preceded by `[Final Answer]:`.

Exemplars: {}

Given the above examples and
Existing reasoner path: {Reasoning path from SUNAR},
the evidence: {top-l evidences across sub-questions}
and use the most relevant information for the question from the most relevant evidence from the given Evidence; and form your own correct reasoning path to derive the answer thinking step by step preceded by `[Answer]:` and subsequently give the final answer as shown in the above examples preceded by `[Final Answer]:` for the Question: {Test Question}.

Figure A.1: Instruction template for the meta-reasoner.

A.2.2. Iterative Retrieval Prompt

MusiqueQA Prompt

Instruction:`Follow the given examples and Given the question determine if followup questions are needed and decompose the original question.`

Exemplars :
Question: When does monsoon season end in the state the area code 575 is located?
Are follow up questions needed here: Yes.
Follow up: Which state is the area code 575 located in?
Intermediate Answer: The area code 575 is located in New Mexico.
Follow up: When does monsoon season end in New Mexico?
Intermediate Answer: Monsoon season in New Mexico typically ends in mid-September.
[Final Answer]: mid-September.

Question: What is the current official currency in the country where Ineabelle Diaz is a citizen?
Are follow up questions needed here: Yes.
Follow up: Which country is Ineabelle Diaz a citizen of?
Intermediate Answer: Ineabelle Diaz is from Peurto Rico, which is in the United States of America.
Follow up: What is the current official currency in the United States of America?
Intermediate Answer: The current official currency in the United States is the United States dollar. [Final Answer]: United States dollar.

Question: Where was the person who founded the American Institute of Public Opinion in 1935 born? Are follow up questions needed here: Yes. Follow up: Who founded the American Institute of Public Opinion in 1935? Intermediate Answer: George Gallup. Follow up: Where was George Gallup born? Intermediate Answer: George Gallup was born in Jefferson, Iowa.
[Final Answer]: Jefferson.

Question: What is the sports team the person played for who scored the first touchdown in Superbowl 1?
Are follow up questions needed here: Yes.
Follow up: Which player scored the first touchdown in Superbowl 1?
Intermediate Answer: Max McGee.
Follow up: Which sports team did Max McGee play for?
Intermediate Answer: Max McGee played for the Green Bay Packers.
[Final Answer]: Green Bay Packers.

Question: The birth country of Jayantha Ketagoda left the British Empire when?
Are follow up questions needed here: Yes.
Follow up: What is the birth country of Jayantha Ketagoda?
Intermediate Answer: Sri Lanka.
Follow up: When did Sri Lanka leave the British Empire?
Intermediate Answer: Sri Lanka left the British Empire on February 4, 1948.
[Final Answer]: February 4, 1948

Input : Based on above examples, given the Question: determine, Are followup questions needed here ?

Figure A.2: Example of In-context learning for MusiqueQA (MQA) and 2WikiMultiHopQA (WQA) through `SELF-ASK` based prompting of LLMs