

Scaling Laws for Compound AI Systems

More Compute, More Problems?
Rethinking Scaling Laws for Retrieval-Augmented
Generation

Pratham Johari

Delft University of Technology

Scaling Laws for Compound AI Systems

More Compute, More Problems?
Rethinking Scaling Laws for
Retrieval-Augmented Generation

MASTER THESIS

submitted in partial fulfilment of the
requirements for the degree of

MASTER OF SCIENCE

in

DATA SCIENCE AND ARTIFICIAL INTELLIGENCE TECHNOLOGY

by

Pratham Johari

born in Udaipur, India

Committee Chair and Supervisor:	Dr. Avishek Anand
Daily Supervisor:	Dr. Mandeep Rathee
First Core Member:	Dr. Julian Urbano
Project Duration:	September 2025 – June 2026
Faculty:	Faculty of EEMCS, TU Delft



Web Information Systems Group
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Preface

This report is the culmination of my studies, marking the completion of my Master's programme in Data Science and Artificial Intelligence Technology at Delft University of Technology. My first machine learning project was back in the 12th grade, where I learned how to perform object detection on my webcam using a very basic TensorFlow setup. That single project started me down a winding path, eventually leading me to this thesis, where I seek to understand Compound AI Systems and uncover where their true performance capabilities lie.

This thesis has been, without a doubt, the most challenging project I have ever undertaken. Juggling hundreds of moving parts has been both exhilarating and exhausting. Even so, I have thoroughly enjoyed the work, all the troubleshooting, debugging code, and analysing results. Every single step held something interesting. I have given this project my all over the past nine months, and I hope whoever reads this finds insights they might not have expected.

I am deeply grateful to Dr. Avishek Anand and Dr. Mandeep Rathee for giving me this opportunity and for their guidance, support, and insightful questions that continually challenged me. I learned so much throughout this process, and this work simply would not have been possible without their support at every step.

I want to thank everyone who helped me reach this point: my parents, who supported me even when things got tough, and my friends, who were there for me even when my focus was elsewhere. I also want to thank Delft for giving me the space to grow up and the countless energy drinks that kept me fueled through the late nights. Finally, I want to express a deep gratitude to myself for pushing through and seeing potential in myself that I sometimes couldn't.

I hope you enjoy reading this report.

Pratham Johari
Delft, June 2026

Scaling Laws for Compound AI Systems

Abstract

Retrieval-Augmented Generation (RAG) extends Large Language Models with external knowledge retrieval, yet the scaling behaviour of inference-time compute in multi-stage RAG pipelines remains poorly characterised. This thesis provides a systematic empirical investigation of test-time compute scaling along three axes: *retrieval depth* (expanding the candidate pool size c presented to a neural reranker), *query breadth* (generating n LLM-produced query reformulations and fusing their results), and *adaptive budget allocation* (dynamically distributing cross-encoder compute via ORE and ReformIR). Experiments span three benchmark datasets (MS MARCO, TriviaQA/DPR-w100, 2WikiMultiHopQA), two first-stage retrievers (BM25, TCT-ColBERT), six fusion algorithms, and two pipeline orderings (merge-then-rerank vs. rerank-then-merge).

Our findings establish six empirical principles for test-time compute allocation in compound retrieval systems. First, depth scaling yields monotonic gains only when the reranker is architecturally robust to noise; the generative reranker MonoT5 degrades at large c while the distilled cross-encoder MiniLM-L6 scales reliably. Second, query breadth scaling requires a continuous embedding space—dense retrievers enable substantially larger fusion gains than sparse retrievers because semantically distinct rewrites activate genuinely different document regions. Third, the optimal pipeline ordering depends on the reranker’s domain calibration: rerank-then-merge excels in-domain, while voting-based fusion dominates out-of-domain. Fourth, adaptive and fusion-based scaling are complementary rather than competing. Fifth, a persistent 15–17% relative Recall@50 Oracle gap reveals that practical fusion algorithms, not candidate generation, constitute the performance bottleneck. Sixth, retrieval gains transfer only partially to downstream generation quality, with volatile LLM-as-Judge trajectories cautioning against assuming proportional end-to-end improvements.

These results demonstrate that, unlike the monotonic scaling laws governing language model pre-training, retrieval scaling is non-monotonic and conditional—effective test-time compute allocation in Compound AI Systems requires domain-aware orchestration across the full retriever–reranker–fusion–reader pipeline.

Contents

Preface	i
Abstract	ii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Gap	2
1.3 Research Questions	3
1.4 Contributions	4
1.5 Thesis Outline	4
2 Related Work	5
2.1 Foundations of Test-Time Scaling	5
2.2 RAG as a Compute-Optimal Reasoning System	6
2.2.1 Retriever Scaling: Expanding the Search Frontier	6
2.2.2 Reranker Scaling: The Verifier / Process Reward Model	6
2.3 Scaling Retrieval Strategies	7
2.3.1 Parallel Scaling: Width and Diversity	7
2.3.2 Sequential Scaling: Depth and Iteration	8
2.4 Adaptive Retrieval and Reformulation Strategies	9
2.4.1 ORE: Adaptive Re-Ranking with a Corpus Graph	9
2.4.2 ReformIR: Adaptive Retrieval via Query Reformulation	10
2.4.3 Comparison and Thesis Relevance	10
2.5 Inference Scaling Laws for RAG	11
2.6 Evaluation Datasets and Benchmarks	11
2.6.1 2WikiMultiHopQA and Multi-Hop Benchmarks	11
2.6.2 Additional Widely-Used Benchmarks	12
2.7 Summary	12
3 Implementation Details and Experimental Setup	13
3.1 Experimental Setup and Design	13
3.2 Hardware and Implementation Details	13
3.2.1 First-Stage Retrievers	14
3.2.2 Second-Stage Rerankers	14
3.3 Evaluation Datasets	14
3.3.1 2WikiMultiHopQA (2WikiMultiHopQA)	15
3.3.2 MSMARCO-Passage	15
3.3.3 DPR-w100 (TriviaQA)	15
3.3.4 Sampling and Determinism	16
3.4 Impact of Retrieval Depth (Scaling c)	16

3.5	Impact of Retrieval Breadth (Scaling n)	17
3.5.1	Query Rewriting Configuration	18
3.5.2	Fusion Strategies	19
3.5.3	Dense Retrieval Fusion	20
3.5.4	Rerank-then-Merge Paradigm	20
3.5.5	Noise Analysis and Oracle Upper Bounds	21
3.6	Compute Proxy Definition	22
3.7	Adaptive Retrieval Methods	22
3.7.1	ORE Adaptive Cutoff Sweep	22
3.7.2	ReformIR Cutoff Sweep	24
3.8	End-to-End RAG Generation	25
3.8.1	Generation Prompt and Decoding	25
3.8.2	Answer Evaluation Metrics	26
3.9	Summary of Experimental Design	27
4	Results	28
4.1	Impact of Initial Retrieval Cutoff Depth	28
4.2	Impact of Query Rewrite Breadth	30
4.2.1	Impact of Query Rewrite Breadth on Sparse Retrieval	30
4.2.2	Impact of Query Rewrite Breadth on Dense Retrieval	32
4.2.3	Impact of Query Rewrite Breadth on Rerank-then-Merge	34
4.2.4	Noise Analysis and Upper Bounds	36
4.3	Comparing Results Across Scaling Strategies	38
4.3.1	Sparse vs Dense Retrieval Dynamics with respect to Query Rewrites	38
4.3.2	RTM vs MTR Retrieval Dynamics	42
4.4	Adaptive Retrieval Methods	46
4.4.1	ORE	46
4.4.2	ReformIR	48
4.4.3	RAG Pipeline Performance	50
5	Conclusion	53
5.1	Synthesis of Findings	53
5.2	Limitations	55
5.3	Future Work	55
5.4	Closing Remarks	56
	References	58
A	Experimental Setup Appendix	63
A.1	Experimental run Details	64
A.2	Query Rewriting Prompts and Examples	65
A.2.1	LLM Rewriting System Prompt	65
A.2.2	BestRewrite Selection Prompt	66
A.2.3	Query Rewrite Examples	66
A.3	LLM-as-a-Judge Few-Shot Examples	66
A.4	LLM-as-a-Judge Failure Cases	68

B	MonoT5 Reranker Results	69
B.1	Merge-then-Rerank with MonoT5 (Experiment 2)	70
B.2	Rerank-then-Merge with MonoT5 (Experiment 2b)	71

List of Figures

3.1	Schematic illustration of the two pipelines compared in the depth scaling experiment. The top panel depicts the Sparse and Dense baseline, where the top- c documents retrieved by the sparse/dense retriever are evaluated directly. The bottom panel depicts the two-stage pipeline, where the initial candidate set is reranked before evaluation.	16
3.3	Experimental workflow for query broadening via parallel rewriting. The original query q is passed to an LLM rewriter, which generates n semantically diverse variants. Each rewritten query is independently issued to BM25. The resulting n ranked lists are aggregated by a fusion algorithm into a single candidate list, which is subsequently reranked by MiniLM-L6 to produce the final ranked context. This architecture corresponds to the merge-then-rerank paradigm illustrated in the top panel of Figure 3.2.	18
3.2	Comparison of fusion architectures for query broadening. The top panel illustrates the merge-then-rerank (MTR) paradigm used in the sparse and dense breadth experiments (Sections 3.5 and 3.5.3). The bottom panel illustrates the rerank-then-merge (RTM) paradigm (Section 3.5.4).	18
4.1	Effect of initial retrieval cutoff depth c on Recall@10 (left column) and nDCG@10 (right column) across all three evaluation corpora (Section 3.4, Table 3.3). Each subfigure compares six pipeline configurations: BM25 and Dense (TCT-ColBERT) baselines, each optionally followed by MiniLM-L6 or MonoT5 reranking. The x -axis is log-scaled, reflecting the logarithmic sweep $c \in \{10, \dots, 1000\}$. Dashed horizontal lines denote single-query baselines (constant across c).	30
4.2	Effect of query rewrite breadth n on retrieval quality under the merge-then-rerank (MTR) paradigm with BM25 first-stage retrieval and MiniLM-L6 reranking (Section 3.5, Table 3.4). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Rows correspond to 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom). Six fusion methods are compared; see Section 3.5.2 for definitions. The dashed horizontal lines denote the BM25 and BM25+MiniLM-L6 baselines (constant across n).	32
4.3	Effect of query rewrite breadth n on dense retrieval (TCT-ColBERT) with MiniLM-L6 reranking across 2WikiMultihopQA (top row) and MS MARCO (bottom row) (Section 3.5.3, Table 3.5). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Four fusion methods are evaluated: Borda, CombMAX, RRF, and Exp4Fuse. TriviaQA is omitted due to computational constraints.	34
4.4	Effect of query rewrite breadth n on retrieval quality under the rerank-then-merge (RTM) paradigm with BM25 first-stage retrieval and MiniLM-L6 reranking (Section 3.5.4, Table 3.6). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Rows correspond to 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom). Six fusion methods are compared; see Section 3.5.2 for definitions.	36

- 4.5 Oracle upper bounds and noise analysis for query rewrite fusion across 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom), using BM25 retrieval with MiniLM-L6 reranking (Section 3.5.5, Table 3.7). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Bold labelled curves show three analytical constructs: Oracle (best single rewrite per query), Dedup Oracle (merged deduplicated pool without cutoff truncation), and BestRewrite (GPT-4o-mini-selected rewrite without retrieval feedback). The shaded grey region indicates the min-max range of the six practical fusion methods from Figure 4.2, with the dashed line showing their mean. The secondary y -axis (top, red) shows the average number of unique documents in the Dedup Oracle candidate pool at each rewrite depth n 38
- 4.6 Recall@50 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking on MS MARCO and 2WikiMultihopQA, across four fusion methods. The upper subfigure shows the relative change from the reranked baseline as a function of the number of rewrites n . The lower subfigure shows peak absolute R@50 values and the n at which they are achieved. 40
- 4.7 nDCG@10 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking on MS MARCO and 2WikiMultihopQA. The upper subfigure shows the relative change from the baseline for varying rewrite counts n . The lower subfigure displays peak absolute nDCG@10 values, including the number of rewrites n used. 41
- 4.8 Recall@10 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking. The upper subfigure illustrates the relative change in R@10 as a function of the rewrite count n . The lower subfigure shows the peak absolute R@10 values achieved for each retrieval-fusion combination. 42
- 4.9 Recall@50 results comparing MTR (merge-then-rerank, Section 4.2.1) and RTM (rerank-then-merge, Section 4.2.3) pipeline orderings with BM25 retrieval and MiniLM-L6 reranking across all three corpora. The upper subfigure shows the relative change from the BM25+RR baseline as a function of rewrite count n . Solid lines denote MTR; dashed lines denote RTM. The lower subfigure shows peak absolute R@50 values and the n at which each was achieved. RTM score-based methods (DedupCutoff, CombMAX) achieve the largest gains on MS MARCO, while MTR voting methods (Borda) are more stable on TriviaQA and 2WikiMultihopQA. 44
- 4.10 nDCG@10 results comparing MTR and RTM pipeline orderings across all three corpora. On MS MARCO (left), RTM score-based methods (DedupCutoff, CombMAX) preserve nDCG@10 (+3–5%), while RTM voting methods degrade by –5% to –12%. On TriviaQA (middle), all RTM methods degrade nDCG@10, with Exp4Fuse reaching –14%. On 2WikiMultihopQA (right), RTM DedupCutoff achieves the best nDCG@10 gains (+6–7%). Layout as in Figure 4.9. 45
- 4.11 Recall@10 results comparing MTR and RTM pipeline orderings with BM25 retrieval and MiniLM-L6 reranking. On MS MARCO (left), DedupCutoff and CombMAX achieve marginal RTM gains (+2%), while Exp4Fuse(RTM) drops –9% to –10%. On TriviaQA (middle), Borda(RTM) reaches –19% and CombMAX(RTM) –15%. On 2WikiMultihopQA (right), RTM DedupCutoff gains +9% R@10, outperforming all MTR methods. Layout as in Figure 4.9. 46

- 4.12 ORE adaptive retrieval performance as a function of initial cutoff depth c across all three evaluation corpora (Section 3.7.1, Table 3.8). Columns show Recall@10 (left) and nDCG@10 (right). Two cross-encoder budget modes are compared: Linear (proportional to c , solid lines) and Fixed ($ce=7$, dashed lines). The BM25 baseline and BM25+MiniLM-L6 reranker baseline are shown for reference. On MS MARCO, ORE uses a fully pre-computed LAFF graph; on TriviaQA and 2WikiMultihopQA, LAFF is computed on-the-fly. 48
- 4.13 ReformIR adaptive retrieval performance as a function of initial cutoff depth c across all three evaluation corpora (Section 3.7.2, Table 3.9). Columns show Recall@10 (left) and nDCG@10 (right). Two cross-encoder budget modes are compared: Linear [L] (solid lines, full cross-encoder coverage) and Fixed [F] (dashed lines, $ce=7$ cap, forcing the λ -regression model to estimate scores for unscored candidates at large c). The BM25 baseline and BM25+MiniLM-L6 reranker baseline are shown for reference. 50
- 4.14 Effect of retrieval cutoff c on RAG generation quality across Wiki (top row) and TriviaQA (bottom row). Metrics shown are LLM-as-a-Judge accuracy (left column), F1 Score (middle column), and Exact Match (right column). The without-reranker baseline is constant across all c because the reader always receives the same top-5 BM25 passages. 52
- B.1 Effect of query rewrite breadth n on retrieval quality under the merge-then-rerank (MTR) paradigm with BM25 first-stage retrieval and **MonoT5** reranking. Layout matches Figure 4.2. Rows: Wiki-MuSiQue (top), TriviaQA (middle), MS MARCO (bottom). Columns: R@10 (left), nDCG@10 (middle), R@50 (right). 70
- B.2 Effect of query rewrite breadth n on retrieval quality under the rerank-then-merge (RTM) paradigm with BM25 first-stage retrieval and **MonoT5** reranking. Layout matches Figure 4.4. Rows: Wiki-MuSiQue (top), TriviaQA (middle), MS MARCO (bottom). Columns: R@10 (left), nDCG@10 (middle), R@50 (right). 71

List of Tables

3.1	Summary of all models used across the experimental pipeline.	15
3.2	Summary statistics of evaluation datasets.	15
3.3	Depth scaling configuration. Six pipelines are evaluated: BM25, BM25+MonoT5, BM25+MiniLM-L6, Dense, Dense+MonoT5, Dense+MiniLM-L6.	17
3.4	Sparse retrieval breadth configuration.	20
3.5	Dense retrieval breadth configuration.	21
3.6	Rerank-then-merge (RTM) breadth configuration.	21
3.7	Oracle configurations used for rewrite-fusion analysis.	22
3.8	ORE scaling configurations for varying retrieval budgets (batch size = 16). Coverage denotes the proportion of the retrieval budget receiving real cross-encoder scores; values exceeding 100% arise because the total real calls ($ce \times 16$) can exceed c due to ceiling rounding of the batch count.	24
3.9	ReformIR scaling configurations for varying retrieval budgets (batch size = 16). Coverage denotes the proportion of the retrieval budget receiving real cross-encoder scores; values exceeding 100% arise because the total real calls ($ce \times 16$) can exceed c due to ceiling rounding.	25
3.10	End-to-end RAG generation configuration.	27
A.1	Experiment run provenance. Each final result is traced to its execution platform via the <code>data_path</code> field recorded in the run metadata.	64
A.2	Examples of LLM-generated query rewrites across all three evaluation datasets. Each original query is rewritten into retrieval-oriented formulations by GPT-4o-mini ($\tau = 0.7$); four representative rewrites are shown per query (out of 30 generated).	67
A.3	Few-shot examples used in the LLM-as-a-Judge prompt. Each example pairs a query, gold answer, and system answer with the expected score.	67
A.4	Representative LLM-as-Judge failure cases from the TriviaQA evaluation. Each row shows the query, gold reference answer, system prediction, and the judge’s score (1–9 scale). Scores highlighted in bold indicate clear misjudgements.	68

Introduction

1.1. Background and Motivation

The field of Artificial Intelligence is undergoing a gradual transition from monolithic model training toward the engineering of *Compound AI Systems* [40]. For the past decade, the dominant strategy for advancing AI capabilities has been governed by training-time scaling laws, where increasingly massive datasets and parameter counts are utilised to minimise perplexity [13, 8]. However, these monolithic Transformers operate as static predictors of the next token—“System 1” agents optimised for rapid, heuristic-driven responses. Consequently, they suffer from “frozen weights,” which limit their ability to access up-to-date information and result in restricted reasoning depths that can lead to hallucinations when forced to extrapolate beyond their training distribution [41].

In contrast, compound architectures treat the Large Language Model (LLM) not as the sole cognitive engine, but as a node within a broader topology [40]. This architectural shift is driven by three primary principles:

- **Flexibility via Modularity:** Unlike individual models that offer a fixed quality-cost ratio, compound systems allow for dynamic configuration. Developers can decouple reasoning from knowledge storage, combining smaller, instruction-tuned models (optimised for reasoning) with specialised search heuristics (optimised for knowledge retrieval) to outperform larger, proprietary models at a fraction of the inference cost.
- **Dynamism and Grounding:** By integrating real-time retrieval mechanisms, these systems mitigate key epistemic limitations of pre-trained models. Rather than relying solely on parametric knowledge, the system can query external sources at inference time, grounding responses in current, verifiable data.
- **Control and Verification:** Compound systems provide architectural hooks for intermediate verification. Unlike monolithic generation where errors cascade, a compound system can filter outputs, cross-reference hallucinations against retrieved evidence, and enforce deterministic guardrails—capabilities that are computationally intractable to embed purely within model weights.

Central to this transition is the emergence of *test-time scaling*, which posits that intelligent allocation of computational resources during inference can yield performance gains comparable to, or even exceeding, those achieved through model scaling alone [35]. Snell et al. [30] demonstrate that smaller models utilising sophisticated search strategies can outperform significantly larger models relying on standard greedy decoding, implying a “compute-optimal” sweet spot where additional inference tokens maximise accuracy before hitting diminishing returns. Inference scaling laws formalise this intuition, suggesting that for a fixed computational budget, there exists an optimal trade-off between pre-trained model size and the volume of inference-time compute allocated to reasoning processes.

These principles apply naturally to Retrieval-Augmented Generation (RAG) [16], which can be viewed as a compound reasoning system where retrievers and rerankers function as specialised forms of test-time compute. Just as an LLM “thinks” by generating intermediate reasoning tokens, a RAG system “deliberates” by expending compute to retrieve, filter, and synthesise external evidence. Recent work has begun to quantify this relationship: Yue et al. [39] report that optimally configured inference scaling achieves gains of up to 58.9% over standard RAG on benchmark datasets, while Zhou et al. [44] establish power-law and sigmoidal scaling laws governing the relationship between retriever size, generator size, number of retrieved documents, and context window length. However, these studies do not investigate large-scale query-rewrite fusion—the setting in which multiple reformulated queries are independently retrieved and their results merged—which is the focus of this thesis.

This deliberation unfolds along three primary strategies:

- *Parallel Scaling (Broadening)*: Generating n diverse query reformulations to explore distinct regions of the semantic space, then fusing their results using algorithms such as Reciprocal Rank Fusion (RRF) [4] or Exp4Fuse [18].
- *Sequential Scaling (Deepening)*: Within retrieval pipelines, we define sequential scaling as increasing the candidate set size (c) examined by the neural reranker, expanding the search frontier beyond the initial retrieval cutoff. (Note: in the broader LLM literature, “sequential scaling” often refers to iterative reasoning; here we adopt the RAG-specific interpretation of deepening the reranking stage.)
- *Adaptive Scaling*: Dynamically allocating cross-encoder budget based on corpus structure (ORE [19, 24]) or query reformulation feedback (ReformIR [32]), concentrating expensive scoring on regions of the candidate space predicted to contain relevant documents.

Despite the intuitive appeal of scaling along these dimensions, the field currently lacks rigorous empirical characterisation of their returns, interaction effects, and comparative efficiency.

1.2. Research Gap

The preceding discussion establishes that (1) test-time scaling via “System 2” reasoning is a powerful paradigm for improving LLM performance, (2) RAG pipelines are a natural instantiation of this paradigm where retrieval and reranking serve as externalised forms of inference-time compute, and (3) both parallel (multi-query fusion) and adaptive (ORE, ReformIR) strategies exist for scaling this compute. Recent studies have begun to probe the limits of query reformulation scaling. Venkatesh et al. [32] show that unbounded scaling can hurt performance due to query drift, yet their analysis is limited to a single ranker-feedback threshold and does not systematically characterise the functional form of the scaling curve. This leaves open the question: is the saturation point universal across retrievers and fusion

algorithms, or does it depend critically on the underlying architecture?

Several fundamental questions remain unanswered in the literature:

1. **Empirical scaling laws for multi-query fusion.** While inference scaling laws have been established for single-retrieval-pass RAG [39, 44], existing work has not comprehensively characterised the functional relationship between the number of LLM-generated query rewrites n and downstream retrieval quality across multiple fusion algorithms. Is the relationship logarithmic, power-law, or exponentially saturating? Where is the compute-optimal operating point?
2. **Fusion algorithm comparison at scale.** Prior work typically evaluates fusion methods in isolation or on a single dataset. A comprehensive comparison of score-based (CombMAX), rank-based (RRF, Borda), and hybrid fusion strategies across multiple datasets, retriever architectures (sparse vs. dense), and pipeline orderings (merge-then-rerank vs. rerank-then-merge) is lacking.
3. **Oracle gap quantification.** The gap between what practical fusion algorithms achieve and what an oracle (perfect) fusion could achieve—given the same set of retrieved documents—has received limited attention. This gap directly quantifies the room for algorithmic improvement in fusion design.
4. **Horizontal vs. vertical scaling.** Multi-query fusion scales compute *horizontally* (more queries, more candidates), while adaptive methods like ORE and ReformIR scale *vertically* (deeper analysis of fewer candidates). To the best of our knowledge, no prior work directly compares these two scaling axes within a unified experimental framework.
5. **Cross-dataset generalisation.** Whether the observed scaling behaviours and optimal configurations transfer across datasets with fundamentally different characteristics—multi-hop reasoning (2WikiMultiHopQA [7]), web passage retrieval (MSMARCO [21]), and open-domain QA (TriviaQA/DPR-w100 [14])—remains unexplored.

This thesis addresses each of these gaps through a systematic experimental programme spanning six experimental conditions, three benchmark datasets, two retriever architectures, six fusion algorithms, and two adaptive retrieval baselines.

1.3. Research Questions

To bridge these gaps between theoretical scaling principles and practical implementation, this thesis addresses three research questions:

- RQ1: Fixed-Compute Scaling.** How does scaling retrieval depth (c) and query breadth (n) influence retrieval performance in two-stage reranking pipelines? At what point do additional documents or reformulations yield diminishing returns? How do these dynamics differ between sparse (BM25) and dense (TCT-ColBERT) retrievers, and does reranking each query rewrite independently before fusion yield superior compute efficiency compared to merging first?
- RQ2: Adaptive-Compute Scaling.** Can adaptive retrieval methods—ORE (graph-based candidate exploration with Learnt Affinity) and ReformIR (iterative query reformulation with RM3 pseudo-relevance feedback)—that dynamically allocate cross-encoder budget achieve superior scaling efficiency compared to uniform budget allocation, and how do they compare against multi-query fusion strategies?
- RQ3: End-to-End RAG Evaluation.** How do improvements in retrieval quality (Recall, nDCG) translate

into downstream answer generation quality (Exact Match, F1, LLM-as-Judge), and do retrieval scaling gains persist through the full retrieve $\rightarrow$ rerank $\rightarrow$ generate pipeline?

1.4. Contributions

The principal contributions of this thesis are:

1. **Empirical scaling laws for multi-query retrieval.** We provide a comprehensive empirical characterisation of the functional relationship between the number of LLM-generated query reformulations (n) and retrieval quality across six fusion algorithms, two retriever architectures, and three benchmark datasets. We investigate whether depth and breadth scaling exhibit predictable saturation patterns.
2. **Comprehensive fusion algorithm comparison.** We conduct a large-scale comparison of score-based, rank-based, and hybrid fusion strategies under controlled scaling conditions, including the Exp4Fuse algorithm [18] and the rerank-then-merge (RTM) pipeline ordering.
3. **Adaptive vs. fusion-based scaling comparison.** We present a unified experimental framework that directly compares horizontal scaling (multi-query fusion) against vertical scaling (ORE [24] and ReformIR [32]), examining the conditions under which each strategy is more effective.
4. **End-to-end RAG evaluation.** We evaluate the full retrieval $\rightarrow$ rerank $\rightarrow$ generate pipeline using a multi-metric framework (Exact Match, F1, and an LLM-as-Judge semantic evaluator) to assess how retrieval improvements transfer to answer generation quality.
5. **Open-source experimental framework.** All code, cached query rewrites, corpus graph build scripts, and evaluation pipelines are released as a reproducible experimental framework built on PyTerrier [20]¹.

1.5. Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2: Related Work reviews the foundations of test-time scaling, positions RAG as a compound reasoning system, surveys parallel and sequential scaling strategies, and discusses the adaptive retrieval methods (ORE, ReformIR) that inform our experimental design.

Chapter 3: Implementation Details and Experimental Setup details the datasets, retriever and reranker configurations, fusion algorithms, adaptive method implementations, and evaluation protocols used across all experimental conditions.

Chapter 4: Results and Discussion presents the experimental findings, including the derived scaling laws, fusion algorithm comparisons, adaptive vs. fusion scaling trade-offs, and end-to-end RAG generation results.

Chapter 5: Conclusion summarises the key findings, discusses limitations, and proposes directions for future work.

¹<https://github.com/pratham2442000/MasterThesis>

2

Related Work

2.1. Foundations of Test-Time Scaling

Building upon the modularity of Compound AI Systems [40], the trajectory of AI research has expanded to include Test-Time Scaling (TTS). Historically, performance gains were predicated on training-time scaling, governed by laws established by Kaplan et al. [13] and the Chinchilla team [8], which quantified the power-law relationship between training compute, parameter count, and dataset size. However, contemporary research posits that intelligent allocation of computational resources during the inference phase can yield comparable, if not superior, gains [35]. Recent surveys, such as those by Ji et al. (2025) [10] and Lyu et al. (2025) [42], systematically organize TTS techniques into unified frameworks—distinguishing between output-focused and input-focused strategies, and exploring what to scale, how to scale, where to scale, and its ultimate effectiveness.

This transition effectively operationalises the cognitive framework of Dual-Process Theory within machine learning. Foundational Large Language Models (LLMs) fundamentally operate as “System 1” agents: they are optimised for rapid, heuristic-driven response generation but often succumb to cognitive biases when facing complex logical challenges. In contrast, test-time scaling facilitates “System 2” reasoning—a slower, analytical mode characterised by deliberate, step-by-step processing and logical validation [41]. Whereas standard generation relies on “frozen” weights to predict the next token intuitively, Reasoning LLMs (such as OpenAI o1 or DeepSeek-R1) introduce “thinking processes” that decouple generation from immediate prediction through mechanisms like Structure Search (e.g., Monte Carlo Tree Search) and Macro Actions, allowing for hierarchical planning and iterative self-correction before committing to a final answer.

The core theoretical contribution here is the formalisation of Inference Scaling Laws. These laws suggest that for a fixed computational budget, there exists an optimal trade-off between the size of the pre-trained model and the volume of inference-time compute. By dynamically allocating resources to “slow thinking”—exploring alternative reasoning paths or verifying intermediate steps—these systems can achieve expert-level performance in domains like mathematics and coding that were previously intractable for stochastic, System 1 models.

Formally, the performance P of a reasoning system is defined as a function of its pre-trained parameters θ , the inference-time compute budget $C_{\text{inference}}$, and the quality of verification V :

$$P = f(\theta, C_{\text{inference}}, V) \quad (2.1)$$

Wu et al. [35] demonstrate that smaller models (e.g., 7B parameters) utilising sophisticated search strategies—such as Chain-of-Thought (CoT) or Tree of Thoughts (ToT)—can outperform significantly larger models (e.g., 34B parameters) relying on standard greedy decoding. This implies a “compute-optimal” sweet spot where the marginal utility of additional inference tokens maximises accuracy before hitting diminishing returns.

2.2. RAG as a Compute-Optimal Reasoning System

While initial test-time scaling research focused on solving logic puzzles via internal chain-of-thought, these principles are directly isomorphic to Retrieval-Augmented Generation (RAG) pipelines. In a RAG context, the “search” for a solution is externalised over a document corpus. Therefore, a RAG pipeline should be viewed as a compound reasoning system where the Retriever and Reranker components function as specialized forms of test-time compute. Just as an LLM “thinks” by generating intermediate reasoning tokens, a RAG system “deliberates” by expending compute to retrieve, filter, and rerank external evidence. This creates a Compound Scaling Law, where the system’s total inference cost C_{total} is the aggregate of compute allocated to each module:

$$C_{\text{total}} = C_{\text{retrieval}}(k) + C_{\text{rerank}}(k, \text{depth}) + C_{\text{generation}}(\text{tokens}) \quad (2.2)$$

Optimising this system requires balancing the breadth of the search against the depth of the verification:

2.2.1. Retriever Scaling: Expanding the Search Frontier

The parameter k (number of retrieved documents) serves as a proxy for Parallel Scaling. Increasing k broadens the system’s “search frontier,” increasing the recall probability—i.e., the likelihood that the correct answer exists somewhere within the retrieved context. This is analogous to the “Reasoning Manifold” hypothesis in broad AI research, where sampling more paths increases the probability of intersecting with the correct solution space. However, naïve scaling of k introduces noise and dilutes the attention mechanism of the generator, necessitating a filtration step.

2.2.2. Reranker Scaling: The Verifier / Process Reward Model

The Reranker acts as the system’s Verifier, akin to a Process Reward Model (PRM) in reinforcement learning. By scoring and re-ordering retrieved documents, the reranker prunes irrelevant “branches” of information before they reach the generator. The complexity of the reranker represents a direct lever for test-time compute. A lightweight bi-encoder offers low latency but lower reasoning fidelity, whereas a computationally intensive cross-encoder (which attends to the query and document simultaneously) offers high fidelity at a higher cost (C_{rerank}). Cross-encoder rerankers such as *MonoT5* [22]—which frames relevance estimation as a sequence-to-sequence classification task—and lightweight distilled models like *MiniLM-L6* [34] have become standard second-stage components. High-fidelity reranking significantly improves the “signal-to-noise” ratio in the final context window. Thus, the optimisation of a RAG pipeline is fundamentally a problem of inference-time resource allocation: balancing the

computational cost of high-recall retrieval against the computational cost of high-precision reranking to maximise the probability of a correct response.

2.3. Scaling Retrieval Strategies

The operationalisation of test-time scaling in retrieval-augmented generation manifests through two primary computational axes: parallel scaling (width) and sequential scaling (depth). These strategies represent a fundamental departure from “System 1” retrieval—characterised by a single-pass vector lookup—toward a “System 2” paradigm that treats the corpus as a dynamic environment for exploration and verification [23]. Parallel scaling focuses on maximising the recall of the initial search frontier by diversifying the entry points into the embedding space, whereas sequential scaling emphasises iterative refinement and multi-turn reasoning to resolve complex informational dependencies.

2.3.1. Parallel Scaling: Width and Diversity

Parallel scaling, often termed scaling the “width” of the retrieval process, operates on the principle of redundancy as a mechanism for reliability. Inspired by the “Self-Consistency” decoding strategies used in reasoning LLMs, parallel RAG generates a distribution of query variations to ensure maximum coverage of the document manifold [30]. In high-dimensional semantic spaces, a single query vector represents merely one point of entry into the corpus, which often fails to capture full user intent due to inherent noise in embedding models and the vagaries of human language. Parallel scaling mitigates this risk by generating a diverse array of retrieval intents and executing them simultaneously.

The genesis of any parallel retrieval strategy lies in query rewriting and expansion. The recent survey by Li et al. (2025) [17] provides a comprehensive review of query expansion in the age of pre-trained and large language models, summarising how different model families enable new expansion behaviours, including stronger context awareness and controllable generation. Drawing inspiration from self-consistency decoding, researchers have developed mechanisms to generate multiple query variations under the assumption that while individual formulations may drift, the intersection of their results will converge on ground truth. The literature identifies three principal mechanisms for generating these parallel streams. *Paraphrasing and clarification* involves prompting an LLM to rewrite the user’s query N times, varying in specificity, tone, and vocabulary to cast a wider net. A more sophisticated variant, *Hypothetical Document Embeddings (HyDE)*, instructs the model to generate hypothetical answers to the query and uses these generations as search vectors [6]. When scaled in parallel, multiple hypothetical documents capture various potential framings of the information, proving particularly effective for vocabulary mismatch problems. Furthermore, *On-policy Pseudo-document Query Expansion (OPQE)* lets LLMs learn to generate pseudo-documents that maximise retrieval performance, combining the flexibility of prompting with the optimisation goals of reinforcement learning [36].

The efficacy of parallel scaling is strictly governed by the Rank Fusion layer, which must distil a consensus from the potentially noisy aggregate of retrieved lists. Research on RAG fusion strategies has become an active field in its own right. While basic deduplication provides a foundation for context management, advanced systems employ score-based and rank-based fusion algorithms. *Reciprocal Rank Fusion (RRF)* has emerged as the de facto standard for hybrid systems due to its robustness in combining disparate retrieval modalities (e.g., dense vector search and sparse BM25) without requiring complex normalisation [4]. Alternatively, score-based methods like *CombMAX*, which utilise the maximum score assigned to a document across all lists, focus on minimising the risk of missing relevant documents [27].

Voting Methods such as Borda Count or Majority@ k are employed to enforce strict evidentiary standards, though research suggests they can be sensitive to the “nomination effect” of irrelevant candidates in noisy corpora. Systems like *RAG-Fusion* operationalise these principles by generating multiple query variants and applying RRF to produce richer, more nuanced final rankings. More recently, *Exp4Fuse* [18] modifies the standard RRF formula with an adaptive multiplier proportional to the number of retrieval routes in which a document appears, effectively rewarding cross-route consensus more aggressively than vanilla RRF while penalising noisy LLM-generated queries through an anchoring mechanism tied to the original user intent.

However, scaling the number of rewrites n is not without risk. Recent work by Venkatesh et al. [32] demonstrates that “more reformulations” can systematically hurt retrieval performance due to **query drift**—where successive paraphrases increasingly diverge from the original user intent, introducing noise into the candidate pool. Their analysis reveals that ranker feedback can be leveraged to detect this drift dynamically, halting the scaling process before performance declines. Critically, they show that the optimal number of reformulations varies substantially across queries, underscoring the necessity of understanding the underlying scaling laws. This thesis directly complements their work by systematically characterising the shape of the performance curve across n , retrievers, and fusion methods, thereby identifying the compute-optimal operating points that their drift-detection mechanism would need to target.

At the retriever level, the choice of first-stage retrieval architecture fundamentally determines the search frontier available to downstream scaling strategies. *BM25* [25], grounded in the probabilistic relevance framework, remains the dominant sparse retrieval baseline due to its efficiency and strong lexical matching capabilities. The paradigm shift to dense retrieval was catalysed by *Dense Passage Retrieval (DPR)* [14], which demonstrated that dual-encoder architectures trained on question–passage pairs could outperform BM25 on open-domain QA by projecting queries and documents into a shared continuous vector space. *ColBERT* and its improved version *ColBERTv2* [26] advanced the frontier further through a late interaction mechanism that performs token-level similarity decomposition—ColBERTv2 combines aggressive residual compression with denoised supervision, drastically improving both quality and space footprint while outperforming prior retrievers across 28 datasets. Bridging the lexical–semantic divide, *SPLADE* [5] introduced a learned sparse retriever that projects dense contextualised representations back into the vocabulary space, performing implicit query expansion while retaining the efficiency of inverted index lookup. The *BEIR* benchmark [31] has become the standard testbed for evaluating the zero-shot generalisation of these retrieval architectures across diverse domains.

2.3.2. Sequential Scaling: Depth and Iteration

Sequential scaling expands the computational budget temporally, treating retrieval as a multi-turn conversation between the model and the index. This “Depth” strategy is essential for multi-hop reasoning tasks where the information required for step $t + 1$ is contingent upon the results of step t . Foundational frameworks such as *Chain-of-Retrieval (CoRAG)* formalise this process by fine-tuning models to predict discrete actions—such as “Retrieve,” “Reason,” or “Refine”—within a continuous reasoning trace [33]. By utilising rejection sampling during training, these models learn to internalise the search process, transforming “slow” search processes into “fast” mental heuristics over time.

The frontier of sequential scaling lies in *Agentic RAG* and search-based architectures. A comprehensive survey by Singh et al. [29] maps the design space of agentic retrieval systems, identifying key architectural patterns including tool-augmented agents, multi-step planners, and self-reflective pipelines. Within this

landscape, foundational work on interleaving reasoning with action has proven particularly influential: *ReAct* [38] demonstrated that alternating between reasoning traces and tool actions (such as issuing search queries) substantially improves complex task-solving by grounding the model’s chain-of-thought in external evidence. Building on this principle, *Self-RAG* [2] introduces a framework where the model learns to reflect on its own generations via special self-reflection tokens, dynamically invoking retrieval only when its internal knowledge is insufficient—effectively learning *when* to retrieve rather than always retrieving. More recently, *MCTS-RAG* [9] applies Monte Carlo Tree Search to the retrieval space, allowing the system to propose multiple search plans, evaluate their “promise” via a Process Reward Model (PRM), and backtrack when a retrieval path leads to a logical dead end. This adaptive compute allocation ensures that resources are concentrated on the most promising reasoning branches.

Furthermore, the “*Test-Time Corpus Feedback*” paradigm introduces algorithms like *Query Affinity Modelling (QUAM)*, which analyze the semantic connectivity of the corpus in real-time to traverse “concept bridges” that do not lexically match the original query [23]. These iterative loops allow smaller, agile models to achieve performance parity with frontier monolithic models by effectively trading increased inference latency for superior reasoning depth. Additionally, frameworks like *REPLUG* (Retrieval-Augmented Black-Box Language Models) [28] treat the LM as a black box and augment it with a tunable retrieval model. *REPLUG-LSR* (LM-Supervised Retrieval) further leverages the black-box LM’s supervision to improve the initial retriever, demonstrating up to a 6.3% performance improvement on GPT-3 (175B) for language modeling tasks.

2.4. Adaptive Retrieval and Reformulation Strategies

Beyond the static parallel and sequential paradigms described above, a growing body of work explores *adaptive* retrieval strategies that dynamically allocate compute based on query difficulty or corpus structure. These methods occupy a middle ground: they are neither purely width-based nor purely depth-based, but instead learn to modulate the retrieval effort at inference time. Two methods are particularly relevant to this thesis: *ORE*, which navigates a corpus graph to discover new candidates, and *ReformIR*, which iteratively reformulates queries to expand the retrieval frontier. Both employ constrained optimisation to allocate cross-encoder budget selectively.

2.4.1. ORE: Adaptive Re-Ranking with a Corpus Graph

The ORE framework, by Rathee et al. [24], scales retrieval quality by combining iterative graph-based candidate exploration with selective cross-encoder reranking under a constrained inference budget. The system comprises three interacting components: (1) a **dual-encoder** (TasB) that provides cheap initial similarity estimates between queries and documents, (2) a **corpus graph** constructed from BM25 k -nearest neighbours ($k = 16$) that encodes structural proximity between documents, and (3) a **cross-encoder** that produces high-fidelity relevance scores at substantially higher computational cost.

The retrieval process follows a bandit-inspired active-learning loop. Given an initial BM25 candidate pool of c documents, ORE maintains a set of “arms” (candidate documents) and iteratively selects batches for expensive cross-encoder scoring. At each iteration, the system identifies *cluster heads*—the top_s highest-scoring documents from the scored pool—and uses the **Learnt Affinity (LAFF)** graph ($k = 128$ neighbours) to discover new candidates in the neighbourhood of these heads. LAFF edge weights encode pairwise document similarity learned from relevance supervision, allowing the system to traverse semantically related regions of the corpus that may not share lexical overlap with the original

query.

Candidate prioritisation is governed by a λ -**regression model** fitted at each iteration via constrained least-squares (`lsq_linear`) on three features: (i) the BM25 score of the candidate, (ii) its mean LAFF affinity to current cluster heads, and (iii) the mean cross-encoder score of its scored neighbours. The learned weights $\lambda_1, \lambda_2, \lambda_3$ allow the system to predict cross-encoder scores for unscored documents, enabling it to rank the full candidate pool without exhaustive cross-encoder evaluation. This regression-based score approximation is the key mechanism that allows ORE to operate under a *fixed* cross-encoder budget: at large retrieval depths (e.g., $c = 1000$), only a fraction of candidates receive real cross-encoder scores, while the remainder are ranked using the regression predictions.

2.4.2. ReformIR: Adaptive Retrieval via Query Reformulation

ReformIR [32] takes a complementary approach by leveraging LLM query reformulations combined with pseudo-relevance feedback to iteratively expand the retrieval frontier. Rather than navigating a static corpus graph, ReformIR dynamically generates new retrieval trajectories through query reformulation, making it a natural counterpart to the multi-query fusion strategies studied in this thesis.

The algorithm operates as follows. Given the original query q and n LLM-generated reformulations $\{q_1, \dots, q_n\}$, ReformIR first retrieves BM25 results for each variant independently and constructs a **hybrid candidate pool** where each document receives a weighted combination score across all sources. The source weights are initialised uniformly ($\lambda_i = 1/(n + 1)$) and subsequently learned via constrained least-squares regression. The iterative scoring loop then proceeds in fixed-size batches: at each iteration, the system (1) selects the highest-priority unscored candidates based on estimated scores, (2) scores them with a cross-encoder, (3) selects the top-15 scored documents as candidates for **RM3 pseudo-relevance feedback**, which uses 5 feedback documents to extract 10 expansion terms with interpolation weight $\lambda_{\text{RM3}} = 0.3$, (4) retrieves additional candidates via BM25 on the expanded query, and (5) re-fits the weight vector using `lsq_linear` with bounds $[0, 1]$ on features including hybrid BM25 scores, RM3 scores, and per-source BM25 scores.

The key design parameter is the **cross-encoder budget** (ce), which caps the number of scoring iterations. When ce is scaled linearly with the retrieval depth ($ce = \lceil c/16 \rceil$), all candidates receive real cross-encoder scores and the regression model’s predictions are unused. When ce is held fixed (e.g., $ce = 7$), the system must extrapolate beyond the scored training set, and the least-squares linear model determines the final ranking for tail candidates. This budget-constrained setting directly tests whether the linear approximation can generalise beyond its calibration regime.

2.4.3. Comparison and Thesis Relevance

Both ORE and ReformIR share a common insight: the cross-encoder—the most expensive component in a retrieval pipeline—should not be applied uniformly across all candidates, but rather allocated strategically. This mirrors the broader test-time scaling principle of concentrating compute where it yields the highest marginal return. However, they differ fundamentally in *how* they expand the candidate frontier: ORE discovers new documents through *structural graph traversal* over a pre-built corpus graph, while ReformIR generates new candidates through *dynamic query reformulation* via RM3 pseudo-relevance feedback. Crucially, these adaptive methods can be compared directly against the multi-query fusion strategies studied in this thesis: whereas fusion scales compute *horizontally* (more queries, more candidates), adaptive methods scale compute *vertically* (deeper analysis of fewer

candidates). Understanding when each strategy dominates is a central question of this work.

2.5. Inference Scaling Laws for RAG

The quantitative understanding of inference-time scaling in RAG is rapidly solidifying. At ICLR 2025, Yue et al. [39] demonstrated that when optimally configured, increasing inference compute leads to *near-linear gains* in RAG performance. They developed compute allocation models to estimate RAG performance under various inference configurations, which closely match experimental results. By applying these optimal configurations, scaling inference compute on long-context LLMs achieved gains of up to **58.9%** over standard RAG on benchmark datasets.

In a more systematic study at AAAI 2026, Zhou et al. [44] examined how inference performance scales with critical factors such as retriever model size, generator model size, number of retrieved documents, and context window size. Through extensive benchmarking, they established empirical *power-law* and *sigmoidal* relationships as scaling laws for these factors and further constructed a theoretically grounded joint inference scaling law. These laws enable us to understand and predict the performance trends of RAG models under varying computational resources, providing rigorous mathematical foundations for the empirical observations found in parallel and sequential scaling strategies.

However, a critical gap remains: existing inference scaling laws characterise the relationship between a *single* retrieval pass and performance, but do not account for the multi-query fusion setting where n independent retrieval passes are aggregated. The functional form of the relationship between n and downstream metrics—whether logarithmic, power-law, or exponentially saturating—has not been established, nor has the interaction between the number of rewrites n , the choice of fusion algorithm, and the pipeline ordering (merge-then-rerank vs. rerank-then-merge) been systematically quantified. This thesis directly addresses this gap.

2.6. Evaluation Datasets and Benchmarks

To rigorously evaluate the efficacy of Retrieval-Augmented Generation pipelines as Compound AI Systems, the underlying evaluation benchmarks must extend beyond simple factoid retrieval. Standard single-hop datasets often fail to adequately challenge the “System 2” reasoning capabilities required for complex test-time compute scaling, as they typically reward surface-level lexical matching rather than deliberative logic. A systematic review of RAG evaluation, surveying 63 academic articles [3], has comprehensively mapped evaluation approaches across four key areas: datasets, retrievers, indexes and databases, and generator components. Additionally, a dedicated survey on RAG datasets analyzed 30 core research papers to distil 148 relevant datasets, proposing a hierarchical taxonomy covering six broad categories: Question Answering, Fact Verification, Slot Filling, Multimodal Tasks, Specialized Applications, and Evaluation & Benchmarking.

2.6.1. 2WikiMultiHopQA and Multi-Hop Benchmarks

To assess the system’s ability to navigate hierarchical planning and multi-step retrieval, we employ the 2WikiMultiHopQA dataset. Constructed from a synthesis of structured Wikidata triples and unstructured Wikipedia text, 2WikiMultiHopQA provides a rigorous testbed by explicitly annotating the logical evidence paths required to resolve multi-hop queries [7]. The dataset is categorised into four distinct reasoning typologies: comparison, inference, compositional, and bridge comparison. This multi-hop structure is particularly well-suited for evaluating the “Broadening” and “Deepening” strategies of

test-time scaling. For instance, resolving a compositional query requires the RAG pipeline to decompose a complex prompt into constituent entities and navigate across “concept bridges” that may not exist within a single document’s context. By measuring performance within this environment, we can precisely quantify the extent to which an expanded inference-time compute budget improves the “signal-to-noise” ratio across intricate reasoning trajectories.

2.6.2. Additional Widely-Used Benchmarks

Beyond multi-hop reasoning, the research community relies on a suite of established benchmarks for comprehensive evaluation. *MSMARCO-Passage* [21] is a large-scale web passage retrieval benchmark containing approximately 8.8 million documents and over 500k queries, originally constructed from Bing search logs and widely used to train and evaluate modern dense retrievers and cross-encoders at scale. *DPR-w100 (TriviaQA)* serves as a dense passage retrieval benchmark using the TriviaQA question set with 100-word Wikipedia passages, testing systems on open-domain question answering [12]. *HotpotQA* [37] is a multi-hop QA dataset requiring models to reason and synthesise answers from multiple Wikipedia paragraphs. *Natural Questions (NQ)* [15] comprises Google search queries with Wikipedia-sourced answers, providing a realistic open-domain retrieval challenge.

Recent research has also explored the efficacy of using LLMs themselves as evaluators for RAG systems. Zheng et al. [43] validated the use of strong models (such as GPT-4) as automated evaluators, establishing MT-Bench and Chatbot Arena as standard preference-tracking platforms. Their findings demonstrate that LLM-based judgments achieve high agreement with human evaluators on most tasks, though systematic biases—including position bias, verbosity bias, and self-enhancement bias—can distort ratings when proper controls are absent. Automated evaluation methods that leverage LLMs to both generate evaluation datasets and perform the evaluation itself offer scalability advantages, but the balance between LLM-based evaluation and human judgment remains an ongoing discussion, focusing on the respective contributions, limitations, and challenges of automation versus human input.

2.7. Summary

The preceding sections establish the theoretical and methodological foundations upon which this thesis builds. Training-time scaling laws [13, 8] and their inference-time counterparts [35, 30] provide the theoretical grounding. RAG pipelines [16] instantiate these principles as compound reasoning systems where retrieval and reranking serve as externalised forms of test-time compute. Both parallel strategies (multi-query fusion via RRF, Borda, Exp4Fuse) and adaptive methods (ORE, ReformIR) offer distinct mechanisms for scaling this compute. The specific research gaps addressed by this thesis and the corresponding research questions are formalised in Section 1.2.

3

Implementation Details and Experimental Setup

3.1. Experimental Setup and Design

To provide a rigorous empirical evaluation of the proposed test-time scaling strategies, we established a multi-phase experimental framework designed to isolate and quantify the scaling laws governing retrieval-augmented generation (RAG) pipelines. This framework operationalises the two primary axes of computational expansion discussed in the previous chapter: parallel scaling and sequential/adaptive scaling.

The “Deepening” axis examines sequential scaling by varying the document candidate pool size (c) for neural reranking, while the “Broadening” axis implements parallel scaling strategies through the generation of n diverse query reformulations. By systematically varying these parameters, we evaluate the system’s ability to explore the semantic manifold and expand the search frontier—mechanisms essential for transitioning from heuristic “System 1” retrieval to deliberative “System 2” reasoning.

To bridge the research gaps identified in Section 1.2, our methodology evaluates these strategies across three benchmark datasets, two first-stage retriever architectures (sparse and dense), two second-stage rerankers, six fusion algorithms, and two adaptive retrieval baselines (ORE and ReformIR). The following sections detail the data provenance, the architectural configurations of the compound system, and the specific protocols employed for each experimental iteration.

3.2. Hardware and Implementation Details

Experiments were conducted across three computational environments depending on the scale and resource requirements of the respective experiment: (1) the DelftBlue High Performance Computing (HPC) cluster, equipped with NVIDIA V100 GPUs (32GB VRAM) and AMD EPYC 7402 24-Core CPUs [1]¹; (2) Google Colab High-RAM instances with NVIDIA T4 GPUs; and (3) a local workstation

¹Preliminary experiments were conducted on DelftBlue; however, final reported experiments were executed using the environments specified in Appendix A.

comprising an AMD Ryzen 7 4800H processor and NVIDIA RTX 2060 GPU. A detailed mapping between experiments and execution environments is provided in Appendix A.

All retrieval and reranking pipelines were implemented in Python 3.11 using the PyTerrier information retrieval framework [20]. Dense retrieval functionality was provided through PyTerrier-DR, neural reranking through PyTerrier-T5, and adaptive retrieval algorithms through PyTerrier-Adaptive. Neural models were executed using PyTorch 2.1 and the Hugging Face Transformers ecosystem. Dense vector indexing utilised Faiss for efficient similarity search [11].

To ensure reproducibility, all query rewrites were generated once and cached prior to experimentation, random seeds were fixed where applicable, and evaluation was performed using deterministic inference settings. Experimental outputs, intermediate retrieval runs, and generated rewrites were persisted to disk to avoid variability arising from repeated model execution.

3.2.1. First-Stage Retrievers

Two architecturally distinct first-stage retrievers were evaluated:

- **Sparse Retrieval (BM25):** Implemented via PyTerrier with default Terrier hyper-parameters. BM25 performs lexical matching using term frequency, inverse document frequency, and document length normalisation, representing the fast, heuristic retrieval component analogous to Kahneman’s System 1 reasoning paradigm, where decisions are made through efficient lexical matching rather than deliberate semantic analysis.
- **Dense Retrieval (TCT-ColBERT v2-HNP):** The model `castorini/tct_colbert-v2-hnp-msmarco` combines the efficiency of ColBERT’s late interaction mechanism with knowledge distillation training from a teacher cross-encoder. Documents are indexed using FaissFlat for exact nearest-neighbour search.

3.2.2. Second-Stage Rerankers

Two cross-encoder rerankers were evaluated in the depth scaling experiment (Section 3.4) to establish architectural baselines:

- **MonoT5:** A sequence-to-sequence model fine-tuned for passage ranking (`castorini/monot5-base-msmarco`), with an inference batch size of 16.
- **MiniLM-L6:** A lightweight cross-encoder (`cross-encoder/ms-marco-MiniLM-L6-v2`), with an inference batch size of 16. In the depth scaling experiment, MiniLM-L6 consistently matched or exceeded MonoT5 on nDCG@10 across all three datasets and both retriever architectures, while exhibiting smaller variance under increasing candidate pool sizes (see Section 4.1). This robustness to noisy candidates at large c —where many irrelevant documents enter the reranking pool—motivated the selection of MiniLM-L6 as the default reranker for all subsequent scaling experiments.

3.3. Evaluation Datasets

To ensure the robustness and generalisability of our findings, all experiments (unless otherwise specified) are conducted across three benchmark datasets spanning fundamentally different retrieval challenges. Table 3.2 provides a summary of dataset statistics.

Table 3.1: Summary of all models used across the experimental pipeline.

Stage	Method	Model / Checkpoint	Type
Retrieval	BM25	Terrier (default params)	Lexical
	TCT-ColBERT	castorini/tct_colbert-v2-hnp-msmarco	Dense bi-encoder
Reranking	MonoT5	castorini/monot5-base-msmarco	Seq2seq
	MiniLM-L6	cross-encoder/ms-marco-MiniLM-L6-v2	Cross-encoder
Generation	Llama 3.2	meta-llama/Llama-3.2-3B-Instruct	Causal LM
Rewriting	GPT-4o-mini	OpenAI API ($\tau = 0.7$)	Decoder-only

Table 3.2: Summary statistics of evaluation datasets.

Dataset	Queries	Corpus Size	Avg. Doc Length	Task	Eval. Split
2WikiMultiHopQA	100 [†]	495,034	393 chars	Multi-hop QA	dev
MSMARCO	43	8.8M	—	Web Retrieval	TREC-DL 2019
DPR-w100	100	21M	100 words	Open-domain QA	dev

[†]Stratified sample from 12,576 queries; see Section 3.3.

3.3.1. 2WikiMultiHopQA (2WikiMultiHopQA)

A multi-hop question answering dataset requiring reasoning across multiple Wikipedia passages[7]. It contains 12,576 queries in the development set, each explicitly annotated with logical evidence paths. The dataset is categorised into four reasoning typologies with a stratified sampling distribution: compositional (50%, 50 queries), comparison (30%, 30 queries), bridge comparison (10%, 10 queries), and inference (10%, 10 queries). The corpus comprises 495,034 documents with an average document length of 393 characters. This dataset is particularly well-suited for stress-testing the “System 2” multi-hop evidence synthesis capabilities of the retrieval pipeline. Following prior retrieval evaluation practice, a representative subset of 100 queries was used to permit exhaustive parameter sweeps across multiple retrieval architectures and scaling configurations while maintaining tractable computational cost. Stratified sampling preserves the original reasoning-type distribution, reducing the risk of evaluation bias.

3.3.2. MSMARCO-Passage

A large-scale web passage retrieval benchmark containing approximately 8.8 million passages[21]. We utilise the TREC Deep Learning 2019 judged query set (`irds:msmarco-passage/trec-dl-2019/judged`), which contains 43 queries with graded relevance judgments. MSMARCO serves as the standard benchmark for evaluating ad-hoc retrieval systems on open-web queries, providing the largest and most diverse corpus in our evaluation suite.

3.3.3. DPR-w100 (TriviaQA)

A dense passage retrieval benchmark using the TriviaQA question set with 100-word Wikipedia passages (`irds:dpr-w100/trivia-qa/dev`) [12]. The first 100 queries in the `ir_datasets` iteration order are used for evaluation (no random sampling is applied); this deterministic selection ensures exact reproducibility across runs. This dataset tests the system’s efficacy on open-domain factoid question answering, where multiple valid answer phrasings may exist across distinct passages.

3.3.4. Sampling and Determinism

For all experiments, we evaluate 100 queries from the respective development sets (or all available queries when the dataset contains fewer than 100 judged topics, as is the case for MS MARCO with 43 queries). For 2WikiMultiHopQA, queries are drawn via stratified random sampling with a fixed seed (`random.seed(42)`) to maintain the original reasoning-type distribution. For DPR-w100, the first 100 queries in the `ir_datasets` iteration order are used without shuffling. All query rewrites are pre-generated offline using GPT-4o-mini (30 rewrites per query, $\tau = 0.7$) and cached to guarantee deterministic experimental conditions across all runs, eliminating stochastic variation from the LLM rewriting process. The full system prompt governing rewrite generation is provided in Appendix A.2.1. The decision to generate substantially more rewrites than required by any individual experiment ensures that all rewrite-count conditions are nested. Consequently, increasing n corresponds to a strict expansion of the available rewrite set rather than regeneration of alternative rewrites, allowing observed performance differences to be attributed solely to the scaling parameter itself.

3.4. Impact of Retrieval Depth (Scaling c)

This experiment establishes the retrieval-quality baseline curves used throughout the thesis by analysing how the initial retrieval cutoff (c) affects downstream retrieval and reranking performance. The experiment evaluates both sparse (BM25) and dense (TCT-ColBERT) first-stage retrieval under progressively larger candidate budgets. The cutoff c is swept logarithmically across 10 values using `numpy.geomspace`. This logarithmic spacing follows common practice in scaling-law studies [13, 8], providing dense measurements in low-compute regions where marginal gains change rapidly while avoiding redundant evaluations in saturation regimes:

$$c \in \{10, 16, 27, 46, 77, 129, 215, 359, 599, 1000\}$$

Each retriever is evaluated in isolation (baseline) and with each of the two second-stage rerankers (MonoT5 and MiniLM-L6), producing a total of six retrieval pipelines. The independent variable is c ; the fixed variable is $n = 1$ (original query only, no rewrites). Unlike the adaptive methods ORE and ReformIR (Section 3.7), this experiment performs *exhaustive reranking*: every document in the top- c candidate set receives a real reranker score without approximation or selective scoring.

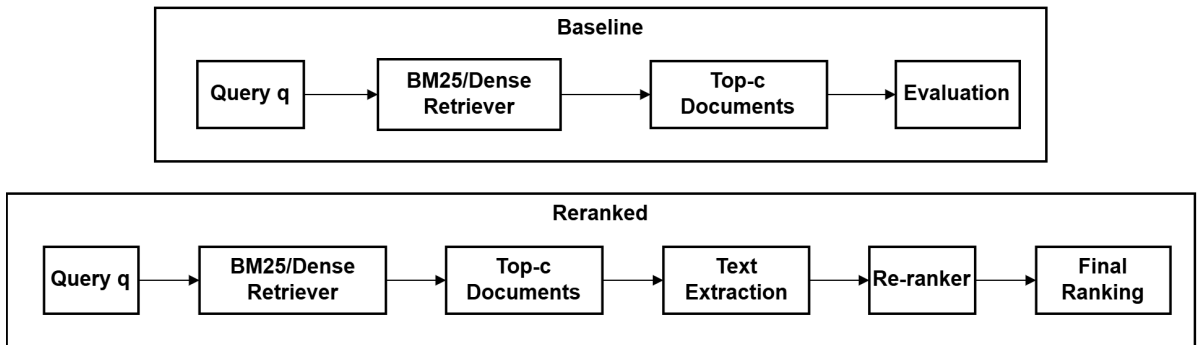


Figure 3.1: Schematic illustration of the two pipelines compared in the depth scaling experiment. The top panel depicts the Sparse and Dense baseline, where the top- c documents retrieved by the sparse/dense retriever are evaluated directly. The bottom panel depicts the two-stage pipeline, where the initial candidate set is reranked before evaluation.

Text Extraction Step. In all two-stage pipelines, a “Get Text” step intervenes between first-stage retrieval

Table 3.3: Depth scaling configuration. Six pipelines are evaluated: BM25, BM25+MonoT5, BM25+MiniLM-L6, Dense, Dense+MonoT5, Dense+MiniLM-L6.

Parameter	Value
Independent variable	$c \in \{10, 16, 27, 46, 77, 129, 215, 359, 599, 1000\}$
Sweep spacing	geospace(10, 1000, 10)
Fixed variable	$n = 1$ (original query only)
Retrievers	BM25, TCT-ColBERT v2-HNP
Rerankers	MonoT5, MiniLM-L6
Topic limit	100 queries (43 for MSMARCO)
Evaluation metrics	R@10, R@50, nDCG@10, R@ c
Batch size	16 (both rerankers)

and reranking. The first-stage retriever returns only document identifiers and retrieval scores; the text extraction step resolves each identifier to its full passage body by looking up the document text from either the index metaindex (for 2WikiMultiHopQA) or the `ir_datasets` corpus (for MSMARCO and DPR-w100, where the pre-built Terrier index does not store passage text in its metadata). This ensures the cross-encoder reranker receives the plain-text passage required for query–document relevance scoring.

Evaluation is performed using four metrics: Recall@10, Recall@50, nDCG@10, and a dynamic Recall@ c that measures the ceiling recall obtainable at each retrieval budget. The logarithmic spacing provides denser sampling at small budgets—where recall curves change most rapidly—and sparser sampling at large budgets where gains begin to saturate.

The resulting curves serve three purposes: (1) quantifying the benefit of semantic retrieval over lexical matching, (2) measuring the effectiveness of each reranking architecture, and (3) establishing the computational ceiling that later budget-aware methods attempt to approximate more efficiently.

3.5. Impact of Retrieval Breadth (Scaling n)

This experiment focuses on “Broadening” via parallel query rewriting with sparse BM25 retrieval. The independent variable is the number of query rewrites n , tested at 10 linearly spaced points using `numpy.linspace` (float values are cast to `int` via NumPy’s floor-truncation):

$$n \in \{1, 4, 7, 10, 13, 17, 20, 23, 26, 30\}$$

The per-rewrite retrieval depth is fixed at $c = 50$, a moderate value selected based on the saturation points observed in Section 3.4. The first-stage retriever is BM25 and the second-stage reranker is MiniLM-L6 (selected based on the robustness analysis from Section 3.4; MonoT5 results are reported in the appendix for completeness).

For each original query, n variants are generated and independently dispatched to BM25, producing n ranked lists. These lists are aggregated using a fusion algorithm into a single candidate list, which is subsequently reranked by the cross-encoder. The basic experimental workflow is illustrated in Figures 3.2 and 3.3.

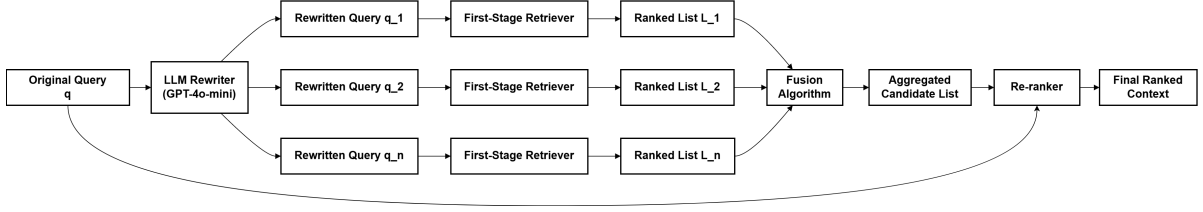


Figure 3.3: Experimental workflow for query broadening via parallel rewriting. The original query q is passed to an LLM rewriter, which generates n semantically diverse variants. Each rewritten query is independently issued to BM25. The resulting n ranked lists are aggregated by a fusion algorithm into a single candidate list, which is subsequently reranked by MiniLM-L6 to produce the final ranked context. This architecture corresponds to the merge-then-rerank paradigm illustrated in the top panel of Figure 3.2.

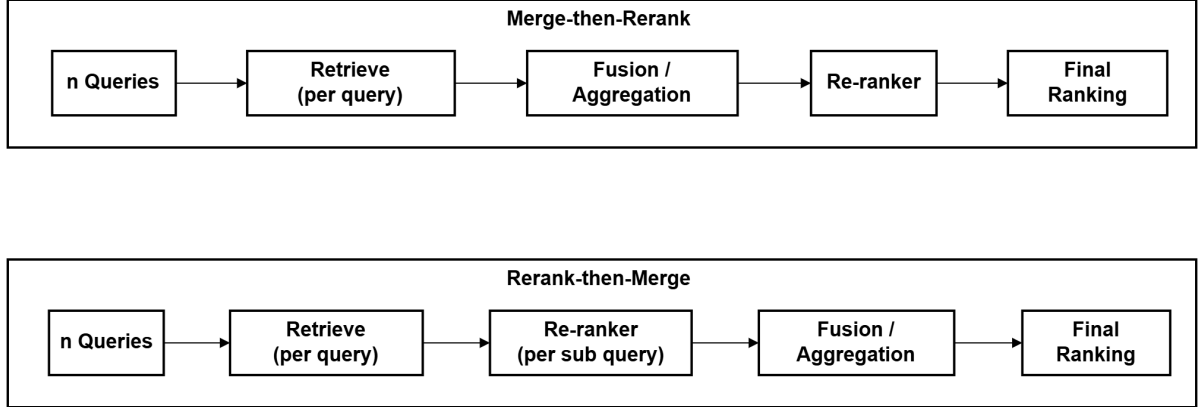


Figure 3.2: Comparison of fusion architectures for query broadening. The top panel illustrates the merge-then-rerank (MTR) paradigm used in the sparse and dense breadth experiments (Sections 3.5 and 3.5.3). The bottom panel illustrates the rerank-then-merge (RTM) paradigm (Section 3.5.4).

3.5.1. Query Rewriting Configuration

The query rewriting module serves as the primary mechanism for parallel scaling. We utilised the GPT-4o-mini model via the OpenAI API, configured with a temperature of $\tau = 0.7$. This temperature setting was empirically determined to balance linguistic diversity with preservation of the original retrieval intent. The rewriting process was governed by a system prompt designed to enforce three-dimensional diversity:

1. **Lexical diversity:** Substitution of synonyms and variation of surface forms.
2. **Structural diversity:** Alteration of grammatical configuration, transitioning between interrogative, declarative, and inverted-order syntaxes.
3. **Semantic diversity:** Generation of variants that emphasise different retrieval sub-tasks, such as entity-centric queries or attribute-focused phrasings.

This multi-dimensional expansion aims to overcome the “query–document mismatch” problem inherent in single-query retrieval. For each query, 30 rewrites are pre-generated and cached; experiments that require $n < 30$ rewrites select the first n from this cached set. Examples of rewritten queries can be found in Appendix A.2.

3.5.2. Fusion Strategies

Employing parallel scaling via query rewriting produces n distinct ranked lists of documents, $\{L_1, L_2, \dots, L_n\}$. To synthesize these into a single cohesive ranking without overwhelming the downstream reranker, we evaluate six distinct fusion algorithms. Each algorithm represents a different hypothesis about how to aggregate evidence from parallel retrieval streams. From a test-time scaling perspective, fusion functions act as compute-allocation mechanisms. Given a fixed retrieval budget, each algorithm determines how evidence collected from parallel retrieval trajectories is aggregated before being exposed to the downstream reranker. Consequently, differences between fusion methods reflect alternative strategies for converting additional inference-time computation into retrieval utility.

Deduplication + Cutoff (DedupCutoff). The deduplication-based fusion strategy merges all sub-query results and retains, for each unique (q, d) pair, only the instance with the highest retrieval score. The merged list is then truncated to the top- c documents by score before reranking, producing a fixed-size reranking pool regardless of n . DedupCutoff is computationally predictable but risks discarding relevant documents that appear only at lower ranks across multiple rewrites. A non-truncated variant (*Dedup*) that passes the full merged pool to the reranker without cutoff is used exclusively in the oracle analysis (Section 3.5.5).

Borda Count (Borda). A rank-based voting mechanism that assumes relevant documents should achieve consensus across multiple query variants. The normalized Borda score for each document d is:

$$\text{Borda}(d) = \frac{1}{n} \sum_{r=1}^n \frac{|L_r| - \text{rank}_r(d)}{|L_r|} \quad (3.1)$$

where $\text{rank}_r(d)$ is the rank of d in list L_r (assigned $|L_r|$ if absent). The top- c documents by aggregated score are retained.

Weighted Borda (BordaWeighted). Standard Borda assumes linear decay in document utility with rank. To model the typically steep power-law distribution of relevance in retrieval, Weighted Borda introduces an exponential weighting function:

$$\text{BordaW}(d) = \frac{1}{n} \sum_{r=1}^n \frac{|L_r| - \text{rank}_r(d)}{|L_r|} \cdot \exp\left(0.1 \cdot \frac{|L_r| - \text{rank}_r(d)}{|L_r|}\right) \quad (3.2)$$

CombMAX. A score-based method that assigns each document a final score equal to the maximum retrieval score it received across all n lists:

$$\text{CombMAX}(d) = \max_{r \in \{1, \dots, n\}} \text{score}_r(d) \quad (3.3)$$

CombMAX preserves strong signals from individual rewrites that achieve excellent semantic alignment, making it effective for ambiguous queries where only one formulation leads to the correct answer.

Reciprocal Rank Fusion (RRF). A robust rank-based fusion that combines lists without requiring score normalisation:

$$\text{RRF}(d) = \sum_{r=1}^n \frac{1}{k + \text{rank}_r(d)} \quad (3.4)$$

where $k = 60$ is a smoothing constant that prevents excessive influence from top-ranked positions [4].

Table 3.4: Sparse retrieval breadth configuration.

Parameter	Value
Independent variable	$n \in \{1, 4, 7, 10, 13, 17, 20, 23, 26, 30\}$
Sweep spacing	<code>linspace(1, 30, 10)</code>
Fixed cutoff (c)	50
Retriever	BM25
Reranker	MiniLM-L6 (MonoT5 results in appendix)
Fusion methods	DedupCutoff, Borda, BordaWeighted, CombMAX, RRF, Exp4Fuse
Topic limit	100 queries (43 for MSMARCO)
Evaluation metrics	R@50, R@10, nDCG@10
Datasets	2WikiMultiHopQA, MSMARCO, DPR-w100

Exp4Fuse. A modified RRF variant from the Exp4Fuse framework [18] that introduces an adaptive multiplier proportional to the number of retrieval routes in which a document appears:

$$\text{Exp4Fuse}(d) = \left(\frac{m_d}{10} + 1\right) \sum_{r=1}^n \frac{w_r}{k + \text{rank}_r(d)} \quad (3.5)$$

where m_d is the number of distinct ranked lists in which document d appears, w_r is the route weight (1.0 for both the original query and expanded queries in our configuration), and $k = 60$. The multiplier $(\frac{m_d}{10} + 1)$ boosts documents retrieved by multiple query variants, rewarding cross-route consensus more aggressively than standard RRF.

Tie-Breaking. All fusion methods break tied aggregated scores by retaining the instance with the highest original retrieval score. If scores remain tied, documents are ordered by document identifier (docno) to ensure deterministic output.

3.5.3. Dense Retrieval Fusion

To assess whether scaling behaviour differs between sparse and dense first-stage retrievers, this experiment replicates the design of Section 3.5 but replaces BM25 with the TCT-ColBERT dense retriever (using FaissFlat indexing). Because dense retrievers operate in a continuous semantic space, distinct rewrites may activate completely different regions of the latent space. We hypothesise that dense retrieval may exhibit a slower saturation rate with increasing n , as semantically distinct rewrites can activate different regions of the learned embedding space, potentially exposing relevant documents that remain inaccessible to lexical retrieval methods.

A key difference from the sparse setting is the set of evaluated fusion methods: this experiment evaluates four fusion algorithms—Borda, CombMAX, RRF, and Exp4Fuse—omitting DedupCutoff and BordaWeighted, which exhibited consistently poor performance or similar performance to other methods in the sparse setting. Due to computational constraints (dense index construction and GPU memory requirements), this analysis is conducted on the 2WikiMultiHopQA and MSMARCO datasets only; TriviaQA (DPR-w100) is omitted from the dense-fusion investigation.

3.5.4. Rerank-then-Merge Paradigm

While the sparse and dense breadth experiments (Sections 3.5 and 3.5.3) apply fusion before reranking (merge-then-rerank, MTR), this experiment reverses the pipeline ordering: each rewrite’s retrieved list is first individually reranked by MiniLM-L6, and the n reranked lists are subsequently fused. This design

Table 3.5: Dense retrieval breadth configuration.

Parameter	Value
Independent variable	$n \in \{1, 4, 7, 10, 13, 17, 20, 23, 26, 30\}$
Fixed cutoff (c)	50
Retriever	TCT-ColBERT v2-HNP (FaissFlat)
Reranker	MiniLM-L6
Fusion methods	Borda, CombMAX, RRF, Exp4Fuse
Datasets	2WikiMultiHopQA, MSMARCO
Evaluation metrics	R@50, R@10, nDCG@10

Table 3.6: Rerank-then-merge (RTM) breadth configuration.

Parameter	Value
Independent variable	$n \in \{1, 4, 7, 10, 13, 17, 20, 23, 26, 30\}$
Fixed cutoff (c)	50
Retriever	BM25
Reranker	MiniLM-L6 (applied per rewrite, before fusion)
Fusion methods	DedupCutoff, Borda, BordaWeighted, CombMAX, RRF, Exp4Fuse
Datasets	2WikiMultiHopQA, MSMARCO, DPR-w100
Evaluation metrics	R@50, R@10, nDCG@10

tests whether higher-quality, reranked per-trajectory lists improve final fusion outcomes. The same six fusion algorithms from Section 3.5 are evaluated (DedupCutoff, Borda, BordaWeighted, CombMAX, RRF, Exp4Fuse).

Crucially, the computational cost profile differs substantially: in the RTM paradigm, the reranker cost scales as $O(n \cdot c)$ because the cross-encoder must score c documents for *each* of the n rewrites independently, compared to $O(|\text{merged}|)$ in the merge-then-rerank paradigm where the reranker is invoked only once on the fused candidate list. This makes the RTM approach significantly more expensive at large n .

3.5.5. Noise Analysis and Oracle Upper Bounds

The oracle analyses provide empirical estimates of the remaining headroom available to practical fusion algorithms. If oracle performance continues to improve while practical fusion methods saturate, the limiting factor is likely the aggregation mechanism rather than candidate generation itself. Conversely, simultaneous saturation suggests that the retrieval frontier has already been exhausted.

The Oracle configuration measures the maximum effectiveness achievable from any individual rewrite, serving as an upper bound for rewrite-selection strategies. The BestRewrite baseline uses GPT-4o-mini ($\tau = 0$) to select, for each query, the single rewrite it judges most likely to retrieve relevant documents (the selection prompt is provided in Appendix A.2.2); this provides a practical estimate of how closely LLM-based rewrite selection can approximate the oracle upper bound. The Dedup Oracle evaluates whether retrieval quality continues to improve when the reranker is exposed to *all* unique documents retrieved across the rewrite set—unlike the standard fusion methods, which truncate the merged pool to a fixed cutoff of $c = 50$ documents. As the number of rewrites increases, the Dedup Oracle produces progressively larger candidate pools, providing insight into whether fusion limitations arise from imperfect aggregation or from insufficient candidate diversity.

Table 3.7: Oracle configurations used for rewrite-fusion analysis.

System	Retrieval Strategy	Purpose
Oracle	Evaluate each rewrite independently; select the highest-performing rewrite per query	Upper bound for rewrite-selection quality
BestRewrite	GPT-4o-mini ($\tau = 0$) evaluates all generated rewrites and selects the single rewrite it judges most likely to retrieve relevant documents (see Appendix A.2.2)	Measures how well LLM-based rewrite selection approximates oracle rewrite quality
Dedup Oracle	Merge all rewrite outputs, deduplicate, and rerank the full merged pool without truncation	Upper bound for candidate-pool expansion

3.6. Compute Proxy Definition

Since this thesis investigates test-time scaling, we define explicit compute proxies for each pipeline component:

- **Retrieval compute:** $C_{\text{ret}} = n \times c$, where n is the number of query variants dispatched and c is the per-query retrieval cutoff. For single-query experiments ($n = 1$), retrieval compute reduces to c .
- **Reranking compute:** $C_{\text{rerank}} = |D_{\text{CE}}|$, where $|D_{\text{CE}}|$ is the number of documents receiving real cross-encoder scores. In merge-then-rerank pipelines, $|D_{\text{CE}}| \leq |\text{merged}|$ (the deduplicated candidate pool). In rerank-then-merge pipelines, $|D_{\text{CE}}| = n \times c$. For adaptive methods (ORE, ReformIR), $|D_{\text{CE}}| = ce \times 16$ (batch size $\times$ number of batches).
- **Generation compute:** Fixed at $K \times T_{\text{max}}$, where $K = 5$ passages are concatenated and $T_{\text{max}} = 64$ new tokens are generated. This cost is constant across all retrieval configurations.

The total inference cost is therefore $C_{\text{total}} = C_{\text{ret}} + C_{\text{rerank}} + C_{\text{gen}}$. This decomposition enables direct comparison of scaling strategies that invest compute along different axes (breadth vs. depth vs. adaptive allocation).

3.7. Adaptive Retrieval Methods

To contextualise our horizontal multi-query fusion scaling, we explicitly evaluate two vertical, adaptive scaling methods that dynamically allocate cross-encoder compute based on corpus structure or query difficulty. Whereas the breadth scaling experiments (Sections 3.5 and 3.5.4) scale compute *horizontally* (more queries, more candidates), ORE and ReformIR scale compute *vertically* (deeper analysis of fewer candidates). Comparing both axes within a unified experimental framework directly addresses Research Gap 4 identified in Section 1.2.

3.7.1. ORE Adaptive Cutoff Sweep

The ORE framework scales retrieval quality by combining iterative graph-based candidate exploration with selective cross-encoder reranking under a constrained inference budget [19, 24]. ORE utilises a Learnt Affinity (LAFF) graph, constructed over a BM25 $k=128$ nearest-neighbour base graph and queried at $k=16$ neighbours at inference time. It iteratively expands the candidate frontier by scoring documents via a cross-encoder in an active-learning loop of batch size 16. A λ -regression model is fitted

at each iteration to predict relevance scores for unscored documents, enabling the system to rank the remaining candidates without exhaustive cross-encoder evaluation. ORE constitutes a representative example of adaptive inference-time scaling. Rather than allocating computation uniformly across all retrieved candidates, ORE dynamically concentrates expensive cross-encoder evaluations on regions of the candidate space predicted to contain relevant documents. This behaviour is conceptually analogous to adaptive computation strategies increasingly studied in large language model reasoning systems.

Corpus Graph Provenance. For 2WikiMultiHopQA, two corpus graphs were built locally over the full corpus (495K passages) using BM25 similarity: a $k=16$ base graph for ORE’s candidate exploration and a $k=128$ base graph serving as input for LAFF. Since the precomputed LAFF graph was not available for this dataset, the LAFF affinity scores are computed on-the-fly at inference time using the `macavaney/laff` fine-tuned transformer model [23], which computes learned document–document similarity scores via a sequence-classification head. The $k=128$ base graph is wrapped with the `pyterrier_adaptive.Laff` module and limited to $k=16$ neighbours via `to_limit_k(16)`. For MS MARCO (8.8M passages), all three graph artifacts—the $k=16$ base graph, the $k=128$ LAFF base graph, and the precomputed LAFF graph—were loaded from the `macavaney/msmarco-passage.corpusgraph` HuggingFace repository, constructed over the complete passage corpus. The precomputed LAFF graph is queried at $k=16$ at inference time. For DPR-w100 (21M passages), a single pre-built corpus graph from HuggingFace (`astappiev/ragwiki-corpusgraph`) was used as both the base graph and the LAFF base graph. As with 2WikiMultiHopQA, no precomputed LAFF graph was available for this dataset, so LAFF affinity scores are computed on-the-fly using the same `macavaney/laff` model.

The central scaling parameter is the cross-encoder budget (ce), which controls how many batches of 16 documents receive expensive real cross-encoder scores. In the original paper configuration, $ce = 7$ with a batch size of 16 was calibrated so that at $c = 100$, all candidate documents still received real scores.

The primary independent variable is the initial BM25 retrieval budget c , swept logarithmically from 10 to 1000. Two cross-encoder budget scaling modes are evaluated:

- **Linear Mode:** $ce = \lceil c/16 \rceil$. The budget scales proportionally with c such that every candidate receives a real cross-encoder score. In this mode, the λ -regression is still fitted at each iteration (as part of ORE’s standard loop), but its predictions are never used for the final ranking because all documents have already received real cross-encoder scores. This isolates the quality of ORE’s LAFF-based graph navigation by removing approximation effects.
- **Fixed Mode:** $ce = \min(7, \lceil c/16 \rceil)$. The budget is capped at the paper’s original value, forcing the system to estimate scores for the majority of documents at large budgets via the λ -regression model. For example, at $c = 1000$ only 11% of candidates receive real scores; the remaining 89% are ranked using regression predictions. This directly evaluates whether the approximation mechanism can extrapolate beyond the original calibration regime ($c = 100$).

The graph exploration parameters are scaled dynamically to maintain proportional coverage of the expanding candidate pool:

- $top_s = \max(25, \lfloor c/4 \rfloor)$: controls the number of highest-scoring documents retained as cluster heads for LAFF neighbour exploration. The floor of $c/4$ ensures that approximately 25% of the candidate pool anchors the graph walk, preventing under-exploration at large budgets while the minimum of 25 avoids degenerate navigation at small c .
- $top_s2 = \max(15, \lfloor c/6 \rfloor)$: controls the expansion frontier by determining how many high-scoring

Table 3.8: ORE scaling configurations for varying retrieval budgets (batch size = 16). Coverage denotes the proportion of the retrieval budget receiving real cross-encoder scores; values exceeding 100% arise because the total real calls ($ce \times 16$) can exceed c due to ceiling rounding of the batch count.

Mode	c	ce	Real CE Calls	Coverage	top_s	top_s2
Linear	100	7	112	112%	25	16
Linear	215	14	224	104%	53	35
Linear	500	32	512	102%	125	83
Linear	1000	63	1008	101%	250	166
Fixed	100	7	112	112%	25	16
Fixed	215	7	112	52%	53	35
Fixed	500	7	112	22%	125	83
Fixed	1000	7	112	11%	250	166

documents from the latest batch introduce new neighbouring candidates into the pool. The ratio $c/6$ is set to approximately two-thirds of top_s , balancing exploration breadth against redundant graph traversals.

3.7.2. ReformIR Cutoff Sweep

ReformIR scales retrieval effectiveness by dynamically combining cross-encoder reranking with iterative RM3-based query reformulation under a fixed inference budget [32]. Unlike ORE’s graph navigation, ReformIR uses iterative BM25 retrieval over RM3-expanded queries to discover new candidate documents. The primary scaling parameter is again the cross-encoder budget (ce), with an active-learning loop batch size of 16.

ReformIR receives $n = 5$ LLM-generated query reformulations as *initial seed queries*. These seed queries are used alongside the original query to bootstrap the first iteration of BM25 retrieval; in subsequent iterations, RM3 pseudo-relevance feedback generates additional reformulated queries dynamically based on the top-scoring documents from the previous iteration. The parameter $top_s = 15$ controls the number of *highest-scoring documents* selected from the scored pool as candidates for RM3 feedback. From these 15 candidates, the RM3 expander uses $fb_docs=5$ feedback documents to extract $fb_terms=10$ expansion terms with interpolation weight $\lambda_{RM3} = 0.3$. The value of top_s is fixed across all budgets because RM3 expansion quality plateaus quickly—empirically, feedback from more than ~ 15 candidate documents introduces noise from non-relevant passages without improving term selection. Whereas ORE expands the search frontier through graph traversal, ReformIR expands the retrieval process through iterative query reformulation. The method therefore provides an alternative form of adaptive test-time scaling in which additional computation is invested in generating improved retrieval trajectories rather than exploring neighbouring documents.

Like ORE, ReformIR is evaluated across the full $c \in \{10, \dots, 1000\}$ sweep using both **Linear** and **Fixed** cross-encoder scaling modes:

- **Linear Mode:** $ce = \lceil c/16 \rceil$. All candidates receive real scores; the `lsq_linear` regression is fitted but its predictions are not used for ranking. This isolates the contribution of RM3-based reformulation.
- **Fixed Mode:** $ce = \min(7, \lceil c/16 \rceil)$. At large c , the least-squares linear regression model must predict scores for unscored tail documents. The regression is trained on three document-level features paired with the corresponding cross-encoder score: (1) the BM25 retrieval score, (2) the

Table 3.9: ReformIR scaling configurations for varying retrieval budgets (batch size = 16). Coverage denotes the proportion of the retrieval budget receiving real cross-encoder scores; values exceeding 100% arise because the total real calls ($ce \times 16$) can exceed c due to ceiling rounding.

Mode	c	ce	Real CE Calls	Coverage
Linear	50	4	64	128%
Linear	100	7	112	112%
Linear	500	32	512	102%
Linear	1000	63	1008	101%
Fixed	50	4	64	128%
Fixed	100	7	112	112%
Fixed	500	7	112	22%
Fixed	1000	7	112	11%

LAFF affinity score from the corpus graph, and (3) the mean cross-encoder score of the document’s scored LAFF neighbours.

Comparing ReformIR to the sparse breadth experiment (Section 3.5) isolates the difference between dynamic RM3 expansion (vertical scaling) and static rank-fusion aggregation (horizontal scaling). The RM3 BM25 retrieval depth (`reform_bm25_num_results`) is set to $\max(128, c)$ to prevent the hardcoded-100 limitation in the original implementation from constraining candidate discovery at large budgets.

3.8. End-to-End RAG Generation

This experiment extends the retrieval-focused evaluations by measuring end-to-end RAG generation quality. The objective is to determine how the retrieval cutoff (c) and subsequent reranking affect the quality of final generated answers, mapping abstract retrieval metrics (Recall, nDCG) directly to downstream generative utility.

The pipeline retrieves c candidate passages using BM25. In the reranked configuration, all candidates are first scored by MiniLM-L6 and reordered before selecting the final passages. The top-5 passages are concatenated into a structured prompt and fed to a locally deployed, offline generator (meta-llama/Llama-3.2-3B-Instruct), which is instructed to produce concise answers of 1–5 words (maximum 2,000 new tokens; the generous limit avoids premature truncation while the system prompt constrains answer length).

The independent variable is the initial cutoff c , swept logarithmically from 10 to 1000. Because the reader model always processes exactly 5 passages regardless of the retrieval budget, generation latency remains constant across the sweep; only retrieval and reranking costs scale with c . Crucially, in the without-reranker configuration, the top-5 passages are always the highest-scoring BM25 results, which are identical for all $c \geq 10$; increasing c only adds lower-ranked passages that the reader never sees. Consequently, the without-reranker baseline is invariant to c , and all observed scaling effects are attributable to the reranker surfacing different passages into the top-5. MSMARCO is excluded from this experiment because the TREC DL 2019 queries lack gold-standard short answers suitable for Exact Match and F1 evaluation.

3.8.1. Generation Prompt and Decoding

The reader model uses the Llama-3.2-Instruct chat template with the following system and user prompt structure:

```

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a precise question-answering assistant. Answer using ONLY the provided passages.
Give a short answer (1-5 words). If you cannot find the answer, say 'unanswerable'. Do
not explain. Do not add extra text.

<|start_header_id|>user<|end_header_id|>

Passages:
- [passage 1 text]
- [passage 2 text]
...

Question: [query]

```

Decoding is performed with near-greedy search (`temperature = 0.1`, `do_sample = False`) and a maximum of 2,000 new tokens (although answers are constrained to 1–5 words by the system prompt, the generous token limit avoids premature truncation of the generation pipeline). The tokeniser uses left-padding, which is required for correct generation with decoder-only architectures in batched inference. No post-processing (lowercasing, trimming, punctuation stripping) is applied to the generated output before evaluation; normalisation is handled within each evaluation metric as described below.

3.8.2. Answer Evaluation Metrics

Generated answers are evaluated using three complementary metrics, each implemented via the `pyterrier_rag` and `ir_measures` frameworks:

Exact Match (EM). Binary agreement between the generated answer and the gold answer after normalisation. The `pyterrier_rag` implementation applies standard SQuAD-style normalisation: lowercasing, removal of articles (“a”, “an”, “the”), stripping of punctuation, and collapsing of whitespace. For multi-answer questions (DPR-w100/TriviaQA, where each query may have multiple valid gold answers), EM takes the *maximum* across all gold answer variants—i.e., the generated answer is marked correct if it exactly matches *any* of the accepted answers.

Token-level F1. Precision–recall harmonic mean computed over whitespace-tokenised word sets (after the same normalisation as EM). Let P be the set of tokens in the predicted answer and G the set of tokens in the gold answer; then:

$$F_1 = \frac{2 \cdot |P \cap G|}{|P| + |G|} \quad (3.6)$$

For multi-answer queries, the maximum F1 across all gold answer variants is reported.

LLM-as-a-Judge. A semantic evaluation metric implemented via a custom `ir_measures.define_byquery` measure. The judge uses the same Llama-3.2-3B-Instruct model as the reader (sharing the GPU backend to avoid out-of-memory errors from loading two model copies). The judge model is not used as a replacement for traditional answer-overlap metrics but rather as a complementary measure capable of recognising semantically equivalent answers that may differ lexically from the reference. Accordingly, conclusions are drawn primarily from the agreement between EM, F1, and judge scores rather than from any single metric in isolation. The judge receives a structured prompt using the Llama-3.2-Instruct chat template with the following components:

```

System: You are a strict factual evaluator. Compare the system answer to the gold reference.
Reply with ONLY a single integer from 1 to 10.

```

Scoring rules (strict):

- 1-2: Completely wrong (wrong type, refusal, irrelevant fact)
- 3-4: Correct type but factually incorrect
- 5-6: Partially correct (contains correct detail but misses main answer)
- 7-8: Mostly correct with minor inaccuracies
- 9-10: Exactly matches gold reference in all essential aspects

[5 few-shot examples; see Appendix A.3]

User: Query: [query]

Gold answer: [reference]

System answer: [prediction]

Score (1-10):

Decoding uses greedy search (`do_sample = False`, max 2,000 new tokens). The raw output is parsed via a cascade of regex patterns to extract an integer rating. Ratings below 1 or above 10 are clamped; unparseable outputs default to 0. For queries with multiple gold answers, the maximum rating across references is reported.

Potential Bias. Using the same model family for both generation and evaluation introduces a known self-preference bias [43]. We mitigate this partially by (1) using a strict, constrained prompt with an explicit scoring rubric and few-shot examples that reduce stylistic preference effects, and (2) reporting EM and F1 alongside the judge score so readers can triangulate across metric families. We acknowledge that a larger or external judge model would strengthen the evaluation.

Table 3.10: End-to-end RAG generation configuration.

Parameter	Value
Cutoff sweep (c)	{10, 16, 27, 46, 77, 129, 215, 359, 599, 1000}
Sweep spacing	geospace(10, 1000, 10)
Topic limit	100 queries
Retriever	BM25
Reranker	MiniLM-L6
Reader model	meta-llama/Llama-3.2-3B-Instruct (offline)
Top-K passages to reader	5
Maximum new tokens	2,000
Decoding	Near-Greedy ($T = 0.1$, no sampling)
Evaluation metrics	Exact Match, F1, LLM-as-a-Judge
Datasets	2WikiMultiHopQA, DPR-w100 (MSMARCO excluded)

3.9. Summary of Experimental Design

Across six experimental conditions, we systematically investigate the scaling behaviour of retrieval-augmented generation systems under increasing inference-time computation. The depth and breadth scaling experiments (Sections 3.4 and 3.5, including dense and RTM variants) evaluate fixed-compute scaling strategies through retrieval depth expansion and query broadening. The ORE and ReformIR experiments (Section 3.7) examine adaptive retrieval methods that dynamically allocate computation based on corpus structure and query difficulty. Finally, the end-to-end RAG experiment (Section 3.8) evaluates how retrieval improvements translate into downstream answer generation quality. Collectively, these experiments provide a comprehensive empirical framework for analysing test-time scaling laws in compound AI systems.

4

Results

The experiments presented in this chapter reveal that scaling test-time compute in multi-stage retrieval pipelines is neither monotonic nor universal: the returns depend jointly on retriever architecture, reranker calibration, corpus characteristics, and aggregation strategy. Our analysis emphasises scaling trajectories and performance patterns across configurations rather than individual absolute scores. We begin by examining how retrieval depth c affects reranker performance (Section 4.1), then investigate query breadth scaling through LLM-generated rewrites—comparing sparse and dense retrievers, two pipeline orderings, and an oracle upper-bound analysis (Sections 4.2.1–4.2.4). Section 4.3 synthesises cross-experiment comparisons, before Sections 4.4.1–4.4.3 evaluate adaptive methods and the full RAG pipeline. Throughout, we abbreviate 2WikiMultihopQA as “Wiki” and report percentage changes as *relative* improvements over the respective baseline unless stated otherwise.

4.1. Impact of Initial Retrieval Cutoff Depth

Following the experimental design described in Section 3.4 and Table 3.3, we swept the initial cutoff depth c across $\{10, 16, 27, 46, 77, 129, 215, 359, 599, 1000\}$ for six pipelines: BM25 baseline, BM25+MonoT5, BM25+MiniLM-L6, Dense (TCT-ColBERT) baseline, Dense+MonoT5, and Dense+MiniLM-L6. Performance is evaluated via nDCG@10 and Recall@10; full results, including R@50 and R@ c , are reported in the appendix. Across all three datasets—MS MARCO, TriviaQA, and 2WikiMultihopQA—two findings emerge consistently (Figure 4.1). First, the reranked pipelines improve with increasing c because a larger candidate pool increases the probability that relevant documents are available for promotion. Second, within a given first-stage retriever, the choice of reranker appears to exert a larger influence on final ranking quality than the retriever itself: Dense+MiniLM-L6 achieves the highest nDCG@10 on MS MARCO at all cutoffs, while on TriviaQA and 2WikiMultihopQA the MiniLM-L6 pipelines consistently match or exceed their MonoT5 counterparts at large c .

Dataset-conditioned scaling trajectories. On MS MARCO, the reranked BM25 pipelines exhibit a sustained climb in nDCG@10 across the entire sweep: BM25+MiniLM-L6 rises from 0.540 at $c=10$ to 0.735 at $c=1000$, and BM25+MonoT5 from 0.538 to 0.720. Dense+MiniLM-L6, by contrast, starts high (0.726 at $c=10$) and reaches a near-ceiling plateau of 0.748 by $c=77$, leaving limited room for further

improvement (Figure 4.1f). The divergence between the BM25 and Dense starting points reflects the substantially stronger initial retrieval quality of TCT-ColBERT; the subsequent convergence at large c indicates that, given a sufficiently deep candidate pool, BM25-based pipelines can approach—but in our experiments do not surpass—the dense-retrieval ceiling. 2WikiMultihopQA and TriviaQA present more constrained scaling environments. On 2WikiMultihopQA, the top-performing pipelines (BM25+MiniLM-L6 and Dense+MiniLM-L6) improve rapidly up to $c \approx 77$ –129 (Figure 4.1b), after which nDCG@10 stabilises around 0.408–0.411. On TriviaQA, MiniLM-L6 pipelines continue to benefit from deeper cutoffs—BM25+MiniLM-L6 reaches 0.391 at $c=1000$, and Dense+MiniLM-L6 reaches 0.398—but the absolute gains diminish beyond $c \approx 129$ (Figure 4.1d). The slower scaling on these corpora is consistent with their narrower relevance distributions: factoid and multi-hop queries tend to have fewer relevant passages in the corpus, so expanding the candidate pool yields diminishing returns once the reranker has already identified the most relevant documents.

Architectural divergence: MonoT5 vs. MiniLM-L6. The two rerankers exhibit markedly different scaling behaviours. MiniLM-L6 scales monotonically or plateaus across all datasets and retriever pairings: its nDCG@10 curves never degrade at large c , regardless of the first-stage retriever. MonoT5, however, is more sensitive to the composition of the candidate pool. On MS MARCO, Dense+MonoT5 peaks at $c=27$ (nDCG@10 = 0.737) and subsequently declines to 0.701 at $c=1000$, suggesting that the additional candidates introduced by deeper dense retrieval disrupt MonoT5’s scoring calibration. In contrast, BM25+MonoT5 on the same dataset scales positively throughout, reaching 0.720 at $c=1000$ —indicating that the nature of the candidate pool, not merely its size, determines whether MonoT5 benefits from deeper retrieval. On TriviaQA and 2WikiMultihopQA, MonoT5’s trajectories are noisier: BM25+MonoT5 oscillates between 0.307 and 0.361 across cutoffs on TriviaQA, while Dense+MonoT5 fluctuates between 0.325 and 0.373. On 2WikiMultihopQA, both MonoT5 pipelines remain largely flat (0.369–0.383), without the clear degradation observed in the Dense+MonoT5 configuration on MS MARCO. We hypothesise that this variability reflects MonoT5’s generative scoring mechanism, which assigns relevance probabilities through sequence-to-sequence prediction and may be more susceptible to distributional shifts in the candidate pool than MiniLM-L6’s representation-matching approach. These patterns indicate that the efficacy of scaling retrieval depth in multi-stage pipelines is jointly determined by the corpus’s relevance distribution and the reranker’s architectural robustness to noise. MiniLM-L6’s consistent scaling across all conditions makes it a more reliable component for test-time compute-scaling experiments, and for this reason, we adopt it as the default reranker in all subsequent experiments.

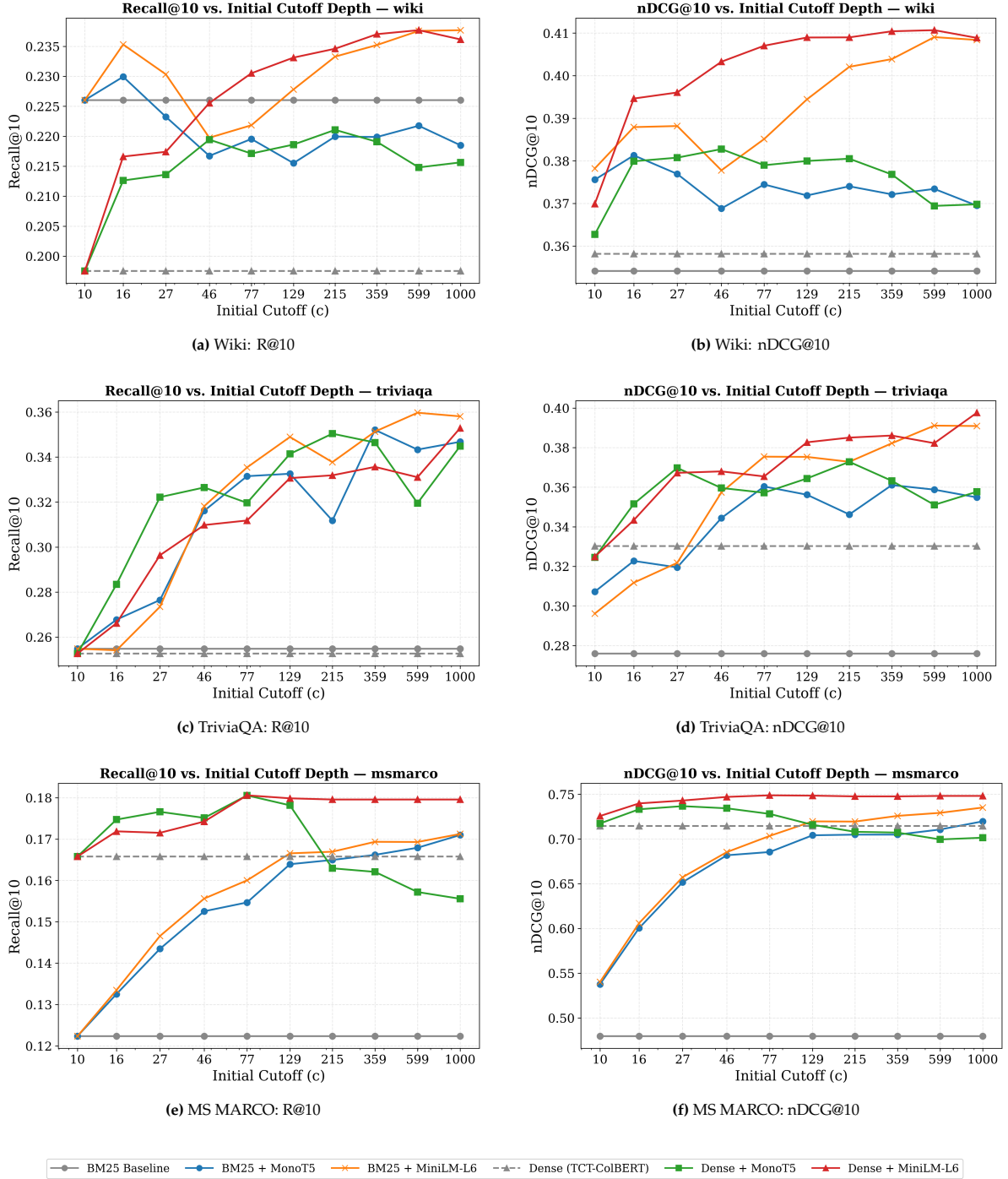


Figure 4.1: Effect of initial retrieval cutoff depth c on Recall@10 (left column) and nDCG@10 (right column) across all three evaluation corpora (Section 3.4, Table 3.3). Each subfigure compares six pipeline configurations: BM25 and Dense (TCT-ColBERT) baselines, each optionally followed by MiniLM-L6 or MonoT5 reranking. The x -axis is log-scaled, reflecting the logarithmic sweep $c \in \{10, \dots, 1000\}$. Dashed horizontal lines denote single-query baselines (constant across c).

4.2. Impact of Query Rewrite Breadth

4.2.1. Impact of Query Rewrite Breadth on Sparse Retrieval

Following the configuration described in Section 3.5 and Table 3.4, we investigate how the number of query rewrites n affects retrieval performance when combined with six fusion strategies (Section 3.5.2).

Based on the robustness established in Section 4.1, our primary analysis focuses on the MiniLM-L6 reranker; MonoT5 results are deferred to the appendix. A consistent pattern emerges across all three datasets: query rewrite fusion yields only marginal improvements in nDCG@10 and Recall@10. The effect on deep recall (Recall@50), however, is strongly dataset-dependent.

MS MARCO and 2WikiMultihopQA: limited gains, substantial risks. On both MS MARCO and 2WikiMultihopQA, the voting-based fusion methods (Borda, Weighted Borda, RRF, and Exp4Fuse) maintain R@50 within $\pm 2\%$ of the BM25+MiniLM-L6 baseline across the full range of rewrite depths. DedupCutoff and CombMAX, by contrast, cause substantial degradation: both methods reduce R@50 by approximately 20–21% on MS MARCO and 11–12% on 2WikiMultihopQA at their worst points. DedupCutoff truncates the merged candidate list to the top- c documents by retrieval score, discarding relevant passages that appear only at moderate ranks across multiple rewrites. CombMAX collapses all evidence for each document into a single maximum retrieval score, losing the rank-consensus signal that voting methods preserve. In both cases, the underlying failure mode is a loss of candidate diversity: because query rewrites produce highly overlapping result sets on these corpora, aggressive truncation or score collapse removes the long-tail candidates that contribute to deep recall. The stagnation of voting methods on these datasets suggests that, at the fixed retrieval depth $c=50$, a single well-formulated BM25 query already retrieves the majority of accessible relevant passages. Additional rewrites primarily introduce redundant candidate overlap rather than novel coverage.

TriviaQA: clear recall gains from rewrite diversity. TriviaQA presents a different pattern. While nDCG@10 fluctuates within a narrow band (0.430–0.448 for Borda, compared to the baseline of 0.438), R@50 exhibits a clear and sustained improvement. Borda achieves the largest gain, peaking at an R@50 of 0.589 at $n=13$ —a 6.3% relative increase over the BM25 baseline of 0.554. Weighted Borda achieves a comparable gain of 6.1% at the same rewrite depth. RRF and Exp4Fuse also improve over the baseline (+4.4% and +4.8% at $n=13$, respectively), though with smaller magnitudes. As with the other datasets, DedupCutoff and CombMAX underperform the baseline, confirming that indiscriminate candidate reduction is consistently harmful.

The larger recall gains on TriviaQA likely reflect the nature of factoid question answering: a single query captures only one lexical formulation of the target answer, whereas multiple rewrites collectively cover diverse phrasings present across relevant passages. Voting mechanisms exploit this reformulation diversity by promoting documents that achieve consensus across multiple retrieval streams. This structural advantage is less pronounced on MS MARCO, where query reformulations produce more overlapping result sets, and on 2WikiMultihopQA, where multi-hop queries already demand broad initial retrieval.

Summary. Voting-based fusion (Borda, Weighted Borda, RRF) consistently preserves or improves performance, while truncation-based methods (DedupCutoff, CombMAX) degrade deep recall across all corpora. A direct comparison with dense retrieval is presented in Section 4.3.1, and the effect of reversing the reranking order is analysed in Section 4.3.2.

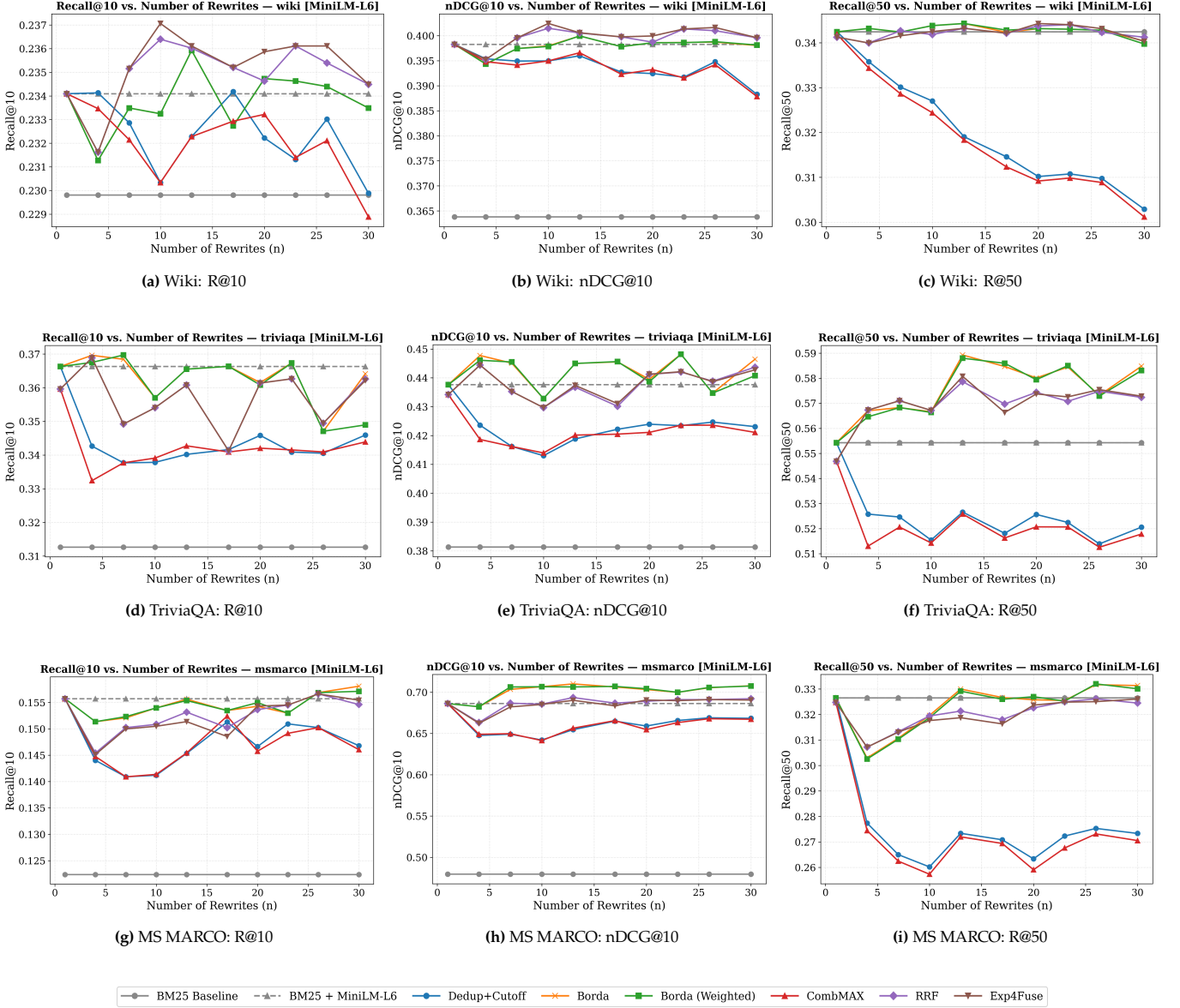


Figure 4.2: Effect of query rewrite breadth n on retrieval quality under the merge-then-rerank (MTR) paradigm with BM25 first-stage retrieval and MiniLM-L6 reranking (Section 3.5, Table 3.4). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Rows correspond to 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom). Six fusion methods are compared; see Section 3.5.2 for definitions. The dashed horizontal lines denote the BM25 and BM25+MiniLM-L6 baselines (constant across n).

4.2.2. Impact of Query Rewrite Breadth on Dense Retrieval

While Section 4.2.1 examined query rewrite fusion with sparse BM25 retrieval, the present experiment extends this analysis to dense retrieval using TCT-ColBERT as the first-stage retriever (Section 3.5.3, Table 3.5). We sweep $n \in \{1, 4, 7, 10, 13, 17, 20, 23, 26, 30\}$ and evaluate four fusion strategies: Borda, CombMAX, RRF, and Exp4Fuse, all followed by MiniLM-L6 reranking. DedupCutoff is omitted based on its consistent degradation in the sparse setting; BordaWeighted is omitted due to its functional redundancy with standard Borda. Due to computational constraints, this analysis is restricted to MS MARCO and 2WikiMultihopQA.

Dense retrieval benefits from rewrite fusion. A clear contrast with the sparse setting emerges: voting-based fusion applied to dense retrieval yields consistent improvements in R@50, whereas the same methods stagnated on BM25 for these two corpora. On MS MARCO, Borda R@50 rises from a baseline of 0.402 to 0.422 at $n=30$, a relative gain of approximately 4.9% (Figure 4.3f). RRF and Exp4Fuse follow a similar trajectory, reaching 0.420 and 0.419 at $n=30$, respectively. CombMAX, by contrast, degrades steadily beyond $n=10$, dropping to 0.387 at $n=30$ —a 3.7% decline from the baseline—confirming that max-score fusion remains harmful when query overlap is high.

On 2WikiMultihopQA, all four fusion methods improve over the baseline ($R@50 = 0.258$), though the absolute gains are smaller and the trajectories noisier (Figure 4.3c). Borda and RRF each reach approximately 0.270 at their peaks ($n=10$ and $n=7$, respectively), representing gains of 4.6–4.8%. Unlike on MS MARCO, CombMAX remains above the baseline on Wiki, ranging between 0.261 and 0.269, suggesting that the smaller corpus produces less problematic score distributions for max-based aggregation.

Across both datasets, nDCG@10 improves modestly: on MS MARCO, Borda and RRF increase nDCG@10 by 1–2% relative to the baseline, while on Wiki the improvement is approximately 2–3%. These gains indicate that dense retrieval fusion benefits not only deep recall but also, to a lesser extent, top- k ranking quality.

Comparison with sparse retrieval. The contrast between the sparse and dense breadth experiments (Sections 4.2.1 and 4.2.2) is instructive. On MS MARCO, sparse Borda achieved at most a 1.6% gain in R@50 over the BM25 baseline, whereas dense Borda yielded 4.9%. This difference may reflect the nature of the candidate pools produced by each retriever: dense retrieval encodes queries and documents in a continuous semantic space, so reformulated queries can activate different regions of the embedding space and surface documents that lexical matching misses. However, the denser first-stage retrieval also starts from a higher baseline ($R@50 = 0.402$ vs. 0.326 for BM25), which may provide a richer substrate for fusion to operate on. Disentangling these two factors would require controlled experiments with matched retrieval quality, which we leave for future work. A detailed cross-retriever comparison with relative gain trajectories is provided in Section 4.3.1.

Summary. Dense retrieval provides a more favourable substrate for query rewrite fusion than sparse retrieval, with voting-based methods yielding consistent R@50 gains of 4–5% while preserving nDCG@10. A detailed cross-retriever comparison with relative gain trajectories is provided in Section 4.3.1.

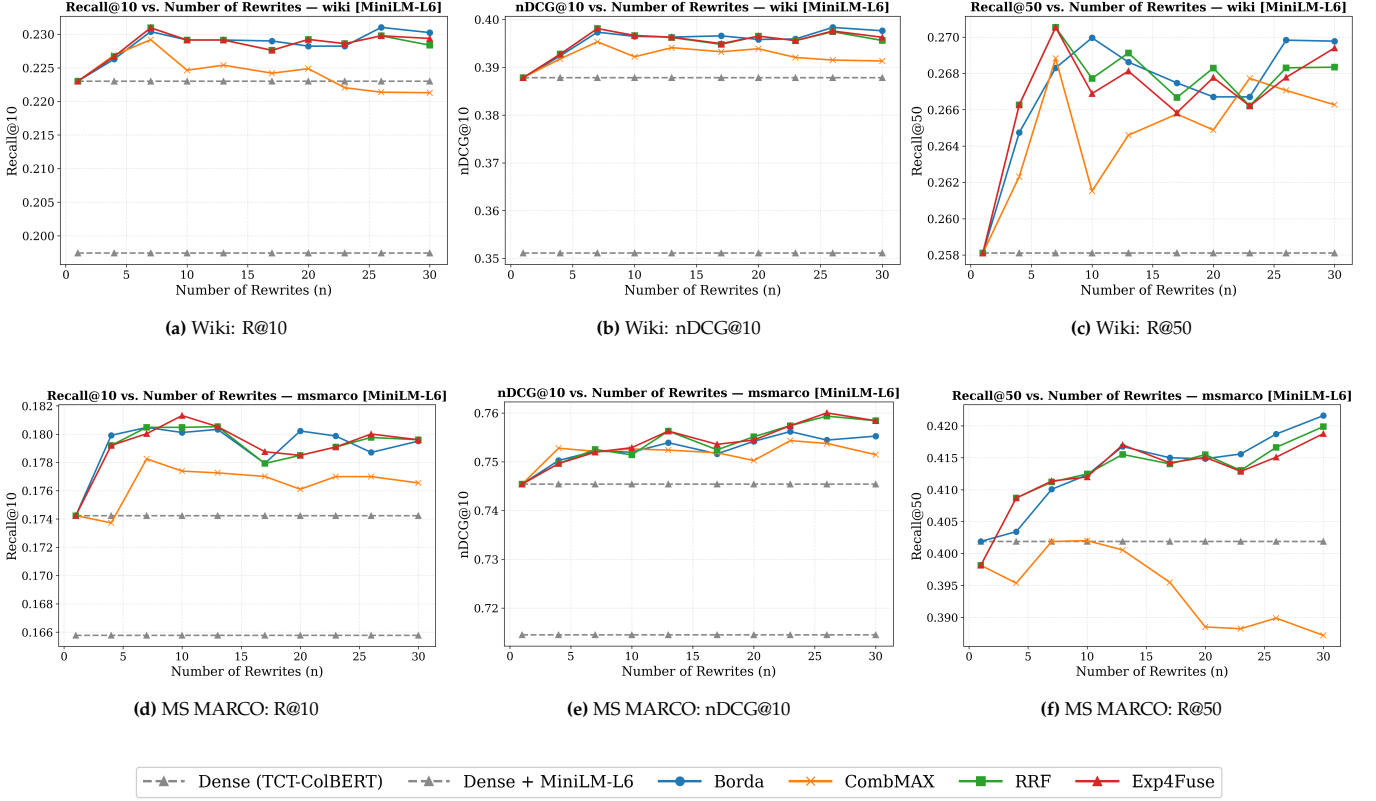


Figure 4.3: Effect of query rewrite breadth n on dense retrieval (TCT-ColBERT) with MiniLM-L6 reranking across 2WikiMultihopQA (top row) and MS MARCO (bottom row) (Section 3.5.3, Table 3.5). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Four fusion methods are evaluated: Borda, CombMAX, RRF, and Exp4Fuse. TriviaQA is omitted due to computational constraints.

4.2.3. Impact of Query Rewrite Breadth on Rerank-then-Merge

Having established the baseline performance of rank-based fusion methods in the merge-then-rerank (MTR) paradigm, we now examine the reverse ordering: rerank-then-merge (RTM), where each rewrite’s BM25 results are individually reranked by MiniLM-L6 before fusion (Section 3.5.4, Table 3.6). This design tests whether aggregating on the basis of the cross-encoder’s fine-grained relevance scores, rather than coarse BM25 ranks, yields superior final rankings. Six fusion algorithms are evaluated: DedupCutoff and CombMAX (which operate on reranker scores), Borda and BordaWeighted (rank-based voting), and RRF and Exp4Fuse (reciprocal rank scoring). The following analysis focuses on the four most representative methods; BordaWeighted tracks Borda closely across all datasets, and Exp4Fuse mirrors RRF.

Score-based dominance on MS MARCO. On MS MARCO, score-based aggregation demonstrates clear superiority over voting-based methods across all rewrite depths $n > 1$. At $n=30$, DedupCutoff achieves $R@50 = 0.398$, a 9% relative improvement over the best voting method, Borda (0.365), and a 22% improvement over the single-query BM25 baseline (0.327). CombMAX, which selects the maximum reranker score for each unique document across rewrites, performs comparably at $R@50 = 0.394$. Both score-based methods show broadly increasing performance as n grows, with a minor dip at $n=23$ before recovering at $n=30$. This advantage can be attributed in part to the fact that MiniLM-L6 was trained on MS MARCO, producing highly reliable relevance scores on in-domain queries. Taking the maximum score across rewrites effectively isolates the rewrite that best aligns a document’s content with the

query’s underlying intent.

Voting advantage on TriviaQA and Wiki. On TriviaQA and Wiki, voting-based strategies generally outperform their score-based counterparts, though the pattern admits exceptions. On TriviaQA at $n=30$, Borda achieves $R@50 = 0.606$, surpassing DedupCutoff (0.587) and CombMAX (0.585). However, at $n=10$ Borda briefly falls below both score-based methods, indicating that the voting advantage is not uniform across all rewrite depths.

On Wiki, Borda exceeds DedupCutoff at every non-trivial rewrite depth, reaching $R@50 = 0.349$ at $n=30$ compared to 0.346 for both DedupCutoff and CombMAX. However, these margins are small ($\Delta \approx 0.003$), and the overall effect sizes on Wiki are modest, with all fusion methods clustering within a narrow performance band relative to the BM25 baseline (0.342). Voting methods such as Borda require consensus across multiple rewrites, which may suppress isolated high-confidence but erroneous scores that CombMAX would otherwise propagate.

Saturation and diminishing returns. Contrary to the expectation of continuous improvement, recall metrics saturate and can degrade at large n . On Wiki, DedupCutoff peaks early at $n=4$ ($R@50 = 0.351$) before declining, while Borda peaks at $n=7$ (0.355). On TriviaQA, DedupCutoff peaks at $n=20$ (0.596) before declining to 0.587 at $n=30$. MS MARCO is the only dataset where performance continues to improve through $n=30$, albeit non-monotonically. This suggests that beyond a dataset-specific threshold, additional rewrites introduce more distractors than relevant candidates, and the fusion algorithm’s capacity to separate signal from noise becomes the binding constraint.

Implications for fusion design. These results reveal that the optimal aggregation basis depends on the interaction between the reranker’s scoring reliability and the corpus characteristics. Where the cross-encoder provides reliable scores—as on MS MARCO, the dataset on which MiniLM-L6 was trained—score-based aggregation efficiently isolates the best query-document alignments. On corpora where the reranker is more error-prone, voting mechanisms provide a consensus filter that suppresses outlier scores. This trade-off motivates the exploration of adaptive fusion strategies that can modulate their aggregation basis in response to observed score distributions. A direct comparison of RTM and MTR trajectories on the same axes is presented in Section 4.3.2.

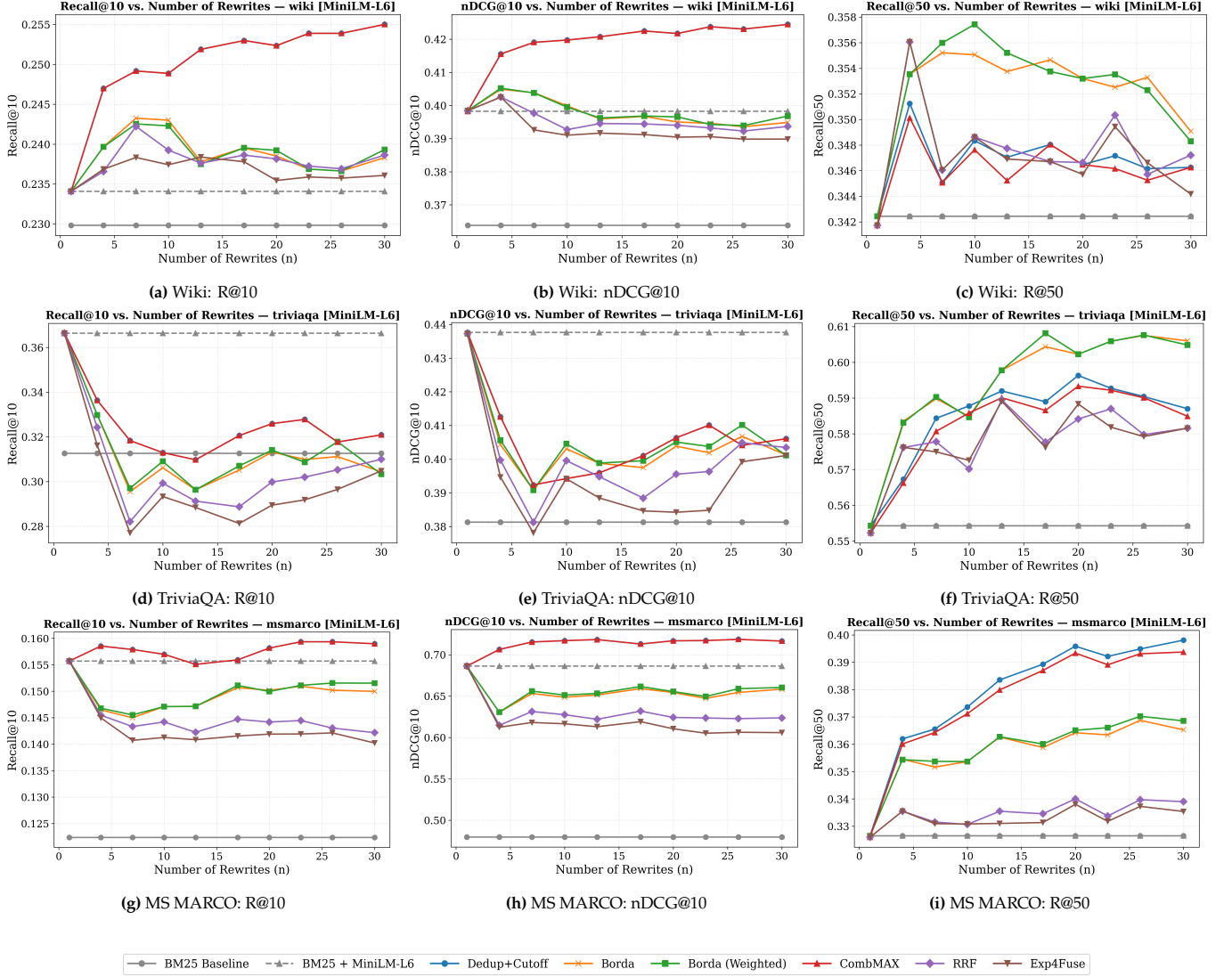


Figure 4.4: Effect of query rewrite breadth n on retrieval quality under the rerank-then-merge (RTM) paradigm with BM25 first-stage retrieval and MiniLM-L6 reranking (Section 3.5.4, Table 3.6). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Rows correspond to 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom). Six fusion methods are compared; see Section 3.5.2 for definitions.

4.2.4. Noise Analysis and Upper Bounds

Having established the performance trends of query rewrite fusion in sparse and dense settings, we now examine the headroom remaining for improved aggregation strategies. Following the oracle configurations described in Section 3.5.5 and Table 3.7, we introduce three analytical constructs. The ORACLE evaluates each rewrite independently and selects the highest-performing rewrite per query, providing an upper bound on rewrite selection quality. The DEDUP ORACLE merges all rewrite outputs, deduplicates the candidate pool, and reranks the full merged set *without* truncation to c documents, thereby establishing an upper bound for candidate-pool expansion. Finally, BESTREWRITE employs GPT-4o-mini to select the single best rewrite per query based solely on the original query and the generated alternatives, without access to retrieved documents or relevance feedback. We conduct this analysis across all three datasets using BM25 with MiniLM-L6 reranking.

Oracle upper bounds. A consistent pattern emerges across all datasets: the ORACLE substantially outperforms all fusion strategies, indicating that significant retrieval performance remains untapped by current aggregation methods. On MS MARCO, the ORACLE achieves an R@50 of 0.383 at $n=30$, compared to Borda’s 0.331 at the same depth—a gap of 0.051 (approximately 15% relative). On TriviaQA, the ORACLE reaches 0.676 versus Borda’s 0.585 at $n=30$, a gap of 0.091 (approximately 16% relative). On Wiki, the ORACLE attains 0.404 versus Borda’s 0.340, a difference of 0.060 (approximately 17% relative). These substantial gaps persist across Recall@10 and nDCG@10, confirming that current fusion methods are far from optimal: considerable room remains for more sophisticated selection or aggregation strategies.

Dedup Oracle: candidate-pool expansion. The DEDUP ORACLE provides complementary insights by removing the fixed cutoff constraint entirely. On MS MARCO, R@50 rises steadily from 0.326 at $n=1$ to 0.434 at $n=30$, substantially exceeding Borda’s 0.331 at every rewrite depth beyond $n=1$. This indicates that BM25 continues to surface new relevant passages as rewrites are added, but the standard fusion methods—which truncate the merged pool to $c=50$ documents—discard much of this complementary coverage. The gap between the DEDUP ORACLE and Borda widens with increasing n , growing from 0.063 at $n=4$ to 0.103 at $n=30$, indicating that the truncation bottleneck becomes more severe as more rewrites contribute unique candidates.

On Wiki, the DEDUP ORACLE shows only modest improvement: R@50 rises from 0.342 at $n=1$ to 0.348 at $n=30$, barely exceeding Borda’s 0.340–0.344 range. This limited headroom suggests that for multi-hop queries, the BM25 retrieval frontier is largely exhausted by a moderate number of rewrites, and expanding the candidate pool yields diminishing returns.

On TriviaQA, the DEDUP ORACLE improves R@50 substantially, reaching 0.621 at $n=30$ —well above Borda’s 0.585. However, the DEDUP ORACLE consistently falls *below* the ORACLE (0.676 at $n=30$), indicating that the reranker struggles to prioritise relevant documents when presented with the full, untruncated candidate pool (which grows to ~550 documents at $n=30$). The Oracle’s advantage arises because it evaluates each rewrite independently, allowing the reranker to operate on smaller, more focused candidate sets of $c=50$ documents where its scoring is better calibrated.

Precision–recall trade-offs. While the DEDUP ORACLE can improve deep recall (R@50), its top- k precision tells a different story. On TriviaQA, the DEDUP ORACLE’s R@10 decreases from 0.366 at $n=1$ to 0.338 at $n=10$, reflecting the noise introduced by expanding the candidate pool: the reranker’s scoring is diluted across hundreds of candidates, pushing marginally relevant documents into the top ranks at the expense of highly relevant ones. On MS MARCO and Wiki, the DEDUP ORACLE’s R@10 and nDCG@10 plateau or decline modestly at higher rewrite depths, confirming that candidate expansion benefits deep recall at the cost of top- k precision.

BestRewrite: LLM selection without retrieval feedback. The BESTREWRITE strategy, which uses GPT-4o-mini to select the single best rewrite per query without access to retrieval results, consistently underperforms both the ORACLE and Borda fusion. On MS MARCO, BESTREWRITE achieves an R@50 of 0.264 at $n=30$ —substantially below Borda’s 0.331 and below the single-query baseline of 0.326. This indicates that the LLM, lacking retrieval feedback, cannot reliably identify which rewrite will yield effective retrieval. We deliberately constrain BESTREWRITE to this “blind” selection setting to establish a lower bound; more sophisticated approaches that incorporate retrieval signals are explored in subsequent experiments.

Summary. The ORACLE demonstrates that 15–17% relative improvement in R@50 remains available beyond current aggregation methods, while the DEDUP ORACLE reveals that candidate-pool expansion benefits MS MARCO and TriviaQA but offers negligible gains on Wiki. The failure of BESTREWRITE confirms that effective rewrite selection requires retrieval feedback—a finding that motivates the adaptive methods explored in subsequent sections.

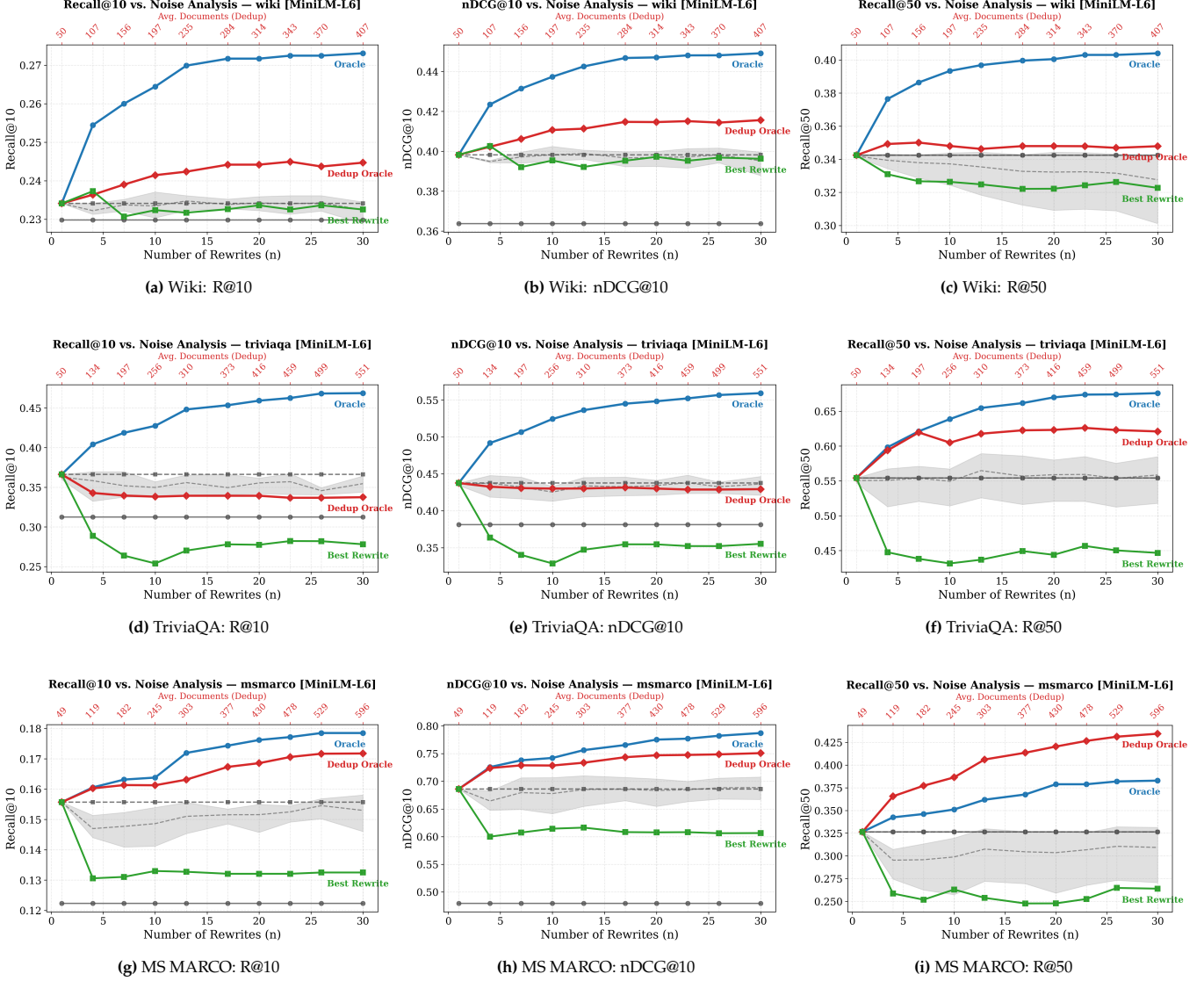


Figure 4.5: Oracle upper bounds and noise analysis for query rewrite fusion across 2WikiMultihopQA (top), TriviaQA (middle), and MS MARCO (bottom), using BM25 retrieval with MiniLM-L6 reranking (Section 3.5.5, Table 3.7). Columns show Recall@10 (left), nDCG@10 (middle), and Recall@50 (right). Bold labelled curves show three analytical constructs: Oracle (best single rewrite per query), Dedup Oracle (merged deduplicated pool without cutoff truncation), and BestRewrite (GPT-4o-mini-selected rewrite without retrieval feedback). The shaded grey region indicates the min-max range of the six practical fusion methods from Figure 4.2, with the dashed line showing their mean. The secondary y -axis (top, red) shows the average number of unique documents in the Dedup Oracle candidate pool at each rewrite depth n .

4.3. Comparing Results Across Scaling Strategies

4.3.1. Sparse vs Dense Retrieval Dynamics with respect to Query Rewrites

The sparse and dense breadth experiments (Sections 4.2.1 and 4.2.2) independently evaluated query rewrite fusion under sparse (BM25) and dense (TCT-ColBERT) first-stage retrieval, respectively. This

section synthesises those findings by directly overlaying the relative performance trajectories of both retriever families on the same axes. All comparisons use MiniLM-L6 reranking and are evaluated on MS MARCO and 2WikiMultihopQA (the two corpora shared between both experiments).

Deep recall (R@50): dense retrieval amplifies fusion gains. Figure 4.6 plots the relative change in R@50 as a function of rewrite count n . On MS MARCO (left panel), the four dense fusion curves (solid lines) rise monotonically and converge to approximately +3–5% relative gain by $n=30$. The sparse counterparts (dashed lines) exhibit markedly different behaviour: Borda and RRF oscillate near the zero line ($\pm 2\%$), while CombMAX(BM25) plunges to -21% before partially recovering. On 2WikiMultihopQA (right panel) a similar asymmetry is visible, though narrower in magnitude: dense Borda and RRF gain +3–5% by $n \approx 7$ and plateau, whereas sparse CombMAX steadily degrades to -10% at $n=30$.

These trajectories are summarised in absolute terms by the bar chart in Figure 4.6b. On MS MARCO, the best dense fusion (Borda at $n=30$) reaches $R@50 = 0.423$, exceeding the Dense+RR baseline (0.402) by 5.2%. By contrast, the best sparse fusion (Borda at $n=26$) achieves only 0.332, a 1.8% gain over BM25+RR (0.326). On 2WikiMultihopQA, the gap narrows—sparse fusion matches or slightly exceeds the BM25+RR baseline (0.342–0.344) while dense fusion peaks at 0.271 (RRF at $n=7$), a 5.0% gain over Dense+RR (0.258)—but the qualitative conclusion is unchanged: dense retrieval provides a more productive substrate for rewrite fusion.

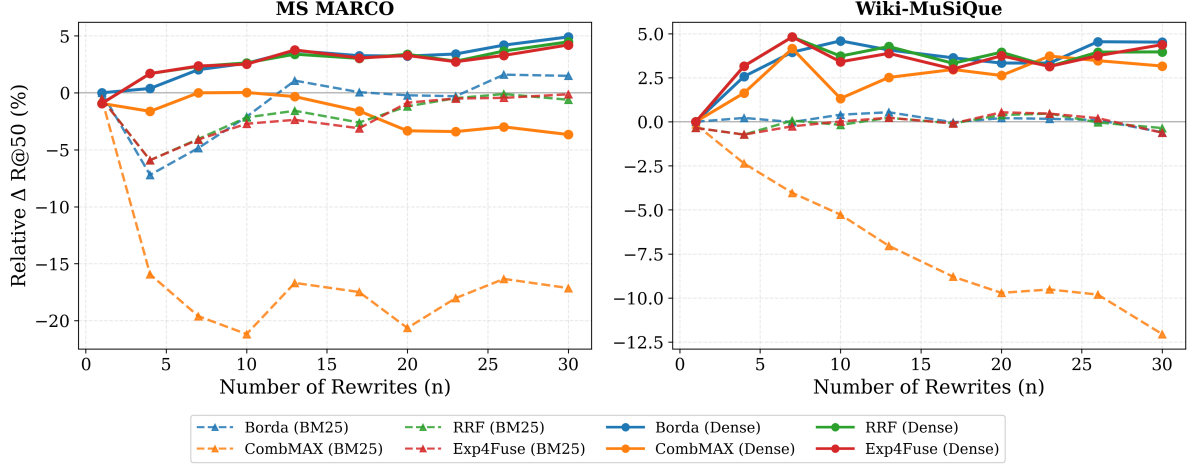
Ranking quality (nDCG@10): complementary patterns across retrievers. Figure 4.7 reveals that the nDCG@10 story differs from R@50. On MS MARCO, sparse Borda(BM25) achieves the single largest relative nDCG gain ($\approx +3\%$ at $n=13$), outperforming all dense curves. Dense fusion methods, while individually more stable, plateau at $\approx +1$ – 1.5% . On 2WikiMultihopQA, dense Borda and RRF sustain a +2–3% advantage through $n=30$, while sparse Borda drifts towards zero. CombMAX degrades nDCG on both retrievers and both corpora, consistent with its destructive score-collapse mechanism. The nDCG@10 bar chart (Figure 4.7b) confirms that peak nDCG values for sparse and dense are comparable on MS MARCO (sparse Exp4Fuse: 0.760 vs. dense RRF: 0.756), whereas on 2WikiMultihopQA both retrievers cluster tightly around 0.40.

Shallow recall (R@10): dense retrieval sustains early gains. The R@10 comparison (Figure 4.8) reveals an additional distinction. On MS MARCO, dense Borda, RRF, and Exp4Fuse reach +3–4% relative R@10 gain by $n \approx 7$ and sustain it through $n=30$. Sparse fusion methods, by contrast, degrade R@10: Borda(BM25) fluctuates between -1% and -2.5% , RRF(BM25) drops to -3% , and CombMAX(BM25) falls to -9% . On 2WikiMultihopQA, dense RRF and Borda lead with +2–3.5%, while sparse methods remain close to zero. The bar chart (Figure 4.8b) confirms that on MS MARCO dense fusion yields peak R@10 values of 0.178–0.181, comfortably exceeding the Dense+RR baseline (0.174), whereas sparse methods cluster near the BM25+RR baseline (0.156–0.157).

Interpretation. Two factors explain the differential response to rewrite fusion. First, BM25’s reliance on exact lexical matching means that reformulated queries largely retrieve the same passages when the vocabulary overlap between rewrites is high—a condition common on MS MARCO and Wiki. Dense retrieval encodes queries in a continuous semantic space where even small paraphrase differences can activate distinct embedding regions, yielding more diverse candidate pools that voting-based fusion can exploit. Second, the initial retrieval quality of the dense retriever provides a richer pool of near-relevant passages that reranking can promote, whereas BM25’s sparser candidate lists offer less headroom for

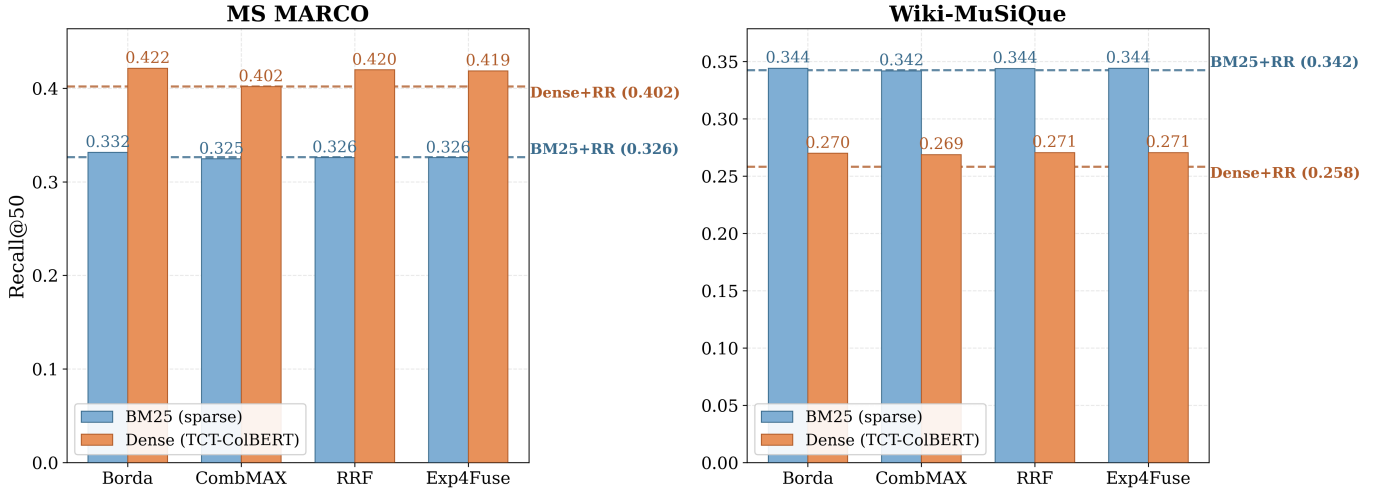
rank aggregation to improve upon. These findings reinforce the recommendation from Section 4.2.2: when deploying query rewrite fusion as a test-time scaling strategy, pairing it with a dense first-stage retriever substantially increases the achievable recall gains.

Sparse vs Dense: Relative Recall@50 Gain from Query Rewrite Fusion



(a) Relative change in R@50 from reranked baseline

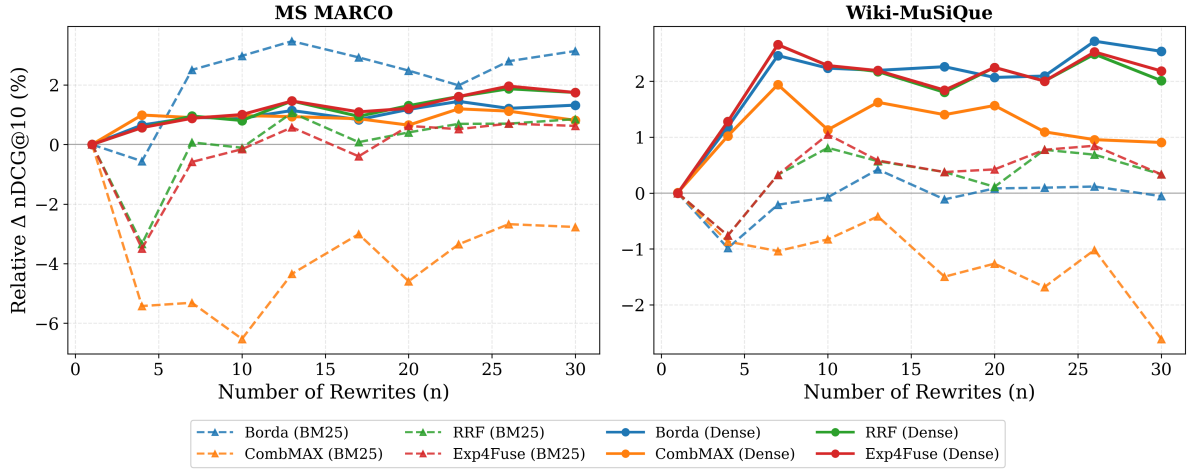
Peak Fusion Recall@50 vs Baselines — Sparse vs Dense



(b) Peak R@50 absolute values

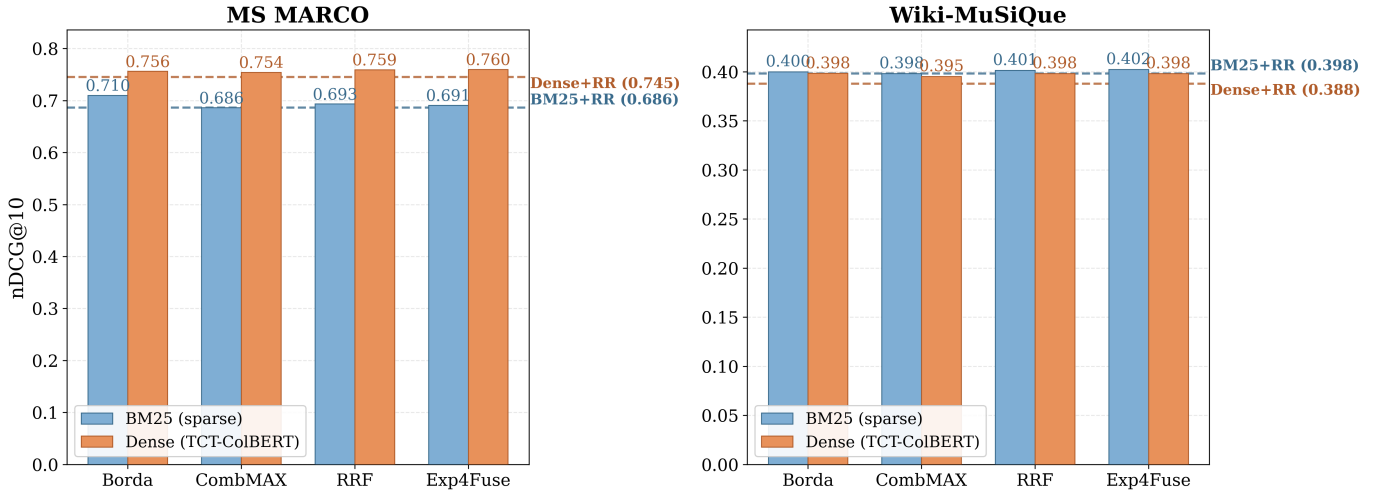
Figure 4.6: Recall@50 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking on MS MARCO and 2WikiMultihopQA, across four fusion methods. The upper subfigure shows the relative change from the reranked baseline as a function of the number of rewrites n . The lower subfigure shows peak absolute R@50 values and the n at which they are achieved.

Sparse vs Dense: Relative nDCG@10 Gain from Query Rewrite Fusion



(a) Relative change in nDCG@10 from reranked baseline

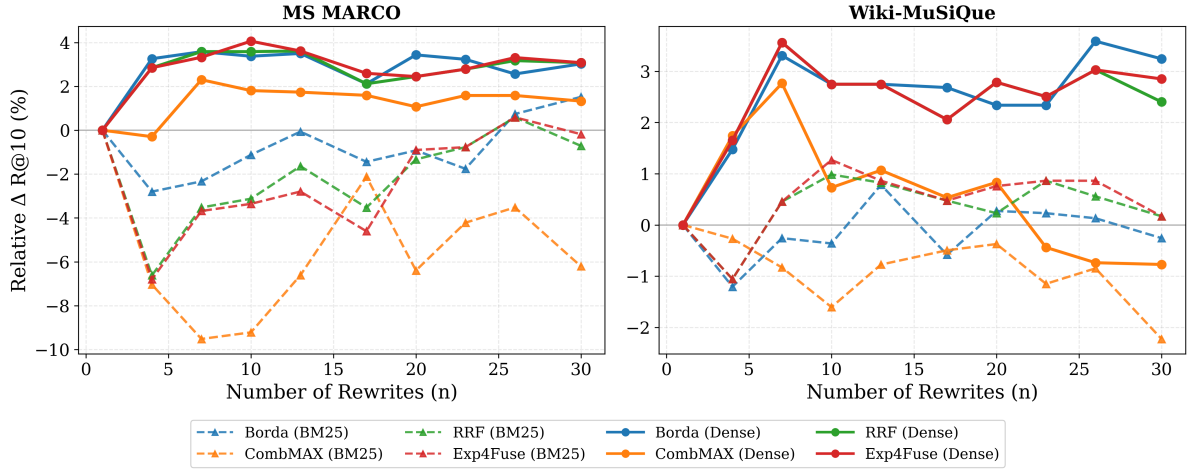
Peak Fusion nDCG@10 vs Baselines — Sparse vs Dense



(b) Peak nDCG@10 absolute values

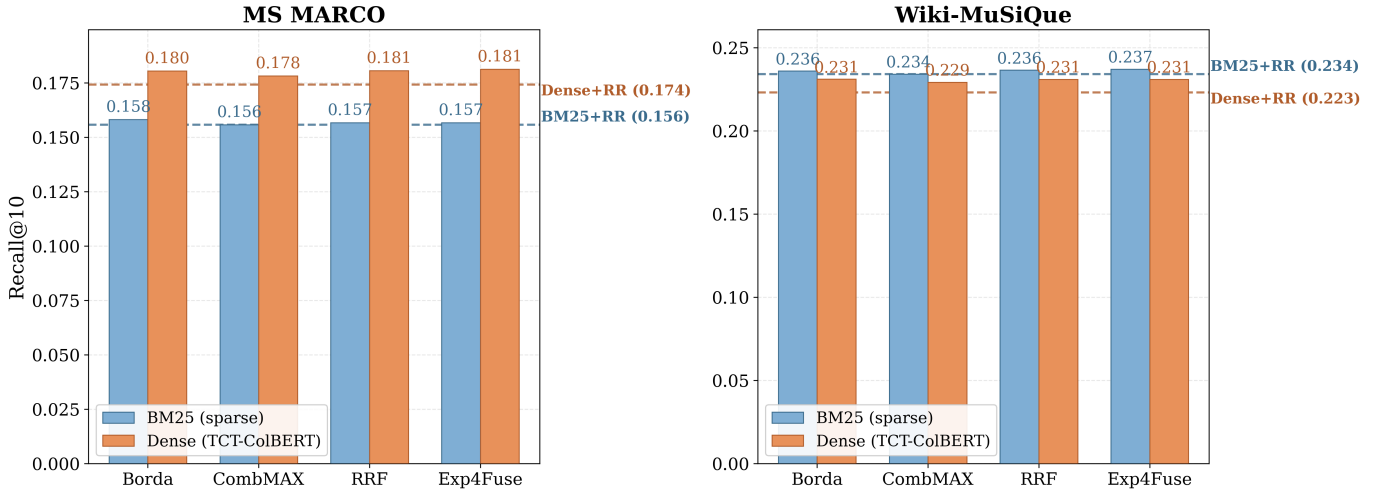
Figure 4.7: nDCG@10 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking on MS MARCO and 2WikiMultihopQA. The upper subfigure shows the relative change from the baseline for varying rewrite counts n . The lower subfigure displays peak absolute nDCG@10 values, including the number of rewrites n used.

Sparse vs Dense: Relative Recall@10 Gain from Query Rewrite Fusion



(a) Relative change in R@10 from reranked baseline

Peak Fusion Recall@10 vs Baselines — Sparse vs Dense



(b) Peak R@10 absolute values

Figure 4.8: Recall@10 results comparing sparse BM25 and dense TCT-ColBERT retrieval with MiniLM-L6 reranking. The upper subfigure illustrates the relative change in R@10 as a function of the rewrite count n . The lower subfigure shows the peak absolute R@10 values achieved for each retrieval-fusion combination.

4.3.2. RTM vs MTR Retrieval Dynamics

The sparse and RTM breadth experiments (Sections 4.2.1 and 4.2.3) evaluated the same set of fusion algorithms under two opposing pipeline orderings: merge-then-rerank (MTR, Section 4.2.1) and rerank-then-merge (RTM, Section 4.2.3). In the MTR paradigm, BM25 results from all rewrites are first fused and then reranked by MiniLM-L6. In RTM, each rewrite’s BM25 results are individually reranked by MiniLM-L6 before fusion operates on the reranked lists. This section directly overlays the relative performance trajectories of both orderings to assess which pipeline structure is more effective, and under what conditions.

Deep recall (R@50): RTM dominates on MS MARCO. Figure 4.9 reveals a stark divergence between the two paradigms on MS MARCO. The RTM variants (dashed lines) climb sharply: DedupCutoff(RTM) and CombMAX(RTM) reach approximately +20% relative R@50 gain by $n=20-30$, while Borda(RTM) and BordaWeighted(RTM) plateau at +10–13%. The MTR counterparts (solid lines) stagnate near zero for voting methods and plunge to –15% to –21% for CombMAX(MTR) and DedupCutoff(MTR). This pattern is confirmed by the bar chart (Figure 4.9b): RTM DedupCutoff peaks at $R@50 = 0.398$ ($n=30$), a 22% relative gain over the BM25+RR baseline (0.326), whereas MTR DedupCutoff achieves only 0.338. The RTM advantage on MS MARCO reflects the fact that MiniLM-L6 was trained on this corpus: per-rewrite reranking produces well-calibrated relevance scores, enabling score-based fusion methods (DedupCutoff, CombMAX) to isolate the best query–document alignments before merging.

On TriviaQA, RTM voting methods (Borda, BordaWeighted) outperform their MTR counterparts at $n=30$, reaching $R@50 \approx 0.60-0.61$ compared to MTR’s 0.58–0.59. However, the trajectories differ: RTM methods start lower and climb more steeply, while MTR methods rise earlier and more steadily. Score-based RTM methods (CombMAX, DedupCutoff) underperform RTM voting at most rewrite depths. On 2WikiMultihopQA, all methods cluster within a narrow $\pm 4\%$ band, with RTM BordaWeighted showing a marginal early advantage (+4.4% at $n=10$) before other methods converge.

Ranking quality (nDCG@10): RTM trades precision for recall. Figure 4.10 exposes the cost of RTM’s recall gains. On MS MARCO, RTM score-based methods (DedupCutoff and CombMAX) preserve and improve nDCG@10, both sustaining +3–5% gains. However, RTM voting methods severely degrade ranking quality: Borda(RTM) drops to –5% to –8%, RRF(RTM) to –8% to –10%, and Exp4Fuse(RTM) to –10% to –12%. BordaWeighted(RTM), despite its name, also degrades nDCG@10 by –4% to –8%. MTR methods, by contrast, maintain nDCG near or above the baseline—Borda(MTR) sustains approximately +3% throughout. The bar chart confirms that MTR Borda peaks at $nDCG@10 = 0.710$ ($n=13$), exceeding RTM Borda’s peak of 0.686.

On TriviaQA, nDCG@10 degrades more severely under RTM across all methods. CombMAX(RTM) drops to –10%, Borda(RTM) to –11%, RRF(RTM) to –13%, and Exp4Fuse(RTM) to –14%, while MTR methods remain within $\pm 2\%$ of the baseline. This inversion—RTM gains recall but loses precision—is most pronounced on out-of-domain corpora where MiniLM-L6’s per-rewrite scores are less reliable, causing both score-based and voting-based RTM fusion to propagate noisy relevance estimates.

On 2WikiMultihopQA, RTM DedupCutoff achieves the best nDCG@10 gains (+6–7%), consistent with its R@50 advantage. MTR methods cluster near zero. This suggests that on multi-hop queries with shorter relevant passage lists, RTM’s per-rewrite reranking helps identify the small number of relevant documents more precisely.

Shallow recall (R@10): both paradigms degrade on MS MARCO and TriviaQA. The R@10 comparison (Figure 4.11) shows that neither paradigm consistently improves top-10 recall. On MS MARCO, most MTR methods degrade R@10 by 2–8%. Among RTM methods, DedupCutoff and CombMAX achieve marginal gains (+2%), while RTM voting methods also degrade; Exp4Fuse(RTM) shows the largest drops (–9% to –10%). On TriviaQA, the degradation is more severe: Borda(RTM) reaches –19%, CombMAX(RTM) –15%, and most methods fall between –5% and –15%. MTR Borda is the most resilient, staying near zero throughout.

2WikiMultihopQA again stands apart. RTM DedupCutoff climbs to +9% R@10 gain, and Borda(RTM) reaches +4%. MTR methods achieve modest gains of 0–1%. This dataset-specific advantage aligns with

the earlier observation that multi-hop queries benefit from per-rewrite reranking, which can surface different hops’ evidence independently before fusion aggregates them.

Interpretation. The RTM–MTR comparison reveals a fundamental trade-off mediated by the reranker’s domain calibration. On MS MARCO (in-domain), RTM score-based methods (DedupCutoff, CombMAX) unlock substantial R@50 gains (+20–22%) because MiniLM-L6 produces reliable per-rewrite relevance scores that these methods can exploit. RTM voting methods also gain recall (+10–13%) but at the cost of nDCG degradation, since rank-based aggregation discards the calibrated scores. On TriviaQA (out-of-domain), all RTM methods degrade nDCG@10, even those that gain R@50, because the reranker’s per-rewrite scores become noisy. MTR sidesteps this issue by deferring reranking until after fusion, allowing the reranker to operate on a single, fused candidate list rather than n independent ones. The practical recommendation is two-fold: (i) when the reranker is well-calibrated for the target corpus, RTM with score-based fusion (DedupCutoff) is the superior strategy for deep recall; (ii) when the reranker is applied out-of-domain, MTR with voting-based fusion (Borda) preserves ranking quality and offers more stable, if smaller, recall gains. These complementary strengths motivate the adaptive strategies explored in Section 4.4.

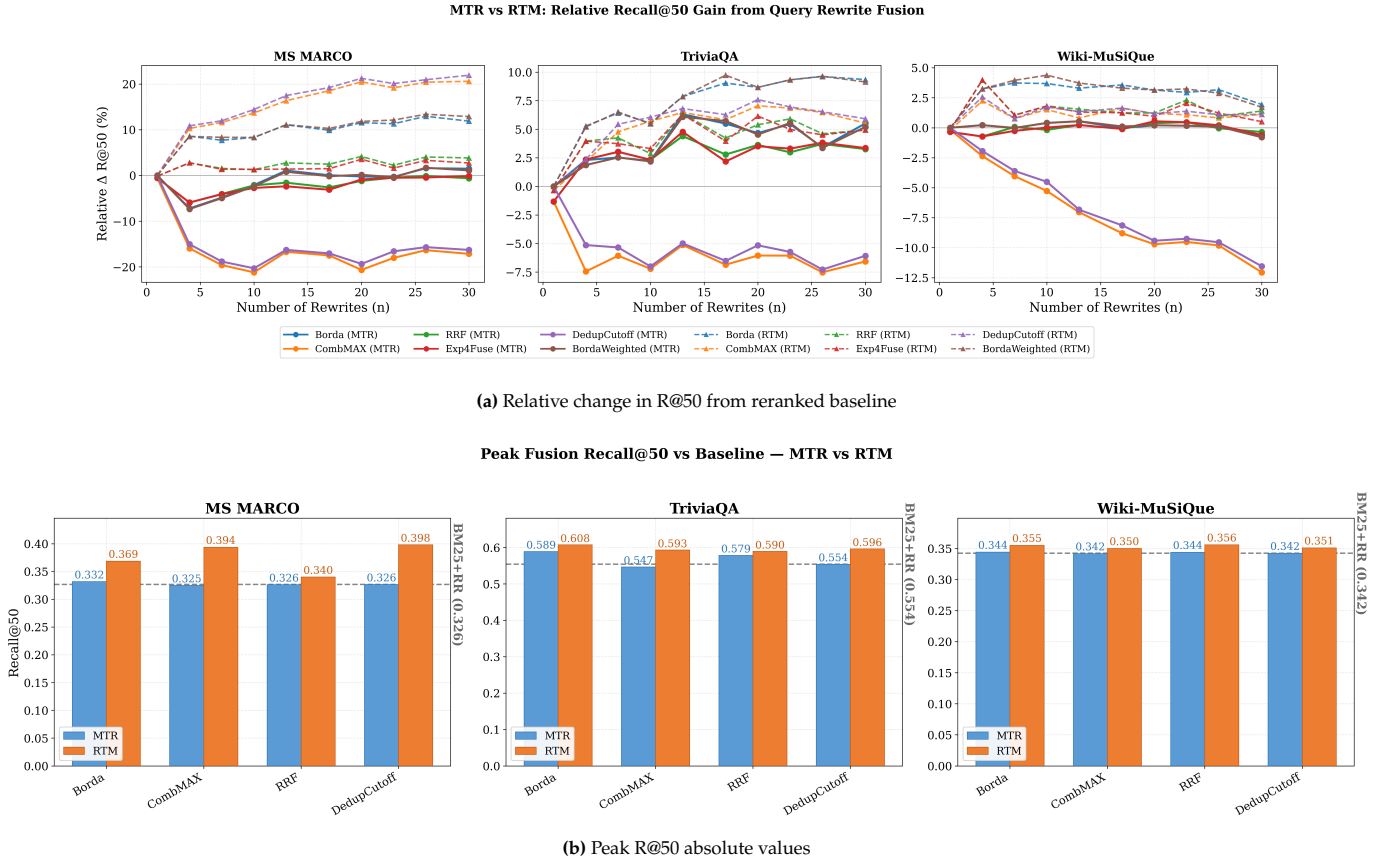
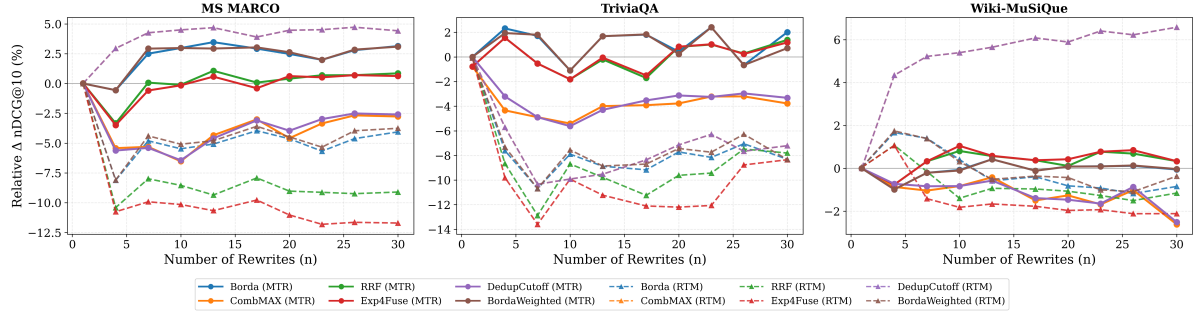


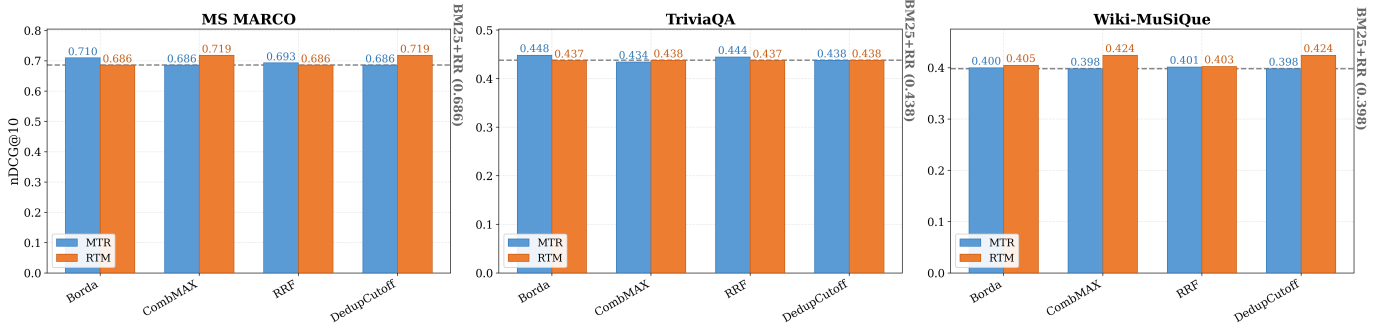
Figure 4.9: Recall@50 results comparing MTR (merge-then-rerank, Section 4.2.1) and RTM (rerank-then-merge, Section 4.2.3) pipeline orderings with BM25 retrieval and MiniLM-L6 reranking across all three corpora. The upper subfigure shows the relative change from the BM25+RR baseline as a function of rewrite count n . Solid lines denote MTR; dashed lines denote RTM. The lower subfigure shows peak absolute R@50 values and the n at which each was achieved. RTM score-based methods (DedupCutoff, CombMAX) achieve the largest gains on MS MARCO, while MTR voting methods (Borda) are more stable on TriviaQA and 2WikiMultihopQA.

MTR vs RTM: Relative nDCG@10 Gain from Query Rewrite Fusion



(a) Relative change in nDCG@10 from reranked baseline

Peak Fusion nDCG@10 vs Baseline — MTR vs RTM



(b) Peak nDCG@10 absolute values

Figure 4.10: nDCG@10 results comparing MTR and RTM pipeline orderings across all three corpora. On MS MARCO (left), RTM score-based methods (DedupCutoff, CombMAX) preserve nDCG@10 (+3–5%), while RTM voting methods degrade by –5% to –12%. On TriviaQA (middle), all RTM methods degrade nDCG@10, with Exp4Fuse reaching –14%. On 2WikiMultihopQA (right), RTM DedupCutoff achieves the best nDCG@10 gains (+6–7%). Layout as in Figure 4.9.

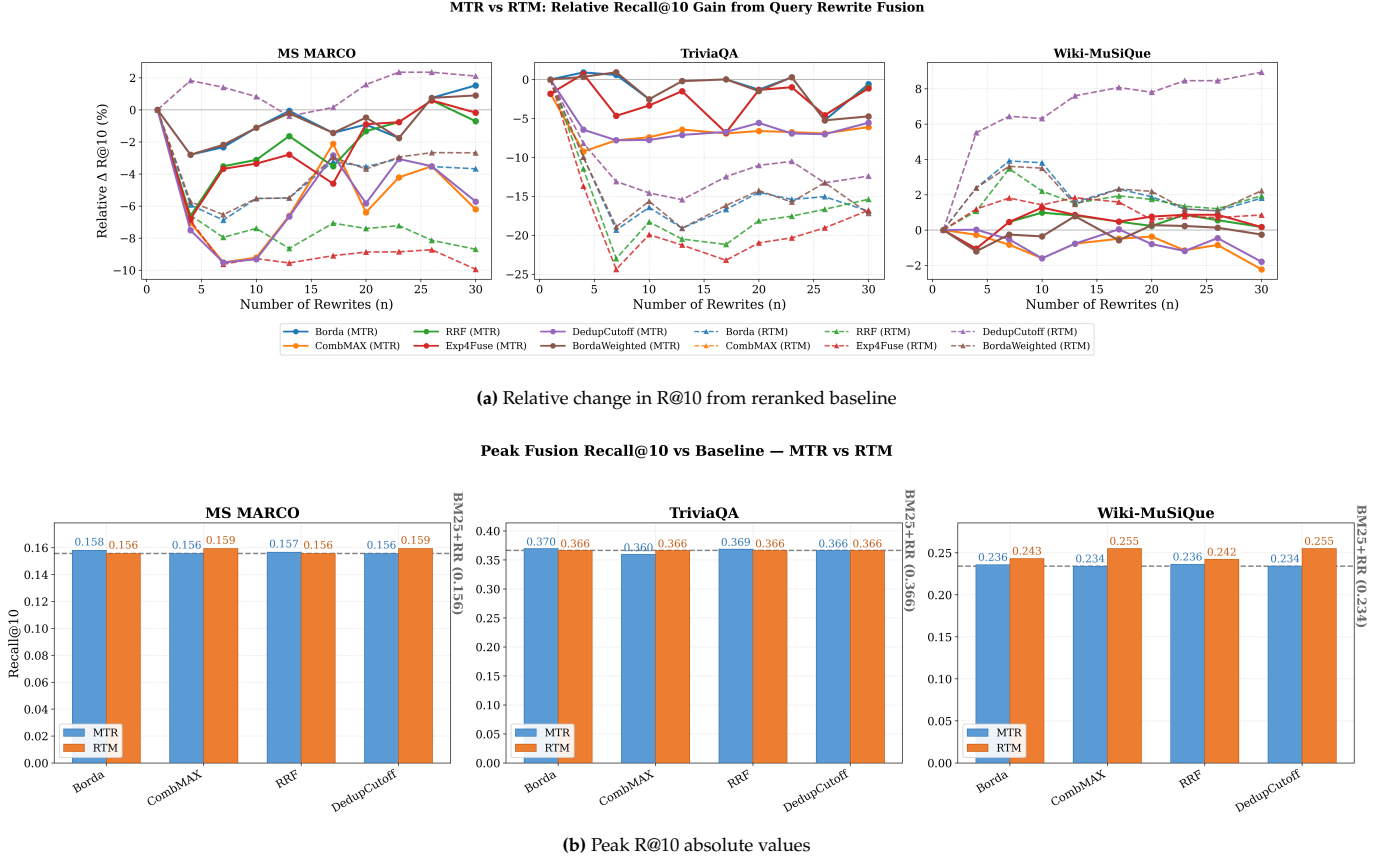


Figure 4.11: Recall@10 results comparing MTR and RTM pipeline orderings with BM25 retrieval and MiniLM-L6 reranking. On MS MARCO (left), DedupCutoff and CombMAX achieve marginal RTM gains (+2%), while Exp4Fuse(RTM) drops −9% to −10%. On TriviaQA (middle), Borda(RTM) reaches −19% and CombMAX(RTM) −15%. On 2WikiMultihopQA (right), RTM DedupCutoff gains +9% R@10, outperforming all MTR methods. Layout as in Figure 4.9.

4.4. Adaptive Retrieval Methods

4.4.1. ORE

Having established the scaling behaviour of static retrieval budgets with reranking in Section 4.1, we now examine whether adaptive inference-time strategies exhibit similar scaling properties. We evaluate the Online Relevance Estimation (ORE) framework (Section 3.7.1), which uses an active-learning loop and λ -regression to concentrate cross-encoder evaluations on promising regions of a Learnt Affinity (LAFF) graph. The initial retrieval budget c is swept logarithmically from 10 to 1000 under two cross-encoder constraints: **Linear Mode** (proportional scaling, isolating graph navigation quality) and **Fixed Mode** (capped at $ce=7$, testing the regression model’s extrapolation capacity).

Empirical scaling patterns. A divergence in scaling behaviour emerges across corpora (Figure 4.12). On MS MARCO—the only dataset with a fully pre-computed LAFF graph—both ORE modes scale progressively with c . The Linear Mode’s nDCG@10 rises from 0.539 at $c=10$ to 0.734 at $c=1000$, closely tracking the BM25+MiniLM-L6 reranker baseline (0.735 at $c=1000$). The Fixed Mode follows a less smooth trajectory: after tracking the Linear Mode through $c=77$ (nDCG@10 = 0.715), it drops to 0.682 at $c=129$ before recovering to 0.710 at $c=1000$. The non-monotonic behaviour at $c=129$ is consistent with instability in the λ -regression model in the transition region where real cross-encoder coverage falls from 100% to 87%. Despite this instability, the Fixed Mode maintains approximately 96–97% of the

Linear Mode’s performance across all evaluated budgets.

On TriviaQA, the Linear and Fixed modes produce *identical* results at every cutoff depth, indicating that the on-the-fly LAFF computation effectively neutralises any difference in cross-encoder budget allocation. ORE’s nDCG@10 improves from 0.305 at $c=10$ to 0.391 at $c=1000$, slightly exceeding the reranker baseline at most cutoff depths (by 0.003–0.018), though the margins are small enough to be within experimental noise.

Wiki exhibits the most constrained scaling. Both ORE modes produce nearly identical results, with nDCG@10 ranging from 0.378 at $c=10$ to 0.396–0.397 at $c=1000$. Neither mode surpasses the reranker baseline (0.408 at $c=1000$), suggesting that ORE’s graph exploration provides limited additional signal on Wiki’s multi-hop query distribution.

Structural constraints on cross-domain generalisation. The cross-dataset variance in ORE’s scaling response is likely rooted in the provenance of the LAFF graphs. MS MARCO used a comprehensively pre-computed LAFF graph spanning 8.8M passages, whereas Wiki and TriviaQA relied on on-the-fly LAFF computation. This infrastructural disparity may constrain the active-learning loop on the latter datasets, limiting the system’s capacity to navigate toward novel semantic neighbourhoods. The identical behaviour of Linear and Fixed modes on TriviaQA further supports this interpretation: when LAFF quality is limited, increasing the cross-encoder budget yields no additional benefit.

Implications for adaptive scaling. On MS MARCO, the data provides preliminary support for adaptive inference-time scaling: graph-based candidate exploration, when operating over a fully pre-computed corpus topology, can achieve ranking quality comparable to the static reranker baseline. However, this effect is observed on a single dataset with pre-computed LAFF graphs. Validation across additional corpora with fully pre-computed graphs is necessary to establish whether adaptive inference-time scaling generalises beyond the MS MARCO setting.

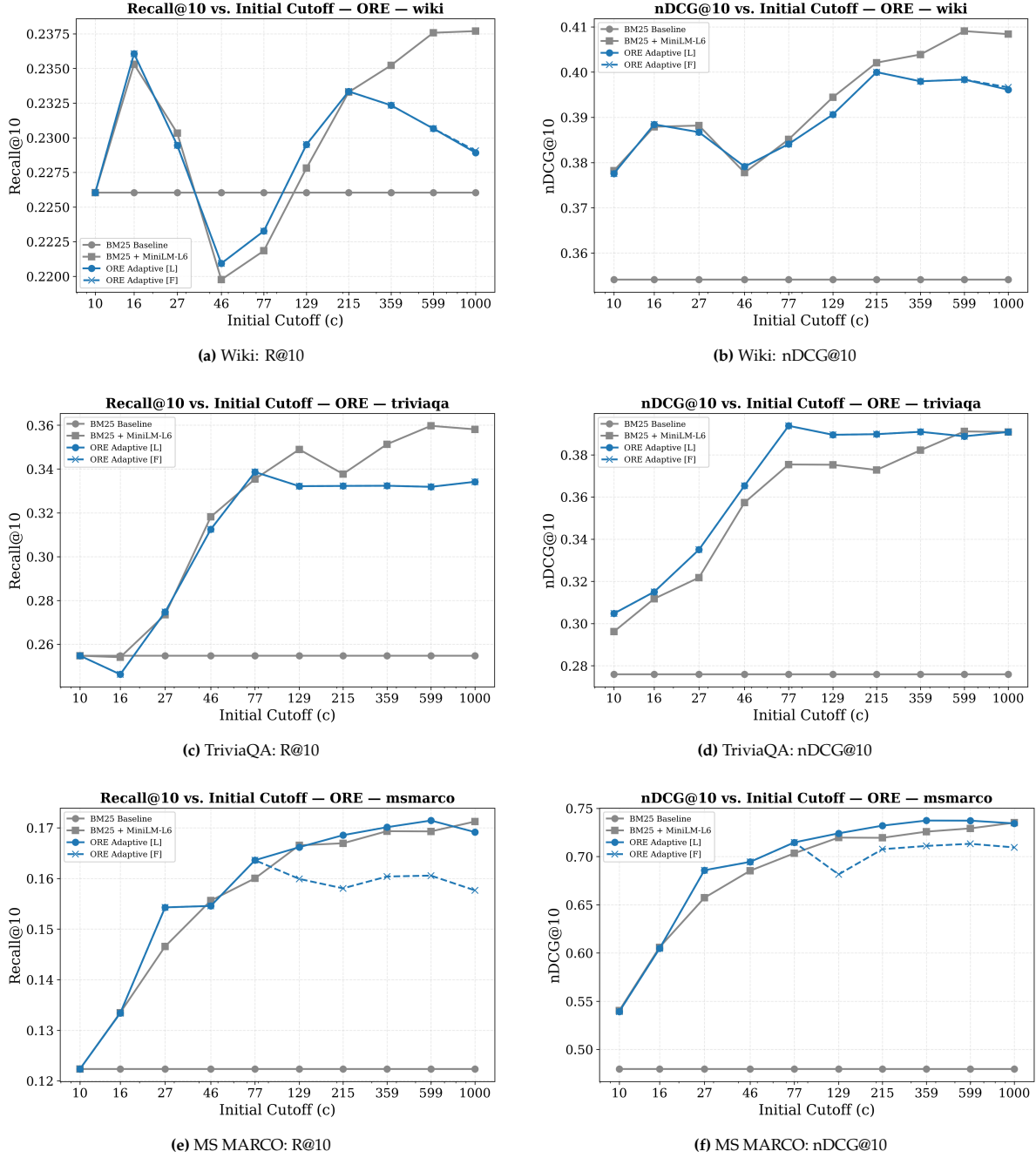


Figure 4.12: ORE adaptive retrieval performance as a function of initial cutoff depth c across all three evaluation corpora (Section 3.7.1, Table 3.8). Columns show Recall@10 (left) and nDCG@10 (right). Two cross-encoder budget modes are compared: Linear (proportional to c , solid lines) and Fixed ($ce=7$, dashed lines). The BM25 baseline and BM25+MiniLM-L6 reranker baseline are shown for reference. On MS MARCO, ORE uses a fully pre-computed LAFF graph; on TriviaQA and 2WikiMultihopQA, LAFF is computed on-the-fly.

4.4.2. ReformIR

Building on the analysis of adaptive graph-based exploration with ORE, we now examine ReformIR, an alternative adaptive retrieval framework operating through iterative query reformulation and dynamic weight learning (Section 3.7.2). ReformIR retrieves initial BM25 results for both the original query and $n=5$ LLM-generated reformulations, constructs a hybrid candidate pool via weighted combination, and iteratively scores candidates with a cross-encoder while learning optimal combination weights via

constrained least-squares regression. The same two scaling modes—Linear (proportional cross-encoder budget) and Fixed (capped at $ce=7$)—are applied, with c swept logarithmically from 10 to 1000.

Linear mode: broadly consistent scaling. In Linear mode, where every candidate receives a real cross-encoder score, ReformIR exhibits a broadly upward trajectory across all three corpora, though the improvement is not strictly monotonic (Figure 4.13). On MS MARCO, nDCG@10 rises from 0.561 at $c=10$ to 0.750 at $c=1000$, while R@10 improves from 0.130 to 0.172. ReformIR exceeds the BM25+MiniLM-L6 reranker baseline at every cutoff depth on MS MARCO, with margins of 0.015–0.036 in nDCG@10.

On Wiki, the gains are more modest but equally consistent: nDCG@10 increases from 0.401 to 0.440 and R@10 from 0.242 to 0.267. ReformIR surpasses the reranker baseline at all ten cutoff depths, with the largest margins at small c (e.g., +0.053 at $c=10$) where the iterative reformulation loop introduces candidates that single-pass retrieval would miss.

TriviaQA presents a mixed picture. nDCG@10 rises from 0.399 at $c=10$ to a peak of 0.435 at $c=46$, then fluctuates between 0.429 and 0.437 through the remainder of the sweep. R@10 improves from 0.306 to 0.337. Unlike MS MARCO and Wiki, ReformIR does not uniformly outperform the reranker baseline on TriviaQA: at $c=10$ and $c=16$, the reranker exceeds ReformIR by 0.004 and 0.014, respectively, though ReformIR surpasses the baseline at most higher budgets (by 0.005–0.016).

This cross-dataset contrast is notable relative to the ORE results in Section 4.4.1, where only MS MARCO exhibited clear scaling. The difference likely stems from the candidate discovery mechanism: ReformIR issues fresh BM25 queries at each iteration, drawing from the full inverted index, whereas ORE is constrained to the neighbourhood structure of a pre-built corpus graph.

Fixed mode: regression-induced degradation at large budgets. The Fixed mode results reveal systematic degradation once the cross-encoder budget is exhausted. On MS MARCO, ReformIR [F] peaks at nDCG@10 = 0.741 at $c=129$ and subsequently declines to 0.731 at $c=1000$, falling below the reranker baseline of 0.735. The pattern is more pronounced on TriviaQA, where nDCG@10 peaks at 0.435 ($c=46$) before dropping to 0.402 at $c=1000$ —well below the reranker baseline of 0.428. On Wiki, nDCG@10 peaks at 0.430 ($c=77$) and declines to 0.398 at $c=1000$, again falling below the reranker baseline of 0.425.

This degradation occurs because, at large Fixed-mode budgets, the least-squares regression model must estimate scores for the majority of candidates. As the ratio of unscored to scored documents increases, the regression model extrapolates beyond its training distribution, producing increasingly noisy score predictions for tail documents. Compared with ORE’s Fixed mode—which plateaued on MS MARCO and showed no effect on TriviaQA and Wiki—ReformIR’s Fixed mode at least matches or exceeds the reranker baseline at moderate budgets ($c \leq 129$) before regression-induced decline takes effect.

Saturation and the RM3 retrieval cap. Across both modes, the rate of improvement slows noticeably beyond $c \approx 129$. A contributing factor is an implementation constraint: the `_generate_rm3_query` method truncates RM3 retrieval results to 128 documents per iteration. Although the BM25 retriever for reformulated queries uses $\max(128, c)$ results, the RM3 feedback loop operates over at most 128 documents per iteration. At $c > 129$, each RM3 iteration therefore covers a diminishing fraction of the candidate pool, reducing the effectiveness of query expansion as a scaling mechanism. The steepest gains occur in $c \in [10, 129]$ across all three datasets, consistent with this architectural ceiling.

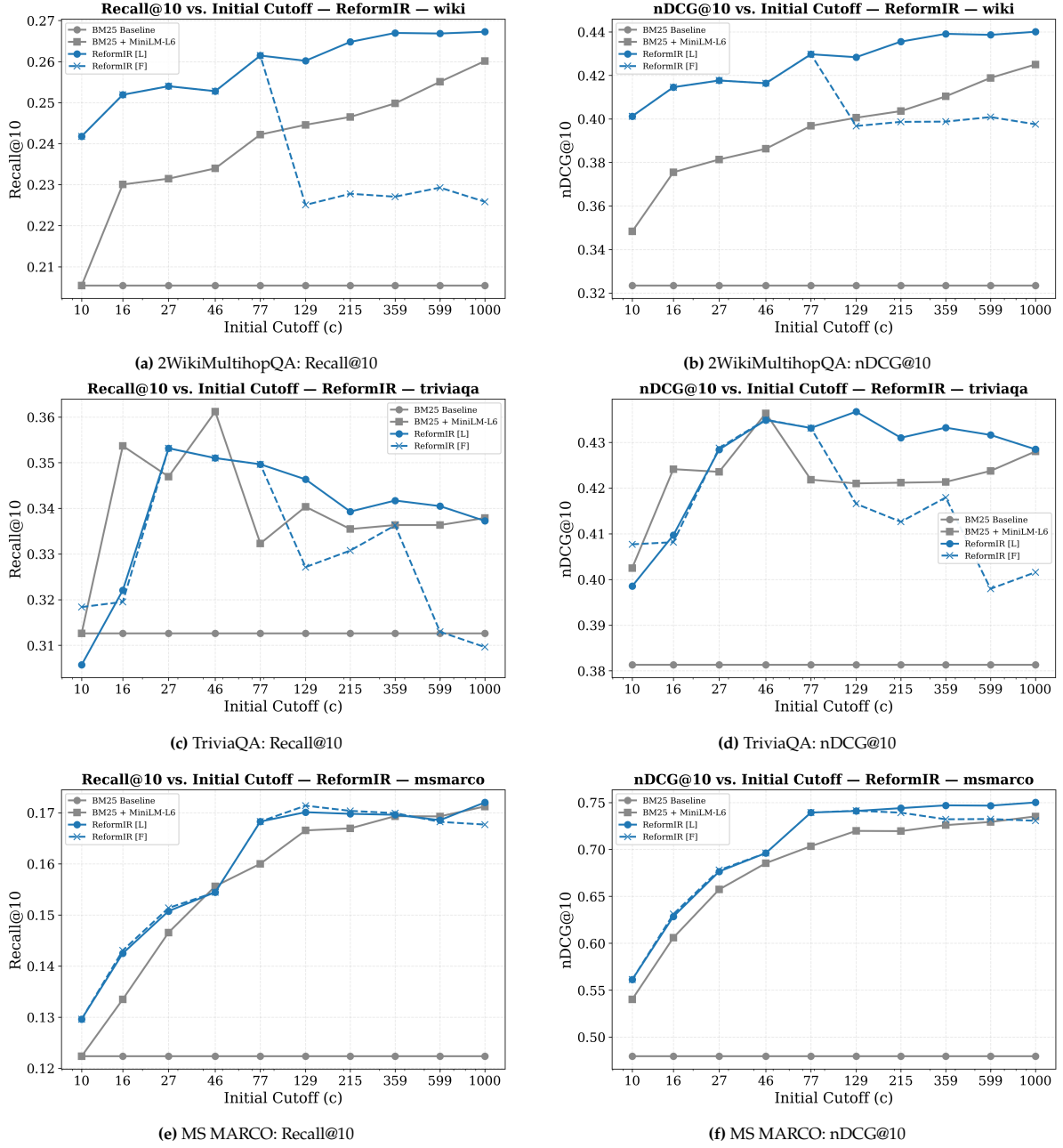


Figure 4.13: ReformIR adaptive retrieval performance as a function of initial cutoff depth c across all three evaluation corpora (Section 3.7.2, Table 3.9). Columns show Recall@10 (left) and nDCG@10 (right). Two cross-encoder budget modes are compared: Linear [L] (solid lines, full cross-encoder coverage) and Fixed [F] (dashed lines, $ce=7$ cap, forcing the λ -regression model to estimate scores for unscored candidates at large c). The BM25 baseline and BM25+MiniLM-L6 reranker baseline are shown for reference.

4.4.3. RAG Pipeline Performance

The end-to-end RAG experiment (Section 3.8) evaluates whether the retrieval scaling patterns observed in previous experiments translate to downstream generation quality. The pipeline feeds the top-5 passages—selected from c BM25 candidates, optionally reranked by MiniLM-L6—to a locally deployed Llama-3.2-3B-Instruct reader. Generated answers are evaluated with Exact Match (EM), token-level F1, and an LLM-as-a-Judge score (1–10 scale).

Methodological caveats. Three evaluation-level caveats frame this discussion. First, the LLM-as-a-Judge uses the same Llama model family as the reader, introducing a self-preference bias that may inflate semantic scores [43]. Second, the without-reranker baseline produces *identical* scores across all c values because the Concatenator always selects the top-5 passages by BM25 rank; increasing c only adds lower-ranked passages that the reader never sees. Consequently, all scaling effects reported below are attributable solely to the reranker’s ability to surface better passages into the top-5 at larger c . Third, EM and F1 scores are extremely low across both datasets (TriviaQA EM ≤ 0.05 ; Wiki EM ≤ 0.02)—a well-documented limitation of lexical matching metrics when applied to free-form generative outputs [43]. We therefore focus the analysis on the LLM judge metric while reporting EM and F1 for completeness.

TriviaQA. The without-reranker baseline achieves a constant judge score of 6.23 across all cutoffs. The reranked pipeline starts at 6.31 ($c=10$), dips to 6.12 at $c=16$, and subsequently recovers to reach 6.59 at $c=1000$ (Figure 4.14d). The trajectory is volatile rather than monotonic, with notable dips at $c=16$, $c=129$, and $c=359$, suggesting that the LLM judge introduces non-trivial evaluation noise. Nevertheless, the reranked pipeline exceeds the baseline at 8 out of 10 cutoff depths, with the largest margin appearing at $c=1000$ (+0.36).

Wiki. The without-reranker baseline holds at a constant 5.55. The reranked pipeline *underperforms* this baseline at small budgets: $c=10$ yields only 5.14, and the reranked pipeline does not surpass the baseline until $c=129$ (5.67). Beyond this crossover, the reranked pipeline improves to a peak of 6.05 at $c=599$ before declining to 5.91 at $c=1000$. The initial disadvantage at small c is consistent with the retrieval-stage finding that MiniLM-L6 occasionally reranks non-relevant passages into the top-5 when the candidate pool is small, degrading context quality. The reranked pipeline surpasses the baseline at 5 out of 10 cutoffs, all at $c \geq 129$.

Reranker impact and scaling. Across both datasets, the reranker confers a net advantage at larger retrieval budgets ($c \geq 129$), confirming that improved passage selection translates to better generation quality. At $c=1000$, the reranker margin over the baseline is +0.36 on both TriviaQA and Wiki. However, the benefit is not universal: on Wiki, the reranker actively harms performance at small c , and on TriviaQA, the volatile judge scores make it difficult to draw firm conclusions about intermediate budgets. The high noise level in the LLM judge metric—combined with the extremely low EM and F1 scores—cautions against over-interpreting fine-grained trends and underscores the need for more robust semantic evaluation protocols in RAG systems.

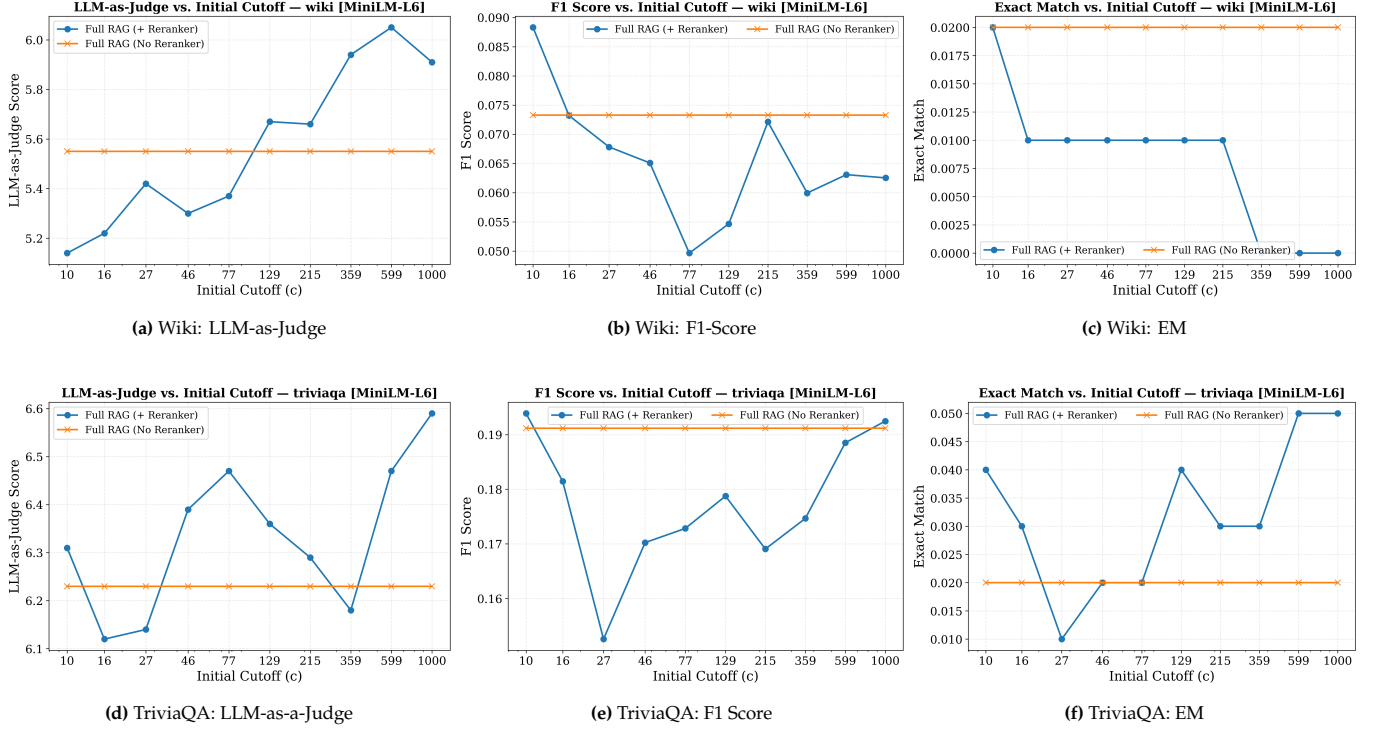


Figure 4.14: Effect of retrieval cutoff c on RAG generation quality across Wiki (top row) and TriviaQA (bottom row). Metrics shown are LLM-as-a-Judge accuracy (left column), F1 Score (middle column), and Exact Match (right column). The without-reranker baseline is constant across all c because the reader always receives the same top-5 BM25 passages.

5

Conclusion

This thesis demonstrates that test-time compute scaling in Retrieval-Augmented Generation is fundamentally a *systems-level coordination problem*: the returns on additional inference compute are not governed by a single scaling law, but by the interaction between retriever architecture, reranker robustness, fusion algorithm, pipeline ordering, and corpus characteristics. Through a systematic experimental programme spanning three benchmark datasets, two retriever architectures, six fusion algorithms, and two adaptive retrieval baselines, we have established empirical scaling characterisations that reveal when additional compute helps, when it hurts, and where current methods leave substantial performance on the table. We revisit the three research questions posed in Section 1.3 before discussing the broader design implications, limitations, and directions for future work.

5.1. Synthesis of Findings

Rather than recapitulating the detailed experimental results of Chapter 4, we distil the cross-cutting behavioural patterns that emerged across all six experimental conditions into a set of empirical principles for test-time compute allocation in compound retrieval systems.

Principle 1: Depth scaling succeeds or fails at the reranker. Expanding the candidate pool (c) monotonically improves retrieval quality when the cross-encoder is architecturally robust to noise—in our experiments, MiniLM-L6 scaled reliably across all datasets, retriever pairings, and pipeline orderings. MonoT5, by contrast, exhibited non-monotonic degradation when flooded with dense retrieval candidates, peaking early and declining at large c . This establishes a practical design rule: *the reranker’s noise tolerance, not the first-stage retriever, is the binding constraint on depth scaling*. Practitioners should invest in robust cross-encoders before increasing retrieval depth.

Principle 2: Breadth scaling requires a continuous embedding space. Scaling the number of query rewrites (n) via multi-query fusion produces meaningful gains only when the underlying retriever operates in a continuous semantic space. In our experiments, dense retrieval (TCT-ColBERT) enabled substantially larger fusion gains than sparse retrieval (BM25) across all three datasets, because semantically distinct rewrites activate genuinely different document regions in dense space. BM25

rewrites, by contrast, largely return overlapping candidate sets due to lexical similarity. For practitioners deploying multi-query fusion, this finding implies that *the retriever architecture should be selected before the fusion algorithm*.

Principle 3: Pipeline ordering interacts with domain calibration. The optimal ordering of reranking and fusion depends on the reranker’s calibration to the target corpus. When the reranker is well-calibrated (in-domain), reranking each query independently before fusing scores (RTM) substantially outperforms the standard merge-then-rerank (MTR) pipeline. When the reranker is poorly calibrated (out-of-domain), voting-based fusion methods that discard raw scores outperform score-based approaches. This conditional interaction means there is no universally optimal pipeline configuration—test-time scaling in RAG demands domain-aware orchestration.

Principle 4: Adaptive and fusion-based scaling are complementary, not competing. Graph-based adaptive retrieval (ORE) and query reformulation feedback (ReformIR) offer favourable compute–quality trade-offs at moderate budgets but do not match the deep recall gains of multi-query fusion at larger scales. In our experiments, adaptive methods excelled at extracting maximum quality from a fixed candidate pool, while fusion methods excelled at expanding the pool itself. Their complementary nature suggests that the most effective compound systems would combine both strategies: use adaptive methods to allocate cross-encoder budget efficiently within each rewrite, and fusion methods to aggregate across rewrites.

Principle 5: A persistent Oracle gap reveals the fusion bottleneck. Across all three datasets and rewrite depths, practical fusion algorithms leave approximately 15–17% relative Recall@50 on the table compared to an Oracle upper bound that selects the best per-query rewrite with perfect hindsight. This gap is remarkably stable: it does not shrink as n increases, indicating that the bottleneck lies in the aggregation mechanism itself rather than in candidate generation. The failure of BestRewrite—which attempts to predict the best rewrite from query text alone without retrieval feedback—confirms that closing this gap requires methods that incorporate early retrieval signals into the selection process.

Principle 6: Retrieval scaling gains are necessary but not sufficient for generation quality. Improvements in retrieval metrics (Recall, nDCG) partially transfer to downstream generation quality, but the signal is attenuated and volatile. In our end-to-end RAG experiments, reranking improved generation quality at large retrieval depths but occasionally harmed it at small depths, and LLM-as-Judge trajectories exhibited substantial variance across cutoff values. Lexical matching metrics (Exact Match, F1) were not sensitive enough to discriminate between pipeline configurations for free-form generative outputs. These observations caution against assuming that retrieval improvements will automatically translate to proportional generation gains, and underscore the need for more robust semantic evaluation protocols.

The overarching implication. Taken together, these six principles establish that test-time scaling in RAG cannot be reduced to a simple “more compute $\Rightarrow$ better results” relationship. Unlike the monotonic scaling laws observed in language model pre-training [13, 8], retrieval scaling is *non-monotonic* and *conditional*: the signal-to-noise ratio of the candidate pool introduces a constraint that has no direct analogue in language modelling. Effective test-time compute allocation in compound AI systems, therefore, requires domain-aware orchestration that jointly considers the retriever, reranker,

fusion algorithm, and pipeline ordering as an integrated system rather than as independently tunable components.

5.2. Limitations

We identify several methodological constraints that frame the scope and generalisability of these findings.

Fixed reader context window. All RAG experiments feed exactly the top-5 reranked passages to the generator, regardless of the retrieval budget c . This design choice creates a bottleneck: the thesis demonstrates that scaling c surfaces substantially better passages, yet the generator never consumes the expanded coverage. Modern long-context LLMs can process significantly more passages; by capping context at five, we limit our ability to measure the true end-to-end benefit of retrieval scaling.

LLM-as-Judge evaluation. The RAG evaluation relies on an LLM-as-Judge metric where the same model family (Llama 3.2) serves as both the answer generator and the quality evaluator. This evaluation protocol is standard practice in the field [43], but it introduces a known self-enhancement bias that may inflate semantic evaluation scores. Lexical matching metrics (Exact Match, F1) proved insufficiently sensitive for free-form generative outputs, leaving the LLM-as-Judge as the primary signal for end-to-end evaluation. Appendix A.4 documents representative failure cases. Future validation with an independent, stronger judge model would strengthen these conclusions.

Limited query set sizes. Evaluation is conducted on 43 queries (MS MARCO) and 100 queries (TriviaQA, 2WikiMultihopQA). While these sizes are standard for retrieval evaluation with expensive reranking pipelines, they limit the statistical power of fine-grained comparisons. Some of the variance observed in RAG evaluation trajectories may reflect sample size limitations rather than genuine pipeline instability.

Single rewrite model and temperature. All query rewrites are generated using a single model (GPT-4o-mini) at a fixed temperature ($\tau = 0.7$). Temperature directly controls the lexical and semantic diversity of generated rewrites, which in turn determines the degree of candidate pool expansion. A more comprehensive characterisation should investigate how temperature interacts with retrieval noise and fusion gains—lower temperatures may produce safer but redundant rewrites, while higher temperatures may increase diversity at the cost of query drift.

Adaptive method infrastructure requirements. ORE’s strong in-domain performance depends on a pre-computed LAFF corpus graph built over 8.8 million passages. When LAFF affinities are computed on-the-fly (as on 2WikiMultihopQA and TriviaQA), the adaptive advantage diminishes substantially. This infrastructure dependency limits the real-world applicability of graph-based adaptive retrieval to settings where corpus graphs can be pre-built and maintained.

5.3. Future Work

The limitations and empirical gaps identified above suggest several concrete directions for future research.

Co-scaling retrieval and reader context. The fixed top-5 context window is the most direct bottleneck identified in this thesis. Future work should systematically vary the number of passages ($K \in \{5, 10, 20, 50\}$) fed to a long-context generator as a function of the retrieval budget c , establishing joint scaling laws across the full retrieve $\rightarrow$ rerank $\rightarrow$ generate pipeline. This would reveal whether the Recall gains from depth and breadth scaling translate into proportional generation quality improvements when the reader can actually consume the expanded passage set.

Retrieval-feedback-guided rewrite selection. The BestRewrite baseline failed because it lacked retrieval feedback: the LLM cannot predict a rewrite’s retrieval yield from its text alone. A promising direction is training lightweight Process Reward Models (PRMs) that evaluate early retrieval signals—such as BM25 score distributions, candidate overlap statistics, or reranker confidence estimates—to identify and discard low-quality rewrites before they reach the expensive cross-encoder. This selective approach could close the Oracle gap identified in Principle 5 at a fraction of the compute cost of exhaustive fusion.

Rewrite diversity and temperature scaling. Since all experiments use a single temperature ($\tau = 0.7$), the interaction between rewrite diversity and fusion effectiveness remains unexplored. Future work should characterise the relationship between temperature, candidate pool diversity, and downstream retrieval quality, identifying whether an optimal diversity–noise trade-off exists and whether it varies across retriever architectures and fusion algorithms.

Stronger and independent evaluation. The LLM-as-Judge limitations documented in this thesis and in Appendix A.4 motivate the development of more robust evaluation protocols. Future work should employ an independent judge model (distinct from the generator) and incorporate reference-free evaluation methods to reduce self-enhancement bias and provide more reliable end-to-end RAG quality signals.

5.4. Closing Remarks

This thesis provides a systematic empirical characterisation of test-time compute scaling in multi-query retrieval-augmented generation. The central finding is that *scaling retrieval compute is effective, but only under the right system-level conditions*: the reranker must be robust to noise, the retriever must operate in a continuous semantic space, the fusion algorithm must match the reranker’s calibration regime, and the downstream reader must be able to consume the expanded passage set. These conditional scaling behaviours distinguish retrieval systems from the monotonic training-time scaling laws that have dominated the field, and establish that Compound AI Systems require domain-aware orchestration rather than uniform compute allocation. We hope these findings provide a practical foundation for building more effective, compute-efficient retrieval pipelines.

Declaration of AI Assistance

During the preparation of this thesis, large language models (including OpenAI’s GPT-4o-mini and Meta’s Llama 3.2) were utilised strictly as computational tools for query reformulation (as part of the experimental methodology) and automated answer evaluation. Additionally, LLMs (Gemini, ChatGPT, and Claude) were used during the drafting process for structural ideation, copy-editing, and clarity refinement. Google Antigravity was used to help code, but all code has been reviewed. Finally, all

experimental design, data analysis, core arguments, and conclusions are the original work of the author.

References

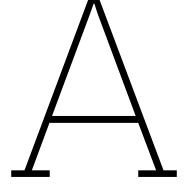
- [1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. <https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2>. 2024.
- [2] Akari Asai et al. “Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection”. In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL: <https://openreview.net/forum?id=hSyW5go0v8>.
- [3] Lorenz Brehme, Thomas Ströhle, and Ruth Breu. “Can LLMs be Trusted for Evaluating RAG Systems? A Survey of Methods and Datasets”. In: *IEEE Swiss Conference on Data Science, SDS 2025, Zürich, Switzerland, June 26-27, 2025*. IEEE, 2025, pp. 16–23. DOI: 10.1109/SDS66131.2025.00010. URL: <https://doi.org/10.1109/SDS66131.2025.00010>.
- [4] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. “Reciprocal rank fusion outperforms condorcet and individual rank learning methods”. In: *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*. Ed. by James Allan et al. ACM, 2009, pp. 758–759. DOI: 10.1145/1571941.1572114. URL: <https://doi.org/10.1145/1571941.1572114>.
- [5] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. “SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking”. In: *SIGIR ’21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz et al. ACM, 2021, pp. 2288–2292. DOI: 10.1145/3404835.3463098. URL: <https://doi.org/10.1145/3404835.3463098>.
- [6] Luyu Gao et al. “Precise Zero-Shot Dense Retrieval without Relevance Labels”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*. Ed. by Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki. Association for Computational Linguistics, 2023, pp. 1762–1777. DOI: 10.18653/v1/2023.acl-long.99. URL: <https://doi.org/10.18653/v1/2023.acl-long.99>.
- [7] 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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. Ed. by Donia Scott, Núria Bel, and Chengqing Zong. International Committee on Computational Linguistics, 2020, pp. 6609–6625. DOI: 10.18653/v1/2020.coling-main.580. URL: <https://doi.org/10.18653/v1/2020.coling-main.580>.
- [8] Jordan Hoffmann et al. “Training Compute-Optimal Large Language Models”. In: *CoRR abs/2203.15556 (2022)*. DOI: 10.48550/ARXIV.2203.15556. arXiv: 2203.15556. URL: <https://doi.org/10.48550/arXiv.2203.15556>.
- [9] Yunhai Hu et al. “MCTS-RAG: Enhancing Retrieval-Augmented Generation with Monte Carlo Tree Search”. In: *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*. Ed. by Christos Christodoulopoulos et al. Association for Computational Linguistics, 2025, pp. 12581–12597. DOI: 10.18653/v1/2025.findings-emnlp.672. URL: <https://doi.org/10.18653/v1/2025.findings-emnlp.672>.

- [10] Yixin Ji et al. *A Survey of Test-Time Compute: From Intuitive Inference to Deliberate Reasoning*. 2025. arXiv: 2501.02497 [cs.AI]. URL: <https://arxiv.org/abs/2501.02497>.
- [11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. “Billion-Scale Similarity Search with GPUs”. In: *IEEE Trans. Big Data* 7.3 (2021), pp. 535–547. doi: 10.1109/TBDATA.2019.2921572. URL: <https://doi.org/10.1109/TBDATA.2019.2921572>.
- [12] Mandar Joshi et al. “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Ed. by Regina Barzilay and Min-Yen Kan. Association for Computational Linguistics, 2017, pp. 1601–1611. doi: 10.18653/V1/P17-1147. URL: <https://doi.org/10.18653/v1/P17-1147>.
- [13] Jared Kaplan et al. “Scaling Laws for Neural Language Models”. In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361. URL: <https://arxiv.org/abs/2001.08361>.
- [14] 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 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 6769–6781. doi: 10.18653/V1/2020.EMNLP-MAIN.550. URL: <https://doi.org/10.18653/v1/2020.emnlp-main.550>.
- [15] Tom Kwiatkowski et al. “Natural Questions: a Benchmark for Question Answering Research”. In: *Trans. Assoc. Comput. Linguistics* 7 (2019), pp. 452–466. doi: 10.1162/TACL_A_00276. URL: https://doi.org/10.1162/tacl_a_00276.
- [16] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>.
- [17] Minghan Li et al. “Query Expansion in the Age of Pre-trained and Large Language Models: A Comprehensive Survey”. In: *CoRR* abs/2509.07794 (2025). doi: 10.48550/ARXIV.2509.07794. arXiv: 2509.07794. URL: <https://doi.org/10.48550/arXiv.2509.07794>.
- [18] Lingyuan Liu and Mengxiang Zhang. “Exp4Fuse: A Rank Fusion Framework for Enhanced Sparse Retrieval using Large Language Model-based Query Expansion”. In: *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*. Ed. by Wanxiang Che et al. Vol. ACL 2025. Findings of ACL. Association for Computational Linguistics, 2025, pp. 163–173. doi: 10.18653/V1/2025.FINDINGS-ACL.9. URL: <https://doi.org/10.18653/v1/2025.findings-acl.9>.
- [19] 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, Atlanta, GA, USA, October 17-21, 2022*. Ed. by Mohammad Al Hasan and Li Xiong. ACM, 2022, pp. 1491–1500. doi: 10.1145/3511808.3557231. URL: <https://doi.org/10.1145/3511808.3557231>.

- [20] Craig Macdonald et al. "PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval". In: *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. Ed. by Gianluca Demartini et al. ACM, 2021, pp. 4526–4533. DOI: 10.1145/3459637.3482013. URL: <https://doi.org/10.1145/3459637.3482013>.
- [21] Tri Nguyen et al. "MS MARCO: A Human Generated MACHine Reading COMprehension Dataset". In: *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*. Ed. by Tarek Richard Besold et al. Vol. 1773. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.
- [22] Rodrigo Nogueira et al. "Document Ranking with a Pretrained Sequence-to-Sequence Model". In: *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Vol. EMNLP 2020. Findings of ACL. Association for Computational Linguistics, 2020, pp. 708–718. DOI: 10.18653/v1/2020.findings-emnlp.63. URL: <https://doi.org/10.18653/v1/2020.findings-emnlp.63>.
- [23] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. "Quam: Adaptive Retrieval through Query Affinity Modelling". In: *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM 2025, Hannover, Germany, March 10-14, 2025*. Ed. by Wolfgang Nejdl et al. ACM, 2025, pp. 954–962. DOI: 10.1145/3701551.3703584. URL: <https://doi.org/10.1145/3701551.3703584>.
- [24] Mandeep Rathee et al. "Breaking the Lens of the Telescope: Online Relevance Estimation over Large Retrieval Sets". In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025*. Ed. by Nicola Ferro et al. ACM, 2025, pp. 2287–2297. DOI: 10.1145/3726302.3729910. URL: <https://doi.org/10.1145/3726302.3729910>.
- [25] Stephen E. Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Found. Trends Inf. Retr.* 3.4 (2009), pp. 333–389. DOI: 10.1561/15000000019. URL: <https://doi.org/10.1561/15000000019>.
- [26] Keshav Santhanam et al. "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*. Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz. Association for Computational Linguistics, 2022, pp. 3715–3734. DOI: 10.18653/v1/2022.naacl-main.272. URL: <https://doi.org/10.18653/v1/2022.naacl-main.272>.
- [27] Joseph A. Shaw and Edward A. Fox. "Combination of Multiple Searches". In: *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*. Ed. by Donna K. Harman. Vol. 500-225. NIST Special Publication. National Institute of Standards and Technology (NIST), 1994, pp. 105–108. URL: <http://trec.nist.gov/pubs/trec3/papers/vt.ps.gz>.
- [28] Weijia Shi et al. "REPLUG: Retrieval-Augmented Black-Box Language Models". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*. Ed. by Kevin Duh, Helena Gómez-Adorno, and Steven Bethard. Association for

- Computational Linguistics, 2024, pp. 8371–8384. doi: 10.18653/V1/2024.NAACL-LONG.463. URL: <https://doi.org/10.18653/v1/2024.naacl-long.463>.
- [29] Aditi Singh et al. “Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG”. In: *CoRR* abs/2501.09136 (2025). doi: 10.48550/ARXIV.2501.09136. arXiv: 2501.09136. URL: <https://doi.org/10.48550/arXiv.2501.09136>.
- [30] Charlie Snell et al. “Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters”. In: *CoRR* abs/2408.03314 (2024). doi: 10.48550/ARXIV.2408.03314. arXiv: 2408.03314. URL: <https://doi.org/10.48550/arXiv.2408.03314>.
- [31] Nandan Thakur et al. “BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html>.
- [32] Venktesh V, Mandeep Rathee, and Avishek Anand. “When More Reformulations Hurt: Avoiding Drift using Ranker Feedback”. In: *CoRR* abs/2605.00560 (2026). doi: 10.48550/ARXIV.2605.00560. arXiv: 2605.00560. URL: <https://doi.org/10.48550/arXiv.2605.00560>.
- [33] Liang Wang et al. “Chain-of-Retrieval Augmented Generation”. In: *CoRR* abs/2501.14342 (2025). doi: 10.48550/ARXIV.2501.14342. arXiv: 2501.14342. URL: <https://doi.org/10.48550/arXiv.2501.14342>.
- [34] Wenhui Wang et al. “MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- [35] Yangzhen Wu et al. “Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for LLM Problem-Solving”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: <https://openreview.net/forum?id=VNckp7JEHn>.
- [36] Zhichao Xu et al. “Rethinking On-policy Optimization for Query Augmentation”. In: *CoRR* abs/2510.17139 (2025). doi: 10.48550/ARXIV.2510.17139. arXiv: 2510.17139. URL: <https://doi.org/10.48550/arXiv.2510.17139>.
- [37] 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, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 2369–2380. doi: 10.18653/V1/D18-1259. URL: <https://doi.org/10.18653/v1/d18-1259>.
- [38] Shunyu Yao et al. “ReAct: Synergizing Reasoning and Acting in Language Models”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=WE_vluYUL-X.
- [39] Zhenrui Yue et al. “Inference Scaling for Long-Context Retrieval Augmented Generation”. In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: <https://openreview.net/forum?id=FSjIrOm1vz>.

- [40] Matei Zaharia et al. *The Shift from Models to Compound AI Systems*. Berkeley Artificial Intelligence Research Blog. Accessed: 2026-06-19. 2024. URL: <https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/>.
- [41] Duzhen Zhang et al. "From System 1 to System 2: A Survey of Reasoning Large Language Models". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 48.3 (2026), pp. 3335–3354. DOI: 10.1109/TPAMI.2025.3637037. URL: <https://doi.org/10.1109/TPAMI.2025.3637037>.
- [42] Qiyuan Zhang et al. "What, How, Where, and How Well? A Survey on Test-Time Scaling in Large Language Models". In: *CoRR* abs/2503.24235 (2025). DOI: 10.48550/ARXIV.2503.24235. arXiv: 2503.24235. URL: <https://doi.org/10.48550/arXiv.2503.24235>.
- [43] Lianmin Zheng et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.
- [44] Shu Zhou et al. "Inference Scaling Law for Retrieval Augmented Generation". In: *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*. Ed. by Sven Koenig, Chad Jenkins, and Matthew E. Taylor. AAAI Press, 2026, pp. 16522–16530. DOI: 10.1609/AAAI.V40I19.38692. URL: <https://doi.org/10.1609/aaai.v40i19.38692>.



Experimental Setup Appendix

A.1. Experimental run Details

Table A.1: Experiment run provenance. Each final result is traced to its execution platform via the `data_path` field recorded in the run metadata.

Reranker	Dataset	Platform	Date	Run ID
<i>BM25 & Dense Cutoff Sweep</i>				
MiniLM	Wiki-MuSiQue	Local (WSL)	2026-05-19	20260519_072823
MiniLM	MSMARCO	Colab	2026-05-19	20260519_150454
MiniLM	DPR-w100	Colab	2026-05-19	20260519_233710
<i>Merge-then-Rerank (BM25 + Fusion)</i>				
MiniLM	Wiki-MuSiQue	Colab	2026-05-20	20260520_024310
MonoT5	Wiki-MuSiQue	Local (WSL)	2026-05-20	20260520_071506
MiniLM	MSMARCO	Local (WSL)	2026-05-20	20260520_043732
MonoT5	MSMARCO	Local (WSL)	2026-05-20	20260520_054215
MiniLM	DPR-w100	Colab	2026-05-20	20260520_040247
MonoT5	DPR-w100	Colab	2026-05-24	20260524_162907
<i>Dense Retrieval + Fusion</i>				
MiniLM	Wiki-MuSiQue	Local (WSL)	2026-05-29	20260529_004144
MiniLM	MSMARCO	Colab	2026-05-29	20260529_174302
<i>Rerank-then-Merge</i>				
MiniLM	Wiki-MuSiQue	Local (WSL)	2026-06-01	20260601_221131
MonoT5	Wiki-MuSiQue	Local (WSL)	2026-06-18	20260618_052111
MiniLM	MSMARCO	Local (WSL)	2026-06-01	20260601_201710
MonoT5	MSMARCO	Local (WSL)	2026-06-19	20260619_231024
MiniLM	DPR-w100	Local (WSL)	2026-06-09	20260609_141022
MonoT5	DPR-w100	Local (WSL)	2026-06-17	20260617_022013
<i>Oracle Recall Analysis</i>				
MiniLM	Wiki-MuSiQue	Local (WSL)	2026-06-07	20260607_002106
MiniLM	MSMARCO	Local (WSL)	2026-06-06	20260606_213946
MiniLM	DPR-w100	Local (WSL)	2026-06-09	20260609_141753
<i>ORE Adaptive</i>				
—	Wiki-MuSiQue	Colab	2026-06-01	20260601_175220
—	MSMARCO	Local (WSL)	2026-06-03	20260603_063745
—	DPR-w100	Colab	2026-06-02	20260602_035903
<i>Full RAG Pipeline</i>				
MiniLM	Wiki-MuSiQue	Colab	2026-06-18	20260618_103900
MiniLM	DPR-w100	Colab	2026-06-18	20260618_111651
<i>ReformIR</i>				
—	Wiki-MuSiQue	Local (WSL)	2026-06-07	20260607_065946
—	MSMARCO	Colab	2026-05-29	20260529_000452
—	DPR-w100	Colab	2026-05-29	20260529_004808

Platform key: *Local (WSL)* = Windows Subsystem for Linux on personal machine; *Colab* = Google Colab (GPU runtime). Platform determined from the `data_path` field in each run’s metadata JSON: paths starting with `/mnt/c/` indicate local execution, paths starting with `/content/` indicate Google Colab.

A.2. Query Rewriting Prompts and Examples

A.2.1. LLM Rewriting System Prompt

The following system prompt is used with GPT-4o-mini ($\tau = 0.7$) to generate query rewrites. The user message contains only the original query. The variable {k} is replaced with the number of requested rewrites (30 in all experiments).

You are an expert retrieval engineer.

Your task is to generate high-diversity, retrieval-oriented rewrites of a user query.

Follow these rules strictly:

1. Diversity Types (you MUST cover all):

A. Lexical Diversity

Use different surface forms, synonyms, and industry-specific vocabulary. Avoid trivial paraphrases (e.g., swapping "nation" with "country"). Make rewrites differ by at least 30-40% token overlap.

B. Structural Diversity

Produce different grammatical forms: question -> declarative, "X of Y" -> "Y's X", long -> short, inverted order, keyword-style queries, multi-hop versions.

C. Semantic Diversity

Introduce meaningfully different retrieval intents: entity identification, attribute extraction, contextual expansion, source-specific phrasing ("IMDb", "Wikipedia", "Wikidata"), catalog / KB schema terms ("citizenship", "birthplace", "founded", "parent organization"), related concepts that commonly co-occur in documents.

D. Multi-Hop Coverage

If the original query has latent steps (e.g., film -> director -> nationality), generate rewrites that address each hop separately.

E. Document-Style Diversity

Match the styles of documents from: Wikipedia, Wikidata, StackExchange, IMDb / Rotten Tomatoes, News, Scholarly articles, Product listings, APIs and schemas.

F. Domain Adaptation

If the domain is: movies -> use film metadata terms; people -> biography, occupation, awards; geography -> location hierarchy; products -> specs, compatibility; science -> definitions, formulas, variations; legal -> statutes, regulations; finance -> metrics, ratios, filings; web search -> keywords + entities. Always produce rewrites that reflect real-world search logs, not artificial paraphrases.

2. Rewrites That Are Forbidden

Never produce rewrites that are: trivial paraphrases, differ only by one swapped noun, only reorder words with no semantic change, too vague or generic, contradict the original question, introduce irrelevant information, hallucinate specifics.

3. Rewrites That Are Required

For every user query, ALWAYS produce: Entity-Surfacing Rewrite, Attribute-Focused Rewrite, Source-Targeted Rewrite, Keyword / Telegraphed Query, Decomposed / Multi-Hop Version, Schema / KB-Style Rewrite, Context Expansion Rewrite.

4. Output Format

Produce {k} rewrites. Each rewrite must be meaningfully distinct. Order does not matter. Return strictly JSON: a list of exactly {k} unique, concise paraphrases. Do not include any explanation or keys, only the JSON array.

5. Optimization Behavior

Maximize recall, not precision. Assume both BM25 and dense retrievers will be used. Favor rewrites that expose latent entities, metadata, or structured attributes. For complex or ambiguous queries, generate rewrites that cover different plausible interpretations. Balance specificity and generality to increase coverage.

A.2.2. BestRewrite Selection Prompt

The following prompt is used with GPT-4o-mini ($\tau = 0$) to select the single best rewrite from a set of candidates. The temperature is set to zero for deterministic selection.

System: You are an expert in information retrieval.

Given an original query and a numbered list of candidate query rewrites, select the ONE rewrite that is most likely to retrieve the most relevant documents when used with BM25 and dense retrieval systems.

Consider:

- Keyword coverage and specificity
- Entity surfacing and attribute extraction
- Retrieval diversity and recall potential
- Conciseness vs. informativeness tradeoff

Return strictly JSON: {"best_index": <int>} where <int> is the 0-based index of the best rewrite. Do not include any explanation.

User: Original query: [query]

Candidate rewrites:

0: [rewrite 0]

1: [rewrite 1]

...

A.2.3. Query Rewrite Examples

A.3. LLM-as-a-Judge Few-Shot Examples

The following five few-shot examples are included in the judge prompt (Section 3.8.2) to calibrate the scoring rubric. Each example demonstrates an edge case in factual evaluation.

Table A.2: Examples of LLM-generated query rewrites across all three evaluation datasets. Each original query is rewritten into retrieval-oriented formulations by GPT-4o-mini ($\tau = 0.7$); four representative rewrites are shown per query (out of 30 generated).

Original Query	Example Rewrites
<i>Wiki-MuSiQue (multi-hop reasoning)</i>	
Which country the director of film <i>One Law For The Woman</i> is from?	1. What is the nationality of the director of the film <i>One Law For The Woman</i>? 2. Country of citizenship for the director of <i>One Law For The Woman</i>. 3. Where is the director of <i>One Law For The Woman</i> from? 4. Nationality of the filmmaker behind <i>One Law For The Woman</i>.
What educational institution did Edward Eversfield's father attend?	1. Identify the university where the father of Edward Eversfield studied. 2. Wikipedia page detailing Edward Eversfield's father's education. 3. Edward Eversfield's father's school or college. 4. What was the alma mater of Edward Eversfield's father?
<i>TriviaQA / DPR-w100 (open-domain factoid QA)</i>	
Who is the real-life baseball manager featured in the film <i>Moneyball</i> ?	1. Identify the baseball manager and ex-player portrayed in <i>Moneyball</i>. 2. Baseball manager associated with the 2011 film <i>Moneyball</i>. 3. <i>Moneyball</i>'s central figure: a baseball manager and player. 4. What is the name of the manager from <i>Moneyball</i>?
Identify the mother of Clytemnestra and Helen in Greek mythology.	1. What is the name of the goddess who bore Clytemnestra and Helen? 2. Parentage of Clytemnestra and Helen in Greek myth. 3. Clytemnestra and Helen's mother. 4. Who is the mother figure associated with Clytemnestra and Helen in Greek lore?
<i>MS MARCO (web passage retrieval)</i>	
Do goldfish grow?	1. Can goldfish increase in size? 2. Growth patterns of goldfish. 3. How large can goldfish get? 4. Goldfish size increase over time.
What is the difference between Wi-Fi and Bluetooth?	1. Compare and contrast Wi-Fi and Bluetooth connectivity methods. 2. Characteristics of Wi-Fi and Bluetooth wireless protocols. 3. Wi-Fi versus Bluetooth: key distinctions. 4. Explain the functionalities of Wi-Fi and Bluetooth.

Table A.3: Few-shot examples used in the LLM-as-a-Judge prompt. Each example pairs a query, gold answer, and system answer with the expected score.

Query	Gold Answer	System Answer	Score
Which film has the director died first, A or B?	Film A	1979	1
Why did person's wife die?	stroke	1951	1
Where was the director of film X born?	New York	None of the above	1
Who is person's father?	John Smith	John Smith Jr.	5
What nationality is the director of film Y?	United States	American	9

Examples 1–3 demonstrate that type mismatches (year instead of film name), temporal answers for causal questions, and refusals all receive the minimum score. Example 4 shows that a partial match (correct family but wrong generation) receives a middling score. Example 5 shows that semantically equivalent but lexically different answers ("United States" vs. "American") receive a near-perfect score.

A.4. LLM-as-a-Judge Failure Cases

Despite the calibration examples above, the LLM-as-Judge metric exhibits systematic failure modes when deployed on real evaluation data. Table A.4 presents representative failure cases observed during the TriviaQA RAG experiment (Section 4.4.3). These failures fall into three categories: (1) *false positives*, where clearly incorrect or refused answers receive high scores; (2) *type confusion*, where the judge conflates related but categorically wrong answer types (e.g., years instead of names); and (3) *inconsistent scoring*, where the same prediction–reference pair receives different scores depending on the reference alias.

Table A.4: Representative LLM-as-Judge failure cases from the TriviaQA evaluation. Each row shows the query, gold reference answer, system prediction, and the judge’s score (1–9 scale). Scores highlighted in **bold** indicate clear misjudgements.

Query	Gold Answer	Prediction	Score
<i>False Positive: Refusal scored as correct</i>			
What famous American writer was granted a patent for a best-selling book that contained no words?	Mark Twain (Self-Pasting Scrapbook)	None of the above.	9
To whom did the Bee Gees pay tribute in Tapestry Revisited?	Carole King	None of the above.	6
<i>False Positive: Type mismatch scored as correct</i>			
In which sport did Hollywood star Sonja Henie win Olympic gold?	Ice skating	1928, 1932, 1936	9
Who had a 70s No. 1 hit with “The Night the Lights Went Out in Georgia”?	Vicki Lawrence	1973	6
<i>Inconsistent scoring across reference aliases</i>			
Balice International Airport is in which country?	Poland	1. Poland	9
	Etymology of Poland	1. Poland	1
	Lenkija	1. Poland	1
Which British political figure became Baroness Kesteven?	Margaret Thatcher	1st Duke of Ancaster and Kesteven	3
	The Baroness Thatcher	1st Duke of Ancaster and Kesteven	6

The most severe failure mode is false-positive scoring of refusals: the judge assigns a score of 9 to “None of the above” for the Mark Twain query, directly contradicting the calibration example in Table A.3 where a refusal receives a score of 1. Type confusion is equally problematic: the Sonja Henie example receives a perfect score despite answering *when* rather than *what sport*. The inconsistent alias scoring (Poland vs. “Etymology of Poland”) reveals that the judge struggles with Wikipedia-style entity aliases that embed the correct answer as a substring. These failure modes motivate the limitation discussed in Section 5.2 regarding LLM-as-Judge self-preference bias.

B

MonoT5 Reranker Results

This appendix presents the MonoT5 reranker results for the merge-then-rerank (MTR) and rerank-then-merge (RTM) query breadth experiments. These results complement the MiniLM-L6 results presented in the main text (Sections 4.2.1 and 4.2.3).

B.1. Merge-then-Rerank with MonoT5 (Experiment 2)

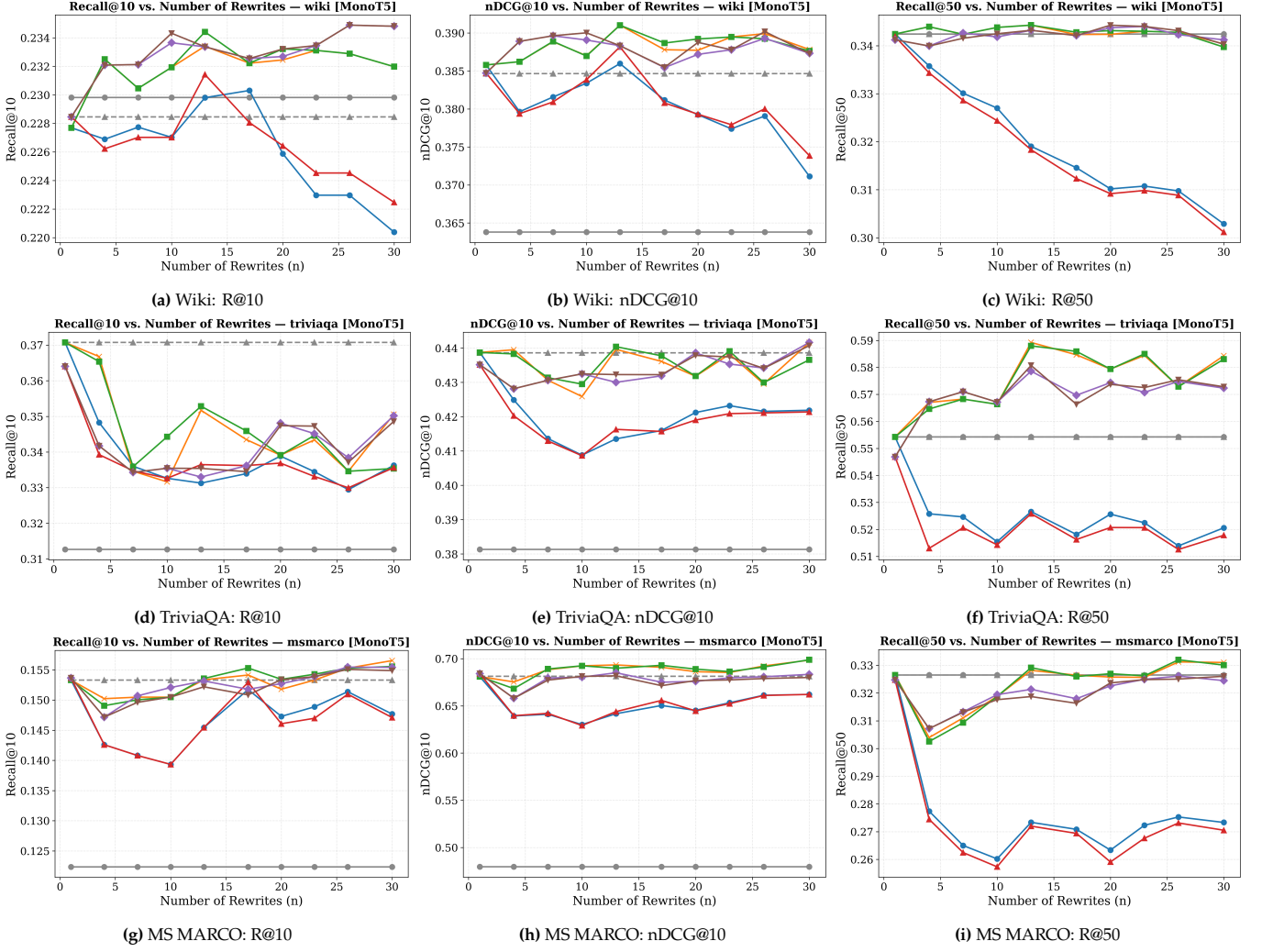


Figure B.1: Effect of query rewrite breadth n on retrieval quality under the merge-then-rerank (MTR) paradigm with BM25 first-stage retrieval and MonoT5 reranking. Layout matches Figure 4.2. Rows: Wiki-MuSiQue (top), TriviaQA (middle), MS MARCO (bottom). Columns: R@10 (left), nDCG@10 (middle), R@50 (right).

B.2. Rerank-then-Merge with MonoT5 (Experiment 2b)

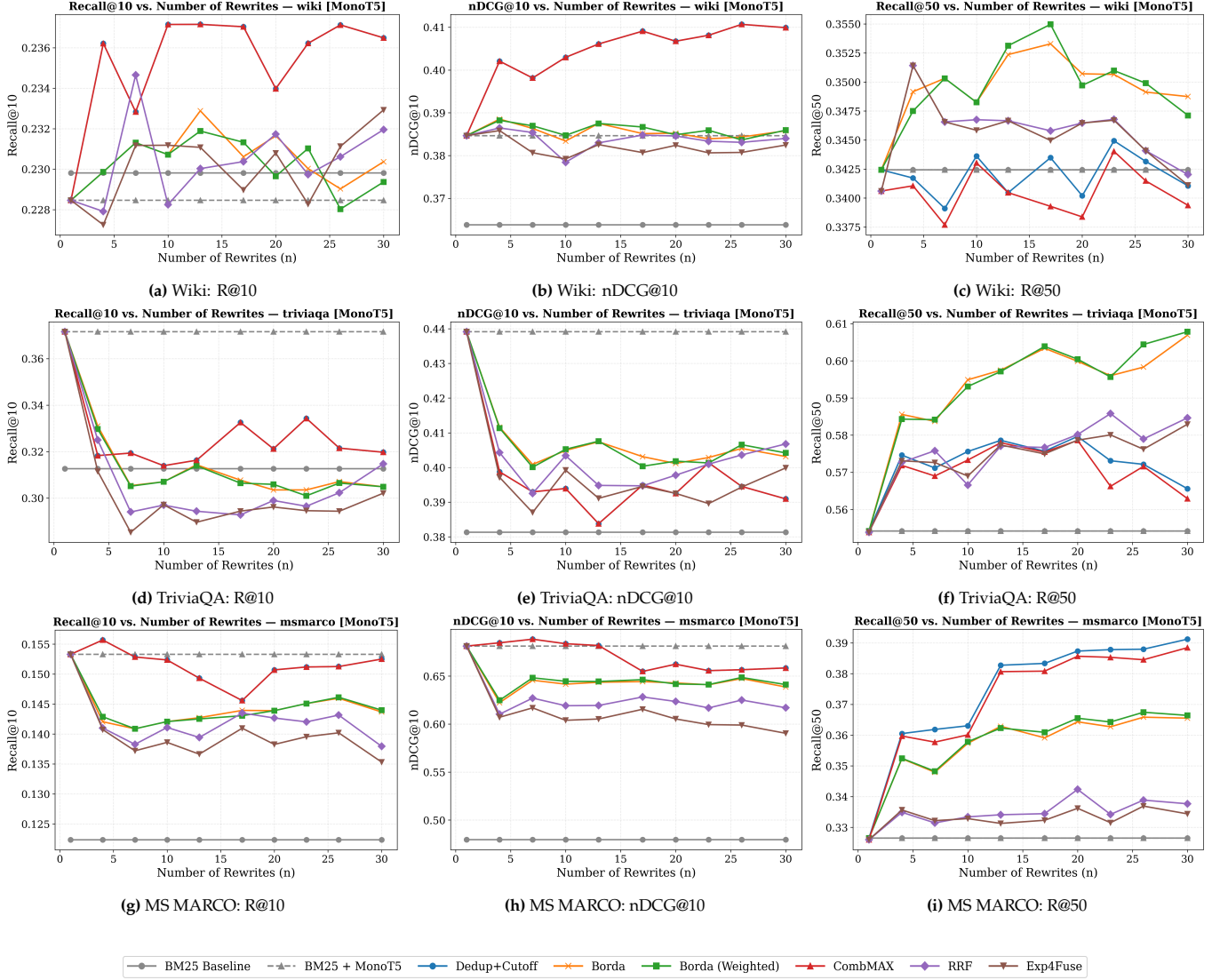


Figure B.2: Effect of query rewrite breadth n on retrieval quality under the rerank-then-merge (RTM) paradigm with BM25 first-stage retrieval and MonoT5 reranking. Layout matches Figure 4.4. Rows: Wiki-MuSiQue (top), TriviaQA (middle), MS MARCO (bottom). Columns: R@10 (left), nDCG@10 (middle), R@50 (right).