

One Size Does Not Fit All!

Dynamic Retriever and Generator Selection for RAG

Neeraj Anand
Media and Data Science Research Lab
Adobe Systems, Noida, India
neeraja@adobe.com

Payel Santra
IACS
Kolkata, India
payel.iacs@gmail.com

Partha Basuchowdhuri
IACS
Kolkata, India
partha.basuchowdhuri@iacs.res.in

Debasis Ganguly
University of Glasgow
Glasgow, United Kingdom
debasis.ganguly@glasgow.ac.uk

Sumit Bhatia
Media and Data Science Research Lab
Adobe Systems, Noida, India
sumit.bhatia@adobe.com

Abstract

Retrieval-Augmented Generation (RAG) systems typically employ fixed retriever and generator configurations across queries, despite substantial differences in query complexity and information needs, leading to inefficient allocation of computational resources. While retrieval and generation adaptivity have been studied independently, their joint effect on end-to-end RAG performance remains underexplored. We systematically analyze how retriever and generator complexity interacts across factoid and multi-hop question answering (QA), including bridge and composition reasoning tasks. Our analysis shows that stronger retrieval generally yields larger gains than increased generation effort, but both exhibit diminishing and non-monotonic returns, indicating that higher-complexity configurations are not uniformly better across queries.

Motivated by these findings, we introduce DRAG, a query-adaptive framework for selecting retriever-generator configurations. We first propose DRAG_{QPP}, a training-free routing approach that uses Query Performance Prediction (QPP) signals to guide retriever selection and perplexity-based measures over retrieved context to guide generator selection. We further introduce DRAG_{SFT}, a supervised routing approach that fine-tunes an LLM to jointly predict retriever-generator configurations. Across three LLM families and four QA benchmarks, DRAG_{QPP} achieves performance comparable to strong static RAG baselines while substantially reducing inference latency, whereas DRAG_{SFT} consistently improves effectiveness over static and training-free adaptive baselines. Overall, DRAG demonstrates that jointly adapting retrieval and generation achieves a more favorable effectiveness-efficiency trade-off than static RAG pipelines.

CCS Concepts

• **Information systems** → **Query intent; Information retrieval query processing.**

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/XXXXXXX.XXXXXXX>

Keywords

Adaptive RAG, Retriever Selection, Generator Selection

ACM Reference Format:

Neeraj Anand, Payel Santra, Partha Basuchowdhuri, Debasis Ganguly, and Sumit Bhatia. 2018. One Size Does Not Fit All! Dynamic Retriever and Generator Selection for RAG. In . ACM, New York, NY, USA, 12 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction

Large Language Models (LLMs) have demonstrated strong capabilities in text generation; however, their responses remain constrained by the knowledge encoded in their parameters during pretraining [35]. As a result, LLMs often struggle to reliably generate or reason over information that is absent from, outdated in, or only sparsely represented in their training data. Retrieval-Augmented Generation (RAG) addresses this limitation by explicitly incorporating external knowledge sources into the generation process [27, 35, 73]. A typical RAG pipeline consists of two components: (i) a *retriever* (such as BM25 [48], or a dense ranker such as E5 [66]) that selects contextually relevant documents from an external corpus, and (ii) a *generator*, which is typically a decoder-based LLM (e.g., LLaMA [63] or GPT [6]), which conditions on both the input query and the retrieved contexts to produce a task-specific response. Most existing RAG systems rely on a fixed retriever-generator configuration that is applied uniformly across all queries [15, 43, 67, 79], thus leading to computational inefficiency and suboptimal effectiveness. However, prior work in IR has long established that queries differ substantially in the complexity of their underlying information needs, which in turn affects retrieval difficulty and downstream generation quality [7, 14, 26].

In a RAG setting, query complexity spans a broad spectrum: from simple factoid queries answerable using a single relevant document [25, 32], to compositional multi-hop queries requiring reasoning across multiple evidence sources [64, 70], as well as queries involving temporal reasoning [23], numerical aggregation [11], or multi-constraint satisfaction [57]. For simpler queries, standard sparse or dense retrieval models (e.g., sparse such as BM25 or bi-encoders such as E5 [66]) are often sufficient without requiring computationally expensive re-ranking strategies [13, 71]. In contrast, more complex queries require stronger retrieval and generation pipelines, achieved either through enhanced reasoning capabilities

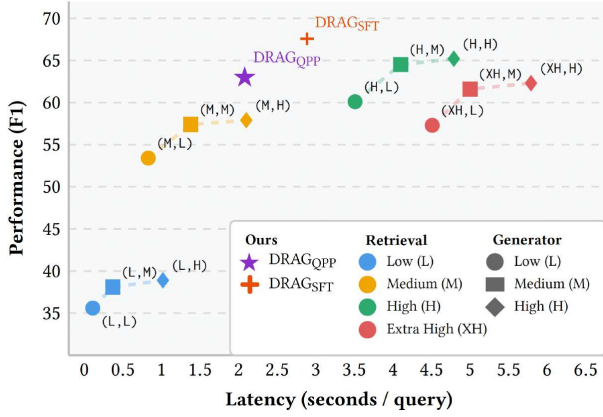


Figure 1: Performance (F1) and efficiency (time/query) on the HotpotQA dataset across 12 retriever-generator configurations (4 retrievers $\times$ 3 generators) of increasing complexity. The DRAG_{QPP} and DRAG_{SFT} methods adaptively select query-specific retriever-generator pairs, achieving a favorable effectiveness-efficiency trade-off by attaining strong downstream performance at lower latency than static RAG with high-complexity configurations.

in the generator, such as multi-step reasoning [44], query decomposition [1], increasing the *thinking* capability of the generator, structured generation strategies such as chain-of-thought [30], or by improving the retrieval pipeline using tools such as *retrieve-rerank* pipelines employing cross-encoders [27, 29, 45] and LLM-based listwise rankers [55, 58, 76, 80, 81]. However, these improvements often come at the cost of increased computational overhead and latency [46]. This suggests that a *query-adaptive RAG workflow that dynamically allocates retrieval and generation capacity based on the requirements of each query can potentially improve the overall effectiveness-efficiency trade-off*.

While promising, implementing such an adaptive framework presents two primary challenges: (i) *generalizability*, since routing decisions must remain robust across diverse query types and adapt to evolving model configurations without relying on relevance judgments or answer quality signals at inference time; and (ii) the *trade-off between effectiveness and efficiency*, since increased retrieval or reasoning effort improves answer quality only selectively while substantially increasing latency. To address these challenges, we first conduct a **systematic empirical analysis of retriever-generator configurations of varying complexity** (Section 3), examining how their interactions affect downstream RAG effectiveness and efficiency. This analysis provides insights into when additional computational capacity is beneficial and motivates query-adaptive configuration selection.

To this end, we introduce **DRAG (Dynamic RAG)**, a query-adaptive framework for selecting retriever-generator configurations. We instantiate DRAG through two complementary approaches (Section 4): a *training-free* dynamic approach, **DRAG_{QPP}**, which routes queries through a cascaded sequence of models of increasing complexity, and a *supervised* approach **DRAG_{SFT}**, that learns to jointly predicts retriever-generator configurations from query signals. Together, these approaches explore the effectiveness-efficiency

trade-off (EET) of adaptive RAG under different levels of supervision and computational overhead [39].

We study both DRAG_{QPP} and DRAG_{SFT} under the common DRAG framework to analyze their respective strengths and limitations across standard benchmarks as well as evaluate them in out-of-domain (OOD) scenarios. We study OOD robustness using the FRAMES dataset, where routing models are applied without dataset-specific tuning and we show that adaptive retriever-generator selection can generalize beyond the training distribution.

The contributions of our work can be summarized as :

- We present the first comprehensive empirical study analyzing the joint impact of retriever-generator pair selection on end-to-end RAG performance.
- We propose **DRAG_{QPP}**, a training-free, query-adaptive framework that leverages unsupervised retrieval performance prediction (QPP-based [78]) and generator performance prediction (GPP-based [52]) to dynamically balance the EET.
- We introduce **DRAG_{SFT}**, a supervised framework that treats configuration selection as a predictive task, which improves selection effectiveness over DRAG_{QPP}, albeit at cost of latency.
- We conduct extensive evaluations across three diverse LLM families and four benchmarks – including out-of-domain evaluation on FRAMES – demonstrating the robustness and generalizability of our adaptive paradigms across full test sets.
- To facilitate reproducible research, we release our dataset and code at <https://anonymous.4open.science/r/DynamicRAG-2A01/>.

2 Related Works

Retrieval Augmented Generation. Retrieval-Augmented Generation (RAG) improves text generation by grounding LLM outputs in external knowledge, thereby mitigating hallucinations and knowledge limitations. Prior work has explored several forms of retrieval augmentation, including query-based retrieval and retrieval-conditioned generation [17, 51], latent and logit-level integration of retrieved evidence into language modeling [28, 77], retrieval refinement through reranking, active, and iterative retrieval strategies [16, 24, 75], as well as architectures that improve reasoning over retrieved documents and multi-document fusion [19, 34, 50]. Another line of work explores supervised fine-tuning of LLMs, that retrofit any LLM with retrieval capabilities [36, 65], distill reader signal back into the retriever [20], or fine-tune the generator to reason over retrieved context [3, 36].

Adaptive Retrieval. Existing studies on adaptive RAG focused on binary decisions, i.e., whether to retrieve or not [38]. This method is sufficient for simple queries, but generally fails multi-hop difficult queries. Thus, some studies introduces a method with a variable number of iterative retrieval steps, i.e., it jointly retrieve-and-rerank until sufficient evidence is accumulated for correct prediction[47]. Furthermore, SELF-RAG [3, 21] introduces training strategy to retrieve, self-critique and refine the outputs. Unlike our method DRAG, it does not explicitly address the problem of selecting among multiple retriever-generator configurations. In contrast, several works explore adaptivity from reasoning point of view, while assuming a fixed retrieval pipeline. Some existing studies varies the number of reasoning steps [68] or samples multiple chain-of-thought

reasoning paths to dynamically select the generation path for improving robustness [72]. Different from previous approaches, this paper jointly analyzes the impact of retriever and generator choices with different capabilities on RAG framework.

Query Performance Prediction. Query performance predictors can be classified into two major categories: pre-retrieval and post-retrieval. Pre-retrieval prediction technique aims to predict query difficulty by analyzing the prior distribution to querying the given collection of documents [10, 18, 41]. The post-retrieval prediction methodology, on the contrary, relies on statistical approaches on retrieval scores [5, 54, 59, 78], learning to rank features inspired predictors [4] and robustness of rank-list perturbation based predictors [49, 53]. Some recent studies explored supervised and neural approaches for QPP by considering both the content and the score features [7, 74]. In this work, we use these pre-retrieval predictors in our downstream pipeline to guide the selection of an appropriate retriever based on query complexity.

3 RAG Configuration Analysis and Problem Formulation

We first formalize the adaptive RAG task and analyze the behavior of *static* RAG systems, where a fixed retriever and generator are used uniformly across all queries. This analysis serves two purposes: (i) characterizing the EET across retriever-generator configurations, and (ii) motivating the need for query-adaptive frameworks.

Task Formulation. Given a query q , a set of candidate retrievers $\Theta = \{\theta_i\}_{i=1}^m$ and a set of candidate generators $\Phi = \{\phi_i\}_{i=1}^n$, the objective of adaptive RAG is to select a query-specific retriever-generator configuration $(\theta, \phi) \in \Theta \times \Phi$ that balances downstream answer quality and computational cost. Unlike static RAG, adaptive RAG seeks to select the optimal retriever and generator based on the query’s specific information need to maximize effectiveness while minimizing latency.

Since the effectiveness of a retriever-generator pair cannot be directly observed at inference time (i.e., the ground-truth answer a_q is unavailable), we need an estimator, $f : (q, \theta, \phi) \mapsto \mathbb{R}$, that predicts the likelihood of generating a correct answer for a given query q . The objective is then to select the most efficient configuration that maximizes downstream effectiveness while minimizing computational overhead.

Retriever and Generator Configuration Choice. The adaptive setup requires a discrete action space of retrievers Θ and generators Φ , ordered by increasing computational cost and capability. Following the conventional view in IR, we maintain the progression of retriever complexity from *lexical retrieval* to *dense semantic retrieval* and further to sophisticated *retrieve-and-rerank* pipelines employing query decomposition and cross-encoder rerankers. Similarly, generator complexity scales from lightweight decoding with no or minimal reasoning to longer, multi-step reasoning i.e., extending the reasoning effort or thinking modes of LLMs. *Importantly, our proposed framework is entirely agnostic to the specific number of retriever or generator choices, allowing it to scale seamlessly across finer- or coarser-grained complexity modes.*

In our experiments for GPT-OSS base-model, we consider four retrieval levels, i.e., *low* (L; sparse retriever), *medium* (M; dense

Gen / Ret	Low (%)	Medium (%)	High (%)	Extra-High (%)
Low	11.2	5.6	12.5	6.9
Medium	5.1	3.8	5.7	4.5
High	13.0	7.1	15.6	9.0

Table 1: Distribution (%) of correctly answered HotpotQA queries under an efficiency-aware oracle RAG system across total 12 configurations, with four retriever levels (L, M, H, XH) and three GPT-OSS generator reasoning levels (L, M, H). The results show that a substantial fraction of queries can be answered using low and medium complexity configurations, indicating that considerable latency reductions are achievable through adaptive framework.

retriever), *high* (H; retrieve-and-rerank pipeline), and *extra-high* (XH; query decomposition with retrieve-and-rerank pipeline), reflecting complex retrieval pipelines with increasing latency and three cascaded levels for retriever component, i.e., *low* (L), *medium* (M) and *high* (H) reflecting the mode of thinking, which means total 12 configurations (4×3). Similarly, for Qwen and Gemma, we consider two generator modes depending on thinking availability (thinking-off/on), resulting in 8 configurations (4×2) per model.

Efficiency-aware Oracle configuration. To understand the query-level EET in static RAG frameworks, we define an *efficiency-aware oracle setup* that assigns each query to the least computationally expensive configuration of retriever-generator pair to correctly answer the query q . Intuitively, the oracle represents an ideal setup that has access to ground-truth effectiveness and therefore reveals how retrieval and generation effort should ideally be allocated across queries. Formally, given a query q , the oracle selects

$$(\theta^*, \phi^*) = \arg \min_{\theta, \phi \in \Theta \times \Phi} \mathcal{L}(\theta, \phi) \quad \text{s.t.} \quad \mathcal{P}(a_q, \phi(q, \theta_k(q))) \geq \delta, \quad (1)$$

where i) $\theta_k(q)$ denotes the top- k documents retrieved by the ranker θ , ii) $\phi \in \Phi : (q, \theta_q) \mapsto \hat{a}$ denotes a generator which takes q along with its retrieved documents θ_q as input to yield a predicted answer $\hat{a}_q$, iii) a_q is the ground-truth answer for query q , iv) $\mathcal{P} : a_q \times \hat{a}_q \mapsto \mathbb{R}$ denotes downstream effectiveness (measured by metrics such as Exact Match (EM) and F1) between $\hat{a}_q$ and a_q , v) $\mathcal{L} : \Theta \times \Phi \mapsto \mathbb{R}$ is the total latency and vi) δ represents a effectiveness threshold. When multiple configurations satisfy the threshold, the oracle selects the least expensive one.

In practice, if no configuration satisfies the target criterion (e.g., EM= 1), we map to the least expensive configuration achieving high partial correctness (F1 $\geq$ 0.8). Following prior works that applied thresholds on answer-level F1 to account for minor lexical variations in partial correct responses [3, 24], we use 0.8 as a threshold. If no configuration satisfies this criterion, the query is assigned to the highest-complexity setting (θ_{XH}, ϕ_H) as a safe fallback.

Discussion. Table 1 reports the oracle distribution across all retriever-generator configurations on the HotpotQA dataset when GPT-OSS is the base model. Note that a substantial fraction of queries are correctly answered using low- and medium-complexity configurations, indicating that uniformly invoking the most expensive retrieval and reasoning pipeline is often unnecessary.

This data underscores a fundamental reality of RAG optimization: query complexity dictates architectural requirements, meaning

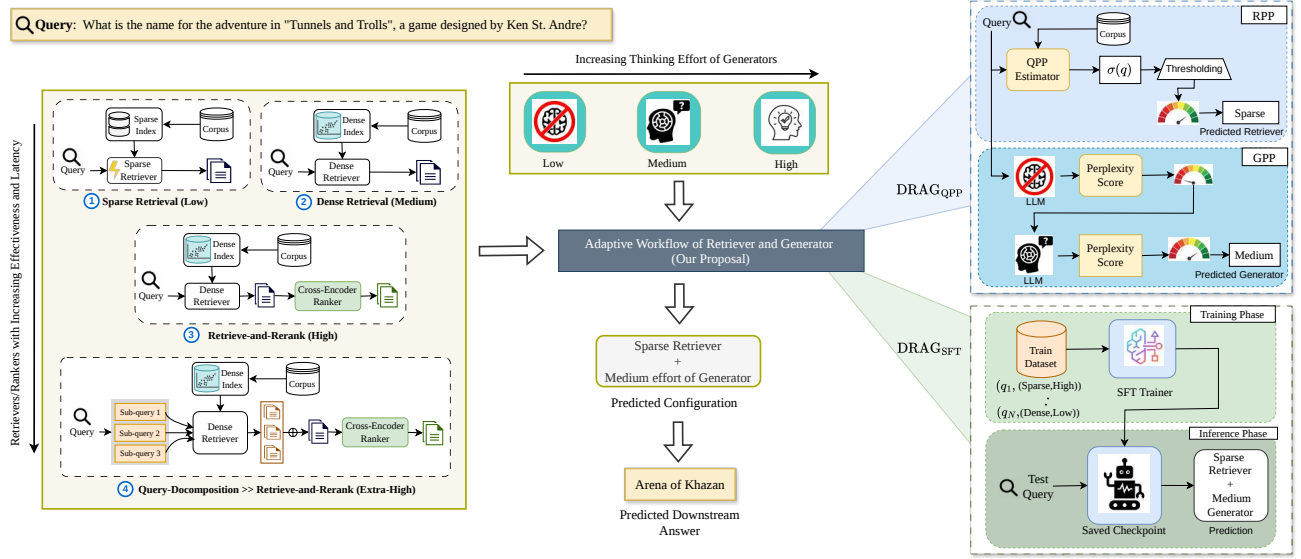


Figure 2: Overview of the proposed query-adaptive framework (DRAG) for GPT-OSS. The left panel shows retriever and generator configurations of progressively increasing complexity, ranging from sparse to query decomposition with retrieve-rerank pipelines, and from low to high thinking effort. Given an input query, DRAG dynamically selects a query-specific retriever-generator configuration instead of uniformly invoking a fixed pipeline. The top-right panel illustrates the training-free DRAG_{QPP} workflow with 2 modules, *retrieval performance prediction* (RPP) followed by *generator performance prediction* (GPP). The bottom-right panel shows DRAG_{SFT} , a supervised framework that learns retriever-generator selection from query-configuration supervision and predicts the optimal configuration during inference. The selected configuration is then executed to produce the downstream answer. While illustrated using GPT-OSS with four retrievers and three generator levels ($4 \times 3 = 12$ configurations), the framework naturally generalizes to other model families (e.g., Qwen and Gemma with 4×2 configurations). For our proposed adaptive DRAG_{QPP} approach, we used training-free RPP and GPP-based methods as explained in Section 4.

some queries are adequately resolved by lightweight configurations while others strictly demand advanced retrieval and reasoning pipelines. For example, a simple factoid question from HotpotQA, “Alfie Allen played Theon Greyjoy on which show?” is correctly answered across *all* retriever-generator configurations; for such simple queries, employing expensive retrieval and reasoning pipelines incurs unnecessary computational cost without yielding any performance gains. Conversely, the oracle distribution (Table 1) also reveals that there are a substantial number of queries that are correctly answered only under an intermediate retrieval setting paired with complex reasoner (e.g., θ_M with ϕ_H), while other modes of retrievers fail irrespective of generator choice. For instance, the query “Belle Gold is a fictional character portrayed by an actress of what nationality?” requires resolving a semantic entity relationship (*Belle Gold* $\rightarrow$ *portraying actress* $\rightarrow$ *nationality*), where dense retrieval (θ_M) successfully retrieves semantically relevant evidence, whereas other retrievers either miss relevant context or retrieved noisy evidence. Thus, higher-complexity configurations are not strict supersets of lower ones, and we need **adaptive per-query routing framework** over the retriever-generator space $\Theta \times \Phi$ to better balance downstream effectiveness and inference latency.

4 Dynamic RAG Workflow (DRAG)

Motivated by the observations in Section 3, which demonstrate that no single configuration is optimal across queries, we introduce DRAG, an adaptive framework that approximates the oracle

routing policy of Equation 1 when the ground-truth answer a_q is unavailable at inference time. Depending on the availability of supervision, we introduce two complementary approaches under this paradigm: (i) DRAG_{QPP} , a training-free framework that sequentially selects retrievers and generators using unsupervised performance signals, and (ii) DRAG_{SFT} , a supervised framework that directly learns optimal, query-specific retriever-generator configurations. An overview of the proposed DRAG workflow, including both DRAG_{QPP} and DRAG_{SFT} frameworks, is illustrated in Figure 2.

4.1 DRAG_{QPP}

As the initial step towards realizing an optimized adaptive framework, we introduce DRAG_{QPP} , a training-free routing framework that approximates the oracle routing policy using unsupervised techniques. Rather than directly predicting a joint configuration, DRAG_{QPP} decomposes the routing process into two sequential prediction stages: (i) Retriever Performance Prediction and (ii) Generator Performance Prediction. The selected retriever and generator are then used for downstream answer generation.

4.1.1 Retriever Performance Prediction (RPP). The first module in DRAG_{QPP} framework aims to select the most suitable retriever for each query by analyzing retrieval difficulty [62] without requiring relevance judgments at inference time. Because adaptive routing must decide required retriever complexity *before* retrieval is executed, we need a *retriever-independent* signal that relies solely the query text and corpus statistics. Query Performance Prediction

(QPP) directly addresses this need by estimating retrieval quality in the absence of relevance judgments [8, 12]. Motivated by this, we leverage a *pre-retrieval* QPP predictor $\sigma(q) : q \mapsto \mathbb{R}$ as a signal to guide per-query retriever selection [9, 40]; the larger values of σ indicate queries expected to have relevant documents after retrieval. Intuitively, queries that are easier to retrieve can be handled by lightweight retrievers, whereas harder queries are escalated to progressively stronger retrieval pipelines.

QPP-aware Retriever Routing. Rather than evaluating the effectiveness of all retrievers for each query, which would incur substantial computational overhead and defeat the purpose of adaptive routing, we adopt a lightweight query-specific routing strategy that sequentially selects the retrievers with increasing complexity. Since retriever selection must be performed without actually performing each retrieval, we used *pre-retrieval* QPP predictors, which are *retriever-independent* and depend on the query and corpus statistics.

Formally, given a query q , we compute a QPP score $\sigma(q)$ and select the least complex retriever satisfying

$$\theta^*(q) = \theta_i, \quad \text{where } i = \min \{j \mid \sigma(q) \geq \tau_j\}, \quad (2)$$

where $\{\tau_j\}_{j=1}^{m-1}$ is a set of $m-1$ thresholds (where, $\tau_j < \tau_{j+1}, \forall j$) dividing the score range into m intervals. A query interval with lower value of σ indicate the need of more complex retrievers, while easier queries compute higher σ and are handled by simpler, cheaper retrievers.

In our experiments, we employ *Average IDF* (Avg-IDF) [33] as our lightweight, pre-retrieval QPP signal to capture query specificity with respect to the underlying corpus. Queries dominated by rare, highly informative content terms typically receive higher Avg-IDF scores and are inherently easier to retrieve. Conversely, queries with ambiguous or common terms yield lower scores, indicating they would benefit from more advanced retrieval pipelines. We select Avg-IDF aligned with recent literature demonstrating its robustness [2], a choice further validated by our initial experiments against alternative pre-retrieval predictors.

Threshold Selection. We derive the routing thresholds $\{\tau_i\}_{i=1}^{m-1}$ from the Avg-IDF score distribution computed over the training set. Concretely, these thresholds are defined via percentile-based partitioning of the predictor values. We select a base percentile p as a tunable hyperparameter and determine subsequent thresholds using uniform offsets, i.e., $\{p, p + \Delta, p + 2\Delta, \dots\}$, where Δ controls the interval spacing between adjacent complexity levels. The score values corresponding to these specific percentiles are then fixed as the routing thresholds $\{\tau_i\}$, which are applied to the test data to map incoming queries to their respective retriever complexity levels based on their online QPP scores.

4.1.2 Generator Performance Prediction (GPP). As the second component of our framework, DRAG_{QPP} selects an appropriate generator for each query, conditioned on the specific retriever θ^* selected in the previous stage. To capture the generator’s uncertainty regarding the retrieved evidence prior to answer synthesis, we evaluate the perplexity of the retrieved context sequence itself under the base language model [52]. Perplexity is computed as the exponential of the mean negative log-likelihood of the tokens within the retrieved context documents. Notably, a lower perplexity score

indicates that the retrieved evidence is highly consistent with the model’s internal language distribution and parametric expectations. Conversely, a high perplexity score signals that the context is noisy, out-of-distribution, or poorly aligned, which drastically escalates the risk of downstream hallucination [22, 69]. Consequently, queries paired with low-perplexity context are routed to lightweight generators, while queries yielding high-perplexity context are escalated to more complex, reasoning-intensive models capable of handling noisy evidence. Formally, for a given query q and retrieved set $\theta_k^*(q)$, we compute the generator-specific perplexity score $h(\theta_k^*(q) \mid q; \phi_j)$ and select the generator according to,

$$\phi^*(q) = \phi_i, \quad \text{where } i = \min \{j \mid h(\theta_k^*(q) \mid q; \phi_j) \leq \nu_j\}. \quad (3)$$

Threshold on Posterior. For each generator ϕ_i , we estimate the distribution of perplexity scores over training queries and define a routing threshold ν_i as the score corresponding to the p -th percentile ($p \in \{10, 20, \dots\}$), where p is treated as a hyperparameter selected on train data. During inference, we select generators sequentially from low to high complexity. We compute the perplexity of the retrieved context and compare it against the corresponding threshold ν_i . If the perplexity exceeds ν_i – implying that the query can be answered by the current generation tier’s capacity to process the retrieved context’s informational density – the current generator is selected. Otherwise, the query is escalated to the next higher-complexity tier. This process continues until a generator satisfies this threshold criterion or the highest-complexity model is deployed by default.

4.2 DRAG_{SFT}

While DRAG_{QPP} provides a lightweight and training-free mechanism for adaptive routing using query and corpus derived signals, its effectiveness relies heavily on data-specific threshold selection and the sequential estimation of generation complexity. This limitation motivates a more generalizable supervised approach that directly models retriever and generator selection jointly within a unified framework.

To this end, DRAG_{SFT} casts routing as a supervised mapping $\hat{\mathcal{P}} : q \mapsto (\theta, \phi)$ from an input query to its optimal retriever-generator configuration. We instantiate $\hat{\mathcal{P}}$ by fine-tuning Qwen as the base model on our constructed training dataset $\mathcal{D}_{\text{oracle}}$ (described in Section 3), where each query is paired with its optimal configuration label. The input context consists of the query, the instruction template, and the available configurations, and the model is trained to produce a structured JSON output of the form $\{\text{"retriever": } \theta, \text{"generator": } \phi\}$, specifying the selected configuration.

At inference time, given a query q , the fine-tuned model predicts the retriever-generator configuration $(\hat{\theta}, \hat{\phi})$ in a single decoding step. The selected retriever $\hat{\theta}$ retrieves the top- k contexts, which are subsequently passed to the generator $\hat{\phi}$ to produce the final answer $\hat{a}_q$. By jointly predicting both components directly from the query, DRAG_{SFT} avoids the sequential routing and threshold calibration required by DRAG_{QPP}, while introducing negligible upfront inference overhead.

Dataset	Task	#Train	#Test	$ Q $	Domain
TriviaQA	Single-hop	78,785	11,313	14.2	Wiki/Web
HotpotQA	Multi-hop	90,447	7,405	17.8	Wiki
MuSiQue		19,938	2,417	26.4	
FRAMES		-	824	16.0	

Table 2: Characteristics of the datasets used in our experiments. We used the entire test set for all evaluations. FRAMES is used as an OOD benchmark while the remaining datasets are in-domain.

5 Experimental Setup

Our empirical evaluation is designed to answer the following core research questions about the behavior of RAG pipelines.

- **RQ1 (Retriever vs. Generator Capacity):** How do retriever and generator complexity individually contribute to end-to-end RAG performance, and does scaling either component in isolation consistently lead to downstream quality gains?
- **RQ2 (Dynamic vs. Static RAG Trade-offs):** Can dynamically selecting the retriever and generator configuration optimize the effectiveness-efficiency trade-off (EET) relative to conventional static, fixed-complexity RAG pipelines?
- **RQ3 (Framework Generalization):** How effectively do our proposed adaptive routing strategies generalize when evaluated across out-of-domain datasets and unseen model configurations?

5.1 Datasets

We evaluate our framework on four question answering (QA) benchmarks selected to represent a spectrum of retrieval and reasoning complexities. **TriviaQA** [25] consists primarily of open-domain factoid questions that are typically answerable using a single supporting document, serving as a baseline for settings with limited evidence dispersion. To capture increasing retrieval and reasoning difficulty, we consider three multi-hop QA benchmarks: **HotpotQA** [70], which requires bridging information across indirectly connected documents, **MuSiQue** [64], which enforces stricter compositional reasoning (2-4 hops) with widely dispersed evidence and, finally we include **FRAMES** [31] as our multi-hop OOD evaluation benchmark, since, unlike the previous datasets, it involves more diverse reasoning requirements (e.g., numerical, temporal, and multi-constraint reasoning), often requiring aggregation across a larger set of documents. Thus, FRAMES provides a challenging testbed for evaluating the robustness and generalizability of the proposed adaptive routing methodologies. Dataset statistics are reported in Table 2.

5.2 Retrievers Investigated

The action space Θ should span retrieval pipelines that differ significantly in cost and capability, so the routing policy has substantial trade-offs to learn. In this study, the granularity of $|\Theta|$ is itself a design choice that can vary from simple binary routing to more fine-grained adaptive configurations.

In this work, motivated by the evolution of retrievers and rankers in IR, we consider Θ as a spectrum of retrieval configurations with progressively increasing complexity: lexical retrieval, dense semantic retrieval, retrieve-rerank pipelines employing cross-encoders,

and query decomposition followed by reranking. These configurations have a different spectrum of latency and effectiveness. Although we focus on these dimensions, the framework is general and can accommodate other configuration choices. Particularly, we consider the following:

- **Low (θ_L) (BM25):** We use BM25 [48], a widely adopted lexical model that relies on exact term matching. It is highly efficient but has no semantic awareness.
- **Medium (θ_M) (E5):** We employ *e5-large-v2* [66], a dense bi-encoder trained using weakly supervised contrastive learning, which captures semantic relationships under vocabulary mismatch and is more computationally expensive than lexical one.
- **High (θ_H) (E5%50 >> BGE):** As the next level of complexity, we employ a retrieve-and-rank model, where we use E5 to retrieve the top-50 documents, which are then re-ranked against the query using BGE, a cross-encoder model [37].
- **Extra-High (θ_{XH}) (QD >> E5%50 >> BGE):** To address multi-hop information needs, we implement a decompose-and-rerank pipeline. The query is broken into s atomic sub-queries [1]. We retrieve the top-50 documents for each sub-query and the original query using E5, creating a candidate pool of $(s + 1) \times 50$ documents, which are then re-ranked against the original query using BGE [37]. This incurs the highest computational cost but maximizes relevance for complex tasks.

5.3 Generators Investigated

The space of generator configurations should span varying levels of reasoning capability and computational cost, enabling the routing policy to adapt generation effort based on query complexity. We evaluate three open-source LLM families that provide distinct mechanisms for scaling test-time compute [56]. Each backbone exposes configurable *reasoning-effort* settings that control the extent of internal chain-of-thought reasoning and inference-time computation. Increasing the reasoning effort generally improves performance on compositional and multi-hop queries, albeit at the cost of higher token usage and increased inference latency.

- **GPT-OSS** [42] considers the *reasoning effort* parameter with three settings, **Low** (minimal internal thought, lowest latency), **Medium** (moderate reasoning budget), and **High** (extensive reasoning paths, highest cost but boost in downstream performance).
- **Qwen-3** [61] considers a binary thinking-mode switch, **Low** (Thinking-Off) generates a direct response as a standard instruction-tuned LLM, while **High** (Thinking-On) emits an explicit step-by-step reasoning trace before the final answer, improving multi-hop performance at the cost of a linearly scaled token budget.
- **Gemma-4** [60] similar to the previous setup, it also considers binary thinking control with **Low** (direct response) and **High** (reasoning-trace-enabled) modes.

5.4 Baselines

We compare our adaptive methodology against the following baselines to evaluate the benefits of dynamic RAG configurations:

- **0-shot:** In this baseline, we directly predict the answer to a particular query without any supporting document, serves as lower bound. It shows the impact of generator’s parametric knowledge.

Ret / Gen	GPT-OSS (20B)									Qwen (8B)						Gemma (26B)					
	EM			F1			Accuracy			EM		F1		Accuracy		EM		F1		Accuracy	
	L	M	H	L	M	H	L	M	H	L	H	L	H	L	H	L	H	L	H	L	H
TriviaQA																					
L	53.3 ⁺	55.1 ⁺	56.7 ⁺	63.5 ⁺	64.7 ⁺	64.8 ⁺	65.4 ⁺	66.4 ⁺	67.4 ⁺	57.4 ⁺	57.1 ⁺	65.5 ⁺	64.9 ⁺	66.0 ⁺	64.8 ⁺	49.6 ⁺	52.1 ⁺	59.1 ⁺	61.4 ⁺	61.6 ⁺	63.4 ⁺
M	59.6 ⁺	62.3 ⁺	64.4 ⁺	71.4 ⁺	73.4 ⁺	73.7 ⁺	74.4 ⁺	76.0 ⁺	77.1 ⁺	64.0 ⁺	64.4 ⁺	73.5 ⁺	73.4 ⁺	74.3 ⁺	74.1 ⁺	56.3 ⁺	58.8 ⁺	66.8 ⁺	69.0 ⁺	70.8 ⁺	72.4 ⁺
H	61.5 ⁺	64.2 ⁺	66.1 ⁺	73.3 ⁺	75.4 ⁺	75.7 ⁺	76.3 ⁺	77.9 ⁺	78.4 ⁺	66.1 ⁺	66.8 ⁺	75.7 ⁺	75.9 ⁺	76.5 ⁺	76.5 ⁺	58.6 ⁺	60.1 ⁺	69.3 ⁺	71.1 ⁺	73.9 ⁺	76.2 ⁺
XH	60.8 ⁺	63.2 ⁺	65.2 ⁺	72.6 ⁺	74.3 ⁺	74.8 ⁺	75.6 ⁺	77.1 ⁺	77.9 ⁺	64.9 ⁺	65.6 ⁺	74.3 ⁺	74.1 ⁺	75.4 ⁺	74.7 ⁺	57.6 ⁺	59.7 ⁺	68.3 ⁺	70.5 ⁺	72.6 ⁺	74.1 ⁺
DRAG _{QPP}	65.4			73.9			76.7			64.9		74.3		75.1		58.5		68.4		74.3	
DRAG _{SFT}	68.1			77.1			79.6			67.1		76.2		76.9		61.2		71.9		76.8	
HotpotQA																					
L	25.5 ⁺	28.8 ⁺	30.2 ⁺	35.6 ⁺	38.1 ⁺	38.9 ⁺	35.8 ⁺	36.4 ⁺	37.4 ⁺	26.7 ⁺	27.8 ⁺	35.5 ⁺	35.4 ⁺	33.4 ⁺	33.5 ⁺	25.2 ⁺	26.8 ⁺	33.1 ⁺	34.4 ⁺	33.1 ⁺	33.9 ⁺
M	38.5 ⁺	43.9 ⁺	45.5 ⁺	53.4 ⁺	57.4 ⁺	57.9 ⁺	53.0 ⁺	54.2 ⁺	54.7 ⁺	41.49 ⁺	45.4 ⁺	53.3 ⁺	57.0 ⁺	49.5 ⁺	52.3 ⁺	41.6 ⁺	44.3 ⁺	53.5 ⁺	56.1 ⁺	52.3 ⁺	54.0 ⁺
H	44.0 ⁺	49.9 ⁺	51.6 ⁺	60.1 ⁺	64.5 ⁺	65.2 ⁺	59.5 ⁺	60.9 ⁺	61.5 ⁺	46.4 ⁺	50.9 ⁺	59.6 ⁺	63.9 ⁺	55.2 ⁺	58.7 ⁺	46.9 ⁺	50.5 ⁺	60.4 ⁺	63.8 ⁺	59.2 ⁺	61.8 ⁺
XH	41.2 ⁺	47.4 ⁺	48.8 ⁺	57.3 ⁺	61.6 ⁺	62.3 ⁺	57.0 ⁺	57.7 ⁺	58.6 ⁺	43.9 ⁺	48.1 ⁺	56.5 ⁺	60.3 ⁺	52.5 ⁺	53.3 ⁺	44.6 ⁺	50.5 ⁺	57.5 ⁺	62.7 ⁺	56.6 ⁺	60.1 ⁺
DRAG _{QPP}	49.2			63.1			60.8			48.2		61.3		55.9		48.4		61.2		59.4	
DRAG _{SFT}	53.2			67.6			63.8			50.5		63.9		58.2		51.1		64.1		62.2	
MuSiQue																					
L	7.1 ⁺	7.2 ⁺	7.6 ⁺	11.4 ⁺	11.4 ⁺	11.7 ⁺	10.1 ⁺	10.6 ⁺	12.0 ⁺	3.9 ⁺	5.5 ⁺	7.9 ⁺	9.2 ⁺	6.5 ⁺	8.2 ⁺	4.4 ⁺	4.5 ⁺	7.6 ⁺	7.3 ⁺	7.5 ⁺	7.1 ⁺
M	14.3 ⁺	17.8 ⁺	18.5 ⁺	22.1 ⁺	25.0 ⁺	25.3 ⁺	20.3 ⁺	23.3 ⁺	25.8 ⁺	9.6 ⁺	15.8 ⁺	15.7 ⁺	22.1 ⁺	13.4 ⁺	19.8 ⁺	11.8 ⁺	14.8 ⁺	17.3 ⁺	20.0 ⁺	16.7 ⁺	20.0 ⁺
H	19.5 ⁺	21.1 ⁺	22.8 ⁺	27.7 ⁺	29.7 ⁺	30.1 ⁺	26.0 ⁺	28.4 ⁺	31.1 ⁺	11.3 ⁺	18.9 ⁺	18.1 ⁺	25.4 ⁺	15.9 ⁺	23.7 ⁺	15.4 ⁺	17.3 ⁺	20.9 ⁺	22.9 ⁺	21.0 ⁺	23.9 ⁺
XH	18.6 ⁺	21.6 ⁺	22.1 ⁺	26.8 ⁺	29.8 ⁺	29.3 ⁺	25.1 ⁺	28.1 ⁺	30.5 ⁺	11.0 ⁺	17.9 ⁺	17.6 ⁺	24.0 ⁺	15.3 ⁺	22.2 ⁺	14.3 ⁺	17.7 ⁺	19.8 ⁺	23.2 ⁺	19.5 ⁺	25.0 ⁺
DRAG _{QPP}	19.5			28.4			28.4			17.1		23.3		20.8		15.9		22.8		23.0	
DRAG _{SFT}	24.5			31.9			32.3			19.3		26.2		24.4		19.2		24.6		25.4	
FRAMES																					
L	14.2 ⁺	13.2 ⁺	12.1 ⁺	22.9 ⁺	20.6 ⁺	18.5 ⁺	18.2 ⁺	17.5 ⁺	17.7 ⁺	8.1 ⁺	12.6 ⁺	13.9 ⁺	19.5 ⁺	9.8 ⁺	13.8 ⁺	6.3 ⁺	9.5 ⁺	11.2 ⁺	15.0 ⁺	8.1 ⁺	11.3 ⁺
M	18.3 ⁺	18.1 ⁺	18.2 ⁺	28.4 ⁺	27.2 ⁺	27.1 ⁺	21.5 ⁺	22.2 ⁺	25.1 ⁺	12.6 ⁺	18.0 ⁺	20.2 ⁺	26.5 ⁺	14.4 ⁺	18.9 ⁺	10.6 ⁺	15.3 ⁺	17.9 ⁺	23.3 ⁺	12.6 ⁺	17.6 ⁺
H	19.4 ⁺	19.5 ⁺	17.6 ⁺	28.6 ⁺	28.8 ⁺	26.3 ⁺	22.9 ⁺	23.3 ⁺	23.5 ⁺	10.7 ⁺	17.0 ⁺	19.1 ⁺	24.9 ⁺	12.1 ⁺	18.0 ⁺	11.7 ⁺	16.8 ⁺	18.4 ⁺	24.0 ⁺	13.8 ⁺	19.0 ⁺
XH	21.8 ⁺	21.1 ⁺	20.4 ⁺	32.1 ⁺	30.6 ⁺	29.6 ⁺	26.5 ⁺	25.2 ⁺	26.0 ⁺	11.3 ⁺	18.6 ⁺	19.6 ⁺	27.6 ⁺	13.1 ⁺	20.3 ⁺	12.0 ⁺	17.8 ⁺	19.0 ⁺	24.3 ⁺	13.8 ⁺	18.3 ⁺
DRAG _{QPP}	19.3			29.5			23.9			15.8		24.8		18.4		15.3		22.9		16.3	
DRAG _{SFT}	23.3			33.9			27.2			19.5		28.1		21.3		18.6		25.8		18.9	

Table 3: End-to-end RAG performance of three LLMs across four QA benchmarks under all retriever-generator configurations. Rows correspond to retriever complexity levels ($\theta \in \{L, M, H, XH\}$), while columns correspond to generator complexity levels ($\phi \in \{L, M, H\}$ for GPT-OSS and $\phi \in \{L, H\}$ for Qwen and Gemma). We report Exact Match (EM), F1, Accuracy, and average per-query inference time. Among static configurations, **bold numbers denote the best result within each retriever block (best generator for a fixed retriever), while underlined numbers denote the best result within each generator block (best retriever for a fixed generator). We additionally bold the best of the two proposed adaptive methods. Asterisks (*) and Dagger (†) indicate statistically significant differences compared to DRAG_{QPP} and DRAG_{SFT} respectively ($p < 0.05$). All reported results use 5 contextual examples.**

- **RAG_{max}**: This represents a standard static RAG setup, where a fixed configuration is uniformly applied to all queries, and performance is reported for the best-performing configuration over the dataset.
- **RAG_{Orac}**: This represents the theoretical upper bound of an ideal adaptive RAG system as mentioned in Equation 1 of Section 3 (Efficiency-aware Oracle Configuration).
- **DRAG_{AP}**: Our routing policy of RPP in DRAG_{QPP} (in Equation 2) is dependent on pre-retrieval QPP, thus we define an upper-bound by replacing QPP with the ground-truth retrieval effectiveness (AP@50) of each retriever. This yields a retrieval-aware, sequential routing strategy that estimates the maximum achievable gain under perfect effectiveness signals.
- **DRAG_{PPT}**: In this case, given a query, the generator is provided with a natural language description of the available retriever-generator configurations and is asked to select the most suitable one, relying solely on parametric knowledge rather than retrieved evidence. The chosen configuration is then used to generate the final answer. This shows whether an LM can infer appropriate configurations from the query’s information need. This is a basic baseline from DRAG’s perspective.

5.5 Experimental Details

Evaluation Metrics. We assess the **effectiveness of the downstream task** using three standard metrics: Exact Match (EM), F1, and Accuracy (Acc), which capture exact correctness, partial lexical overlap, and answer containment, respectively. We report **statistical significance** at the query level using McNemar’s test for EM and Accuracy (due to comparison between binary outcomes), and a paired t -test for F1, which captures the comparison between systems. The **retrieval quality** is evaluated using AP@10 to assess the relevance of the retrieved contexts provided to the generator.

Hyperparameter Details. For Extra-High mode of retriever we use same the target LLM to decompose each query into atomic sub-queries (temperature = 0.1). In this paper, for RPP module in DRAG_{QPP} we employ Avg-IDF as we pre-retrieved QPP which is computed over query terms with respect to the corpus. It uses thresholds set of 75th, 50th, and 25th percentiles for distinguishing all four retriever levels. For GPP, we use threshold set of perplexity scores at the 20th percentile of the training distribution (where lower is better). Additionally, for DRAG_{SFT} we used Qwen3-4B-Instruct. We train for 5 epochs with a learning rate of 2e-5, a cosine decay schedule with warmup ratio 0.15, weight decay 0.05, and an effective batch size of 256. The maximum sequence length is set to 1,280 tokens. The best checkpoint is selected based on combined (retrieval

	Model	TriviaQA				HotpotQA				MuSiQue				FRAMES (OOD)			
		EM	F1	Acc	t(s)	EM	F1	Acc	t(s)	EM	F1	Acc	t(s)	EM	F1	Acc	t(s)
GPT-OSS	RAG _{Orac}	77.5	86.1	85.5	2.56	63.9	76.9	71.9	1.83	37.1	44.5	43.3	5.36	38.9	52.4	44.3	4.89
	RAG _{max}	66.1	75.7	78.4	10.73	51.6	65.2	61.5	4.36	22.8	30.1	31.1	10.2	21.8	32.1	26.5	5.14
	0-shot	42.5	52.3	53.0	1.02	20.7	27.9	34.8	0.83	5.4	10.4	21.6	1.73	10.9	17.6	23.8	1.72
	DRAG _{AP}	67.3	76.2	78.9	5.74	52.3	67.9	64.2	2.67	24.6	32.2	32.5	6.43	22.8	33.7	27.8	3.20
	DRAG _{PPT}	52.7	65.8	65.2	5.12	43.7	57.9	55.3	2.87	16.8	26.7	26.9	6.45	16.4	27.3	21.1	3.22
	DRAG _{QPP}	65.4	73.9	76.7	5.14	49.2	63.1	60.8	2.18	19.5	28.4	28.4	6.36	19.3	29.5	23.9	3.15
Qwen	DRAG _{SFT}	<u>68.1</u>	<u>77.1</u>	<u>79.6</u>	5.37	<u>53.2</u>	<u>67.6</u>	<u>63.8</u>	3.11	<u>24.5</u>	<u>31.9</u>	<u>32.3</u>	6.85	<u>23.3</u>	<u>33.9</u>	<u>27.2</u>	3.28
	RAG _{Orac}	76.0	84.3	83.3	2.34	61.8	75.3	68.9	1.13	29.6	39.8	35.3	1.65	31.8	45.1	33.7	0.78
	RAG _{max}	66.8	75.9	76.5	8.76	50.9	63.9	58.7	3.16	18.9	25.4	23.7	8.64	18.6	27.5	20.3	4.39
	0-shot	50.6	58.3	56.5	0.82	21.3	29.8	24.1	0.86	4.9	11.6	7.5	1.07	10.4	17.9	11.5	1.41
	DRAG _{AP}	67.2	75.6	76.3	4.37	50.8	63.9	58.9	1.28	19.9	26.5	24.9	4.87	18.8	27.8	21.1	2.30
	DRAG _{PPT}	54.7	67.8	66.5	4.35	46.3	58.4	54.4	1.18	15.3	21.6	19.2	4.76	12.7	20.5	16.8	1.93
Gemma	DRAG _{QPP}	64.9	74.3	75.1	4.10	48.2	61.3	55.9	1.21	17.1	23.3	20.8	4.23	15.8	24.8	18.4	1.84
	DRAG _{SFT}	<u>67.1</u>	<u>76.2</u>	<u>76.9</u>	4.46	<u>50.5</u>	<u>63.9</u>	<u>58.2</u>	1.37	<u>19.3</u>	<u>26.2</u>	<u>24.4</u>	5.10	<u>19.5</u>	<u>28.1</u>	<u>21.3</u>	2.14
	RAG _{Orac}	76.5	85.4	84.2	2.86	63.3	76.7	72.2	1.32	27.7	36.1	36.1	6.87	27.3	38.2	29.8	4.36
	RAG _{max}	60.1	71.1	76.2	10.86	50.4	63.8	61.8	4.87	17.7	23.2	24.9	11.13	17.7	24.3	18.3	5.68
	0-shot	55.8	64.3	66.9	1.32	27.1	36.8	30.3	1.09	9.4	18.8	11.9	1.04	16.3	26.1	18.0	1.55
	DRAG _{AP}	61.4	71.7	75.8	5.7	50.8	63.4	60.5	2.7	17.4	24.4	25.1	7.7	18.2	25.5	18.6	3.4
Gemma	DRAG _{PPT}	51.8	61.6	64.5	3.77	45.8	59.3	57.3	2.1	15.2	22.3	22.8	7.3	15.1	21.2	16.5	3.1
	DRAG _{QPP}	58.5	68.4	74.3	5.34	48.4	61.2	59.4	2.2	15.9	22.8	23.0	7.16	15.3	22.9	16.3	3.35
	DRAG _{SFT}	<u>61.2</u>	<u>71.9</u>	<u>76.8</u>	5.67	<u>51.1</u>	<u>64.1</u>	<u>62.2</u>	2.30	<u>19.2</u>	<u>24.6</u>	<u>25.4</u>	7.34	<u>18.6</u>	<u>25.8</u>	<u>18.9</u>	3.51

Table 4: We compare static, oracle, and adaptive RAG configurations across TriviaQA, HotpotQA, MuSiQue, and FRAMES. Our adaptive methods, DRAG_{QPP} (training-free) and DRAG_{SFT} (supervised), select retriever-generator from available configurations. We report EM, F1-score, Accuracy, and average inference latency (time/second, represented in table as t(s)). The number of contexts used for few-shot experiments is 5. All evaluations are conducted on the full test sets to ensure statistical robustness. **Bold denotes the best result among static and adaptive methods, while underline indicates the best over our proposed methods (i.e., grey portion).**

+ generator) macro-F1. For generation, we use GPT-OSS (20B¹), Qwen3 (8B²) and Gemma (26B³) both deployed using the vLLM inference engine. To accommodate the necessary reasoning tokens required by these strategies, we set the maximum generation budget (max_new_tokens) to 512, 2048, and 4096 for the **Low**, **Medium**, and **High** modes, respectively. All experiments are conducted on NVIDIA A100 GPUs with a batch size of 12.

6 Results and Discussion

We present our experimental results to answer the proposed research questions in Section 5.

6.1 Effect of Retriever and Generator Complexity

Table 3 reports the performance of each retriever-generator pair in terms of EM, F1-score, and Accuracy across all four QA benchmarks using three different families of LLMs. To address **RQ1**, we examine how varying the complexity *individually* influences end-to-end RAG performance, and whether scaling either component separately consistently yields gains.

Overall, we observe that **increasing component complexity leads to performance improvement, but the gains are not monotonic**. When we fix the choice of generators and increase

the retriever complexity, performance generally improves from lexical retrieval (θ_L) to dense (θ_M) and reranking-based pipelines (θ_H), with gains in all datasets, particularly higher gains for multi-hop benchmarks such as HotpotQA, MuSiQue, and FRAMES, where evidence is dispersed across multiple documents and stronger retrieval is needed. However, it saturates at higher retrieval levels (e.g., $\theta_H \rightarrow \theta_{XH}$) for all datasets except FRAMES, i.e., **diminishing returns from retrieval scaling** despite substantially higher computational cost. A likely reason of the effect is less evident in FRAMES is that its queries impose multiple constraints (numerical, temporal, multi-entity) and require evidence assembled from genuinely distinct sub-aspects, so atomic sub-query retrieval supplies a signal that single-pass retrievers miss.

In contrast, increasing generator reasoning effort under a fixed retriever (particularly, for weak retrieval) shows comparatively smaller and inconsistent gains for all datasets. For example, for multi-hop dataset, MuSiQue, increasing reasoning effort under weak retrieval (θ_L) yields only a **7% improvement** in terms of EM, but improves performance by **16.9%** in terms of EM for θ_H . It shows that **reasoning cannot compensate for poor retrieval quality and generator effectiveness is fundamentally constrained by retrieval quality**, with stronger retrieval typically yielding larger gains than increased reasoning effort alone.

Effectiveness-Efficiency Trade-off. Table 3 reports downstream effectiveness, while Table 5 shows per-query latency for static RAG configurations. We observe a clear trade-off between effectiveness and computational cost, i.e., increasing retriever or

¹<https://huggingface.co/openai/gpt-oss-20b>

²<https://huggingface.co/Qwen/Qwen3-8B>

³[google/gemma-4-26B-A4B-it](https://huggingface.co/google/gemma-4-26B-A4B-it)

(a) Static RAG Latency					(b) DRAG Latency Breakdown		
$\phi \backslash \theta$	L	M	H	XH	Time	DRAG _{QPP}	DRAG _{SFT}
L	0.11	0.33	1.04	3.51	Decision	0.28	0.36
M	0.38	0.67	2.34	4.10	RAG	1.90	2.75
H	1.02	1.70	4.36	5.87	Total	2.18	3.11

Table 5: Per-query latency (seconds) on HotpotQA using GPT-OSS. (a) Static retriever-generator configurations across retrieval (θ) and generator (ϕ) complexity levels. (b) Latency breakdown of adaptive methods, where decision time denotes routing overhead (IDF computation and perplexity estimation) and RAG time corresponds to downstream retrieval and generation. Retrieval indexing is excluded, as it is a one-time offline preprocessing cost. Green and red cells indicate, respectively, lower and higher total latency w.r.t DRAG_{SFT}.

generator complexity incurs substantially higher latency while providing only marginal downstream gains. For example, on HotpotQA (GPT-OSS), increasing generator complexity from ϕ_M to ϕ_H under θ_H improves EM only marginally (49.9→51.6), while increasing latency (2.34s→4.36s, per query). Similarly, under θ_{XH} , the same increase very marginally (47.4→48.8) despite a high latency increase (4.10s→5.87s, per query). Moreover, stronger configurations are not always strictly better (e.g., θ_H outperforming θ_{XH}), so **uniformly allocating maximum retrieval and reasoning effort is inefficient**. These findings motivate an **adaptive routing strategy** that dynamically allocates retrieval and generation complexity based on query requirements.

6.2 Dynamic vs. Static RAG Performance

To address RQ2, Table 4 compares our adaptive methods against oracle setup, RAG_{Orac} and other baselines across downstream QA metrics for three LLM families. Being an oracle setup, RAG_{Orac} consistently outperforms all adaptive methods and baselines. This performance gap highlights the difficulty of routing under realistic settings, where configuration decisions must be made with or without supervision. Among the baselines, DRAG_{PPT} consistently outperforms 0-shot across nearly all datasets (except TriviaQA with Gemma), suggesting that directly prompting an LLM to predict the answer is insufficient, whereas explicitly predicting the retriever-generator configuration provides a more appropriate downstream prediction. Furthermore, DRAG_{AP} consistently improves over DRAG_{QPP} across datasets, indicating that retrieval quality estimation remains a key bottleneck for unsupervised adaptive routing. Notably, both DRAG_{SFT} and DRAG_{AP} outperform the static configuration, RAG_{max}, thus query-adaptive selection is more effective than uniformly assigning a fixed high-complexity pipeline to all queries. Finally, DRAG_{SFT} consistently improves over DRAG_{QPP}, suggesting that jointly learning query-specific retriever-generator interactions is more effective than sequential threshold-based routing, albeit at the cost of supervision.

Performance vs Latency. Figure 1 and Tables 4 illustrate the effectiveness and latency (sec/query) values of adaptive and static RAG systems. Overall, increasing retriever and generator complexity improves downstream performance but at substantially higher inference cost, with marginal gains at stronger configurations. We

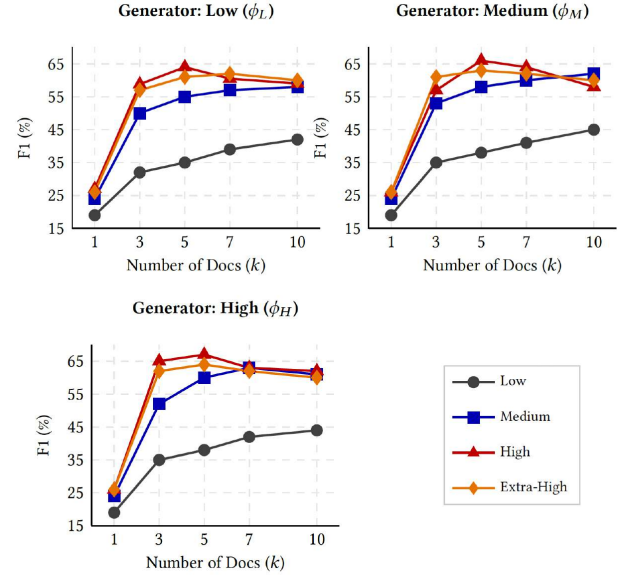


Figure 3: Effect of change in retrieved document count (k) on the performance of HotpotQA dataset. We analyze F1 scores under varying Retriever (θ) and Generator (ϕ) complexities. The results demonstrate how increasing the number of documents impacts performance across different system configurations. Each line corresponds to a retrieval complexity. The shared legend applies to all subplots.

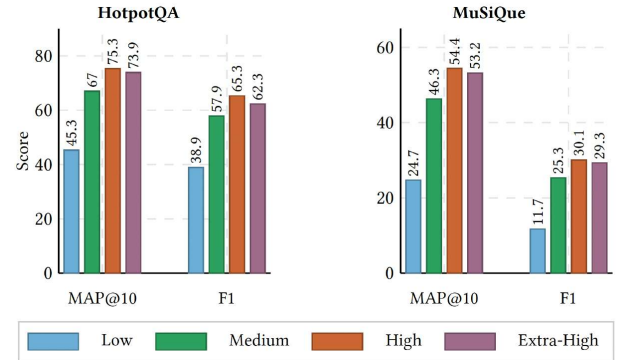


Figure 4: Impact of retriever effectiveness on downstream QA performance using GPT-OSS on HotpotQA and MuSiQue. MAP@10 measures retrieval effectiveness, while F1 measures end-to-end QA performance under static RAG configurations. The bars correspond to retriever modes.

also observe a **clear trade-off emerges between our two proposed DRAG methods**, DRAG_{SFT} consistently achieves better downstream effectiveness than DRAG_{QPP}, typically improving performance by approximately 2-12% across datasets, but incurs a modest increase in latency (approximately 5-40%). While DRAG_{QPP} remains training-free and lightweight, DRAG_{SFT} leverages supervision to model query-specific interactions, resulting in better downstream performance with higher inference cost.

Query and Gold Answer	RAG _{Orac}		DRAG _{SFT}		Failure Type
	Config	Output	Config	Output	
What is Mohamed Atta's car, manufactured by the company that includes Datsun, an instance of? Ans: Nissan Altima	XH / L	<i>a 2001 Nissan Altima</i>	XH / H	<i>a blue rental car</i>	Overestimation
What is the oldest college in the state that contains Borough House Plantation? Ans: College of Charles Towne	H / H	<i>College of Charles Towne</i>	L / M	<i>Columbia University</i>	Underestimation
What is the tallest building in the state where KZAR is located? Ans: JPMorgan Chase Tower	H / H	<i>JPMorgan Chase Tower</i>	L / M	<i>Transamerica Tower</i>	Overestimation
What is the position of the first governor general of India? Ans: Governor-General of India	L / H	<i>Governor-General of British India</i>	L / L	<i>Warren Hastings</i>	Wrong Entity
What team does the person who hit the first home run at the new South Bronx stadium play for? Ans: New York Yankees	M / L	<i>New York Yankees</i>	H / H	<i>Indians</i>	Wrong Entity

Table 6: Qualitative failure cases on MuSiQue dataset. Each row shows a query where DRAG_{SFT} selects the failure/suboptimal cases relative to the oracle RAG_{Orac}, along with the resulting output error. Config columns represent the mode of retriever/generator. never use vspace; the publishers won't allow this anyway

Significance Testing. Table 3 further highlights the statistical behavior of the DRAG framework. Across in-domain datasets, DRAG_{SFT} consistently yields statistically significant gains over static configurations, whereas DRAG_{QPP} often remains statistically comparable to static RAG with medium/high-complexity generators. Under the OOD setting (FRAMES), although DRAG_{SFT} generally maintains stronger performance, improvements are not always statistically significant, suggesting that transferring routing decisions across datasets with different query types remains challenging.

Latency. Table 5 reports the per-query latency breakdown of adaptive RAG systems. We observe that the routing cost, i.e., model configuration prediction cost, is relatively less. Instead, most of the latency difference arises from retrieval and downstream generation (1.90s vs. 2.75s RAG time), here DRAG_{SFT} picks higher-complexity configurations, trading latency for effectiveness, while DRAG_{QPP} favors cheaper settings at the cost of a modest accuracy drop.

6.3 Model Generalization

To address RQ3, we evaluate the robustness of our adaptive methods under an OOD setting using FRAMES, where models don't rely on dataset-specific training (i.e., use training data of MuSiQue and are evaluated directly on FRAMES). From Tables 4 and 3, we observe that both DRAG_{QPP} and DRAG_{SFT} provide better results w.r.t all the baselines, including DRAG_{AP} across all three LLM families. Interestingly, Table 3 shows that for GPT-OSS on FRAMES dataset, the best static RAG performance is achieved under low or medium-reasoning generator settings with a fixed retriever choice, thus uniformly increasing reasoning effort is not always beneficial under distribution shift. In contrast, our adaptive methods not only achieve better downstream effectiveness than RAG_{max} and other baselines, but also incur a lower inference cost. Thus, our proposed framework, DRAG is robust and generalizable across OOD datasets.

6.4 Ablation Studies

Impact of Increasing Context Size. Figure 3 analyzes the sensitivity of RAG performance to the number of retrieved documents k across retrieval complexity levels and generator reasoning modes. While low-complexity retrieval (θ_L) shows steady improvement

with increasing k , reflecting its lower per-document precision and reliance on broader context coverage, higher-complexity retrievers ($\theta_M, \theta_H, \theta_{XH}$) plateau early, as their stronger relevance signals are largely captured within a compact candidate set. This saturation behavior is consistent across all three generator reasoning levels, suggesting it is driven by retrieval quality rather than generation capacity, and collectively justifies our choice of $k = 5$ as a balance between evidence coverage and computational efficiency.

Correlation between retrieval and downstream performance. Figure 4 examines the relationship between retrieval effectiveness (MAP@10) and downstream QA performance (F1) across retrieval complexity levels on HotpotQA and MuSiQue. A clear positive correlation is observed: as retrieval complexity increases from lexical (θ_L) to dense (θ_M) and reranking-based pipelines (θ_H, θ_{XH}), both MAP@10 and F1 improve consistently and proportionally across both benchmarks, confirming that stronger retrieval directly translates to better answer generation, particularly for multi-hop queries where evidence is dispersed across multiple documents.

6.5 Per-Query Analysis

Table 6 presents qualitative failure cases of our best performing adaptive method DRAG_{SFT} (as shown in Table 3) on MuSiQue, revealing three primary error patterns. *Overestimation* occurs when DRAG_{SFT} routes simple queries to unnecessarily expensive configurations (e.g., θ_H/ϕ_{XH} instead of θ_L/ϕ_{XH}), introducing noisy evidence that degrades answer quality. *Underestimation* occurs when DRAG_{SFT} assigns insufficient capacity to genuinely complex queries (e.g., θ_M/ϕ_L instead of θ_H/ϕ_H), leading to factually incorrect answers. Finally, *wrong entity* errors occur when suboptimal configuration selection (e.g., θ_H/ϕ_H instead of θ_L/ϕ_M) causes the model to retrieve incorrect evidence, leading to entity confusion during multi-hop reasoning. Together, these cases highlight that routing errors are systematically tied to query complexity estimation, and that better query representations could further improve adaptive configuration selection.

7 Conclusions and Future Work

We presented a systematic study of how retrieval complexity and generation effort jointly affect end-to-end RAG performance. Our

results show that increasing complexity does not consistently improve accuracy, and that high-complexity configurations are not strict supersets of simpler ones. This highlights the non-monotonic, query-dependent nature of RAG and the limitations of fixed retriever-generator settings. To address this, we introduced a training-free approach (DRAG_{QPP}) and a supervised approach (DRAG_{SFT}) under a common framework DRAG for per-query retriever-generator selection. Across multiple QA benchmarks, DRAG_{QPP} matches strong static baselines while reducing latency, and DRAG_{SFT} further improves the effectiveness-efficiency trade-off, achieving performance close to oracle routing.

Future work will focus on developing more expressive routing policies that better capture the non-separable interactions between retrieval and generation, including richer supervision and reinforcement learning for optimizing cost-quality trade-offs.

GenAI Usage Disclosure

Generative AI tools were not used for core idea generation or experimental design. Its use was limited to minor writing and formatting.

References

- [1] Paul J. L. Ammann, Jonas Golde, and Alan Akbik. 2025. Question Decomposition for Retrieval-Augmented Generation. In *ACL (4)*. Association for Computational Linguistics, 497–507.
- [2] Negar Arabzadeh, Andrew Drozdov, Michael Bendersky, and Matei Zaharia. 2026. Can QPP Choose the Right Query Variant? Evaluating Query Variant Selection for RAG Pipelines. *CoRR* abs/2604.22661 (2026).
- [3] Akari Asai, Zequi Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *ICLR*. OpenReview.net.
- [4] Adrian-Gabriel Chifu, Léa Laporte, Josiane Mothe, and Md. Zia Ullah. 2018. Query Performance Prediction Focused on Summarized Letor Features. In *SIGIR*. ACM, 1177–1180.
- [5] Ronan Cummins, Joemon M. Jose, and Colm O’Riordan. 2011. Improved query performance prediction using standard deviation. In *SIGIR*. ACM, 1089–1090.
- [6] Robert Dale. 2021. GPT-3: What’s it good for? *Nat. Lang. Eng.* 27, 1 (2021), 113–118.
- [7] Suchana Datta, Debasis Ganguly, Derek Greene, and Mandar Mitra. 2022. Deep-QPP: A Pairwise Interaction-based Deep Learning Model for Supervised Query Performance Prediction. In *WSDM*. ACM, 201–209.
- [8] Suchana Datta, Debasis Ganguly, Sean MacAvaney, and Derek Greene. 2024. A Deep Learning Approach for Selective Relevance Feedback. In *ECIR (2) (Lecture Notes in Computer Science)*. Springer, 189–204.
- [9] Suchana Datta, Debasis Ganguly, Mandar Mitra, and Derek Greene. 2023. A Relative Information Gain-based Query Performance Prediction Framework with Generated Query Variants. *ACM Trans. Inf. Syst.* 41, 2 (2023), 38:1–38:31.
- [10] Claude De Loupy and Patrice Bellot. 2000. Evaluation of document retrieval systems and query difficulty. In *LREC Workshop, Athens*. 32–39.
- [11] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 2368–2378. <https://doi.org/10.18653/v1/N19-1246>
- [12] Sajad Ebrahimi, Maryam Khodabakhsh, Negar Arabzadeh, and Ebrahim Bagheri. 2024. Estimating Query Performance Through Rich Contextualized Query Representations. In *ECIR (4) (Lecture Notes in Computer Science)*. Springer, 49–58.
- [13] Matteo Gabburo, Nicolaas Paul Jedema, Siddhant Garg, Leonardo F. R. Ribeiro, and Alessandro Moschitti. 2024. Measuring Retrieval Complexity in Question Answering Systems. In *ACL (Findings) (Findings of ACL)*. Association for Computational Linguistics, 14636–14650.
- [14] Debasis Ganguly, Johannes Leveling, Walid Magdy, and Gareth J. F. Jones. 2011. Patent query reduction using pseudo relevance feedback. In *CIKM*. 1953–1956.
- [15] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023).
- [16] Michael R. Glass, Gaetano Rossiello, Md. Faisal Mahub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gizzo. 2022. Re2G: Retrieve, Rerank, Generate. In *NAACL-HLT*. Association for Computational Linguistics, 2701–2715.
- [17] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Augmented Language Model Pre-Training. In *ICML (Proceedings of Machine Learning Research)*. PMLR, 3929–3938.
- [18] Ben He and Iadh Ounis. 2006. Query performance prediction. *Inf. Syst.* 31, 7 (2006), 585–594.
- [19] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *EACL*. Association for Computational Linguistics, 874–880.
- [20] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *J. Mach. Learn. Res.* 24 (2023), 251:1–251:43.
- [21] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *NAACL-HLT*. Association for Computational Linguistics, 7036–7050.
- [22] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12 (2023), 248:1–248:38.
- [23] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. TempQuestions: A Benchmark for Temporal Question Answering. In *Companion Proceedings of the The Web Conference 2018 (Lyon, France) (WWW ’18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1057–1062. <https://doi.org/10.1145/3184558.3191536>
- [24] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *EMNLP*. Association for Computational Linguistics, 7969–7992.
- [25] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL (1)*. Association for Computational Linguistics, 1601–1611.
- [26] Evangelos Kanoulas, Ben Carterette, Paul D. Clough, and Mark Sanderson. 2011. Evaluating multi-query sessions. In *SIGIR*. ACM, 1053–1062.
- [27] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP (1)*. Association for Computational Linguistics, 6769–6781.
- [28] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *ICLR*. OpenReview.net.
- [29] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 39–48. <https://doi.org/10.1145/3397271.3401075>
- [30] Michael Kleimman, Matthew Trager, Alessandro Achille, Wei Xia, and Stefano Soatto. 2025. e1: Learning Adaptive Control of Reasoning Effort. *CoRR* abs/2510.27042 (2025).
- [31] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohanane, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqi. 2025. Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation. In *NAACL (Long Papers)*. Association for Computational Linguistics, 4745–4759.
- [32] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.
- [33] K. L. Kwok. 1996. A New Method of Weighting Query Terms for Ad-Hoc Retrieval. In *SIGIR*. ACM, 187–195.
- [34] Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is All You Need. In *ICLR*. OpenReview.net.
- [35] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*.
- [36] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: Retrieval-Augmented Dual Instruction Tuning. In *ICLR*. OpenReview.net.
- [37] Kun Luo, Zheng Liu, Shitao Xiao, and Kang Liu. 2024. BGE Landmark Embedding: A Chunking-Free Embedding Method For Retrieval Augmented Long-Context Large Language Models. *CoRR* abs/2402.11573 (2024).
- [38] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *ACL (1)*. Association for Computational Linguistics, 9802–9822.

- [39] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked List Truncation for Large Language Model-based Re-Ranking. In *SIGIR*. ACM, 141–151.
- [40] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2025. Query Performance Prediction Using Relevance Judgments Generated by Large Language Models. *ACM Trans. Inf. Syst.* 43, 4 (2025), 106:1–106:35.
- [41] Josiane Mothe and Ludovic Tanguy. 2005. Linguistic features to predict query difficulty. In *SIGIR, Predicting query difficulty-methods and applications workshop*. 7–10.
- [42] OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. *CoRR* abs/2508.10925 (2025).
- [43] Andrew Parry, Debasis Ganguly, and Manish Chandra. 2024. "In-Context Learning" or: How I learned to stop worrying and love "Applied Information Retrieval". In *SIGIR*. ACM, 14–25.
- [44] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. 2026. Multi-Step Reasoning with Large Language Models, a Survey. *ACM Comput. Surv.* 58, 6 (2026), 160:1–160:35.
- [45] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. *CoRR* abs/2101.05667 (2021).
- [46] Ronak Pradeep, Sahel Sharifmoghadam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *CoRR* abs/2312.02724 (2023).
- [47] Peng Qi, Haejun Lee, Tg Sido, and Christopher D. Manning. 2021. Answering Open-Domain Questions of Varying Reasoning Steps from Text. In *EMNLP (1)*. Association for Computational Linguistics, 3599–3614.
- [48] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Documentation* 60, 5 (2004), 503–520.
- [49] Hagga Roitman, Shai Erera, and Bar Weiner. 2017. Robust Standard Deviation Estimation for Query Performance Prediction. In *ICTIR*. ACM, 245–248.
- [50] Payel Santra, Madhusudan Ghosh, Debasis Ganguly, Partha Basuchowdhuri, and Sudip Kumar Naskar. 2025. HF-RAG: Hierarchical Fusion-based RAG with Multiple Sources and Rankers. In *CIKM*. ACM, 5202–5207.
- [51] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-Augmented Black-Box Language Models. In *NAACL-HLT*. Association for Computational Linguistics, 8371–8384.
- [52] Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. 2026. A Survey on Uncertainty Quantification of Large Language Models: Taxonomy, Open Research Challenges, and Future Directions. *ACM Comput. Surv.* 58, 3 (2026), 63:1–63:38.
- [53] Anna Shtok, Oren Kurland, and David Carmel. 2010. Using statistical decision theory and relevance models for query-performance prediction. In *SIGIR*. ACM, 259–266.
- [54] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting Query Performance by Query-Drift Estimation. *ACM Trans. Inf. Syst.* 30, 2 (2012), 11:1–11:35.
- [55] Nilanjan Sinhababu, Andrew Parry, Debasis Ganguly, Debasis Samanta, and Pabitra Mitra. 2024. Few-shot Prompting for Pairwise Ranking: An Effective Non-Parametric Retrieval Model. In *EMNLP (Findings)* (*Findings of ACL*). Association for Computational Linguistics, 12363–12377.
- [56] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *CoRR* abs/2408.03314 (2024).
- [57] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 641–651. <https://doi.org/10.18653/v1/N18-1059>
- [58] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiT-ing Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. *CoRR* abs/2312.16098 (2023).
- [59] Yongquan Tao and Shengli Wu. 2014. Query Performance Prediction By Considering Score Magnitude and Variance Together. In *CIKM*. ACM, 1891–1894.
- [60] Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. 2026. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770* (2026).
- [61] Qwen Team. 2025. Qwen3 Technical Report. *CoRR* abs/2505.09388 (2025).
- [62] Fangzheng Tian, Debasis Ganguly, and Craig Macdonald. 2026. Predicting Retrieval Utility and Answer Quality in Retrieval-Augmented Generation. In *ECIR (1) (Lecture Notes in Computer Science)*. Springer, 368–385.
- [63] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *ArXiv preprint abs/2302.13971* (2023). <https://arxiv.org/abs/2302.13971>
- [64] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♣ MuSiQue: Multihop Questions via Single-hop Question Composition. *Trans. Assoc. Comput. Linguistics* 10 (2022), 539–554.
- [65] Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoyebi, and Bryan Catanzaro. 2024. InstructRetro: Instruction Tuning post Retrieval-Augmented Pretraining. In *ICML (Proceedings of Machine Learning Research)*. PMLR / OpenReview.net, 51255–51272.
- [66] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *CoRR* abs/2212.03533 (2022).
- [67] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving Text Embeddings with Large Language Models. In *ACL (1)*. Association for Computational Linguistics, 11897–11916.
- [68] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *ICLR*. OpenReview.net.
- [69] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2025. A Theory for Token-Level Harmonization in Retrieval-Augmented Generation. In *ICLR*. OpenReview.net.
- [70] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*. Association for Computational Linguistics, 2369–2380.
- [71] Mohammad Yani and Adila Alfa Krisnadhi. 2021. Challenges, Techniques, and Trends of Simple Knowledge Graph Question Answering: A Survey. *Inf.* 12, 7 (2021), 271.
- [72] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. 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. https://openreview.net/pdf?id=WE_vluYUL-X
- [73] Qinyuan Ye, Iz Beltagy, Matthew E. Peters, Xiang Ren, and Hannaneh Hajishirzi. 2023. FiD-ICL: A Fusion-in-Decoder Approach for Efficient In-Context Learning. In *ACL '23 (1)*. 8158–8185.
- [74] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. 2018. Neural Query Performance Prediction using Weak Supervision from Multiple Signals. In *SIGIR'18*. 105–114.
- [75] Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023. RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation. In *EMNLP'23*. 2471–2484.
- [76] Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2024. A Two-Stage Adaptation of Large Language Models for Text Ranking. In *ACL '24 (Findings)*. 11880–11891.
- [77] Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training Language Models with Memory Augmentation. In *EMNLP*. Association for Computational Linguistics, 5657–5673.
- [78] Yun Zhou and W. Bruce Croft. 2007. Query performance prediction in web search environments. In *SIGIR*. ACM, 543–550.
- [79] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. *CoRR* abs/2101.00774 (2021).
- [80] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *SIGIR*. ACM, 2308–2313.
- [81] Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-R1: Enhancing Reasoning in LLM-based Document Rerankers via Reinforcement Learning. *CoRR* abs/2503.06034 (2025).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009