CARP-RAG: Compute-Aware Adaptive Retrieval for Efficient Retrieval-Augmented Generation

Abhishek Kumar
Department of Information Technology
University of the Cumberlands
Williamsburg, KY, USA
ORCID: 0009-0001-9987-0510
akumar48250@ucumberlands.edu

***Abstract*—Retrieval-augmented generation (RAG) commonly uses a fixed retrieval depth even though evidence requirements vary across queries. This paper presents CARP-RAG, a compute-aware retrieval planning framework that treats retrieval depth as a controllable inference parameter. We evaluate two lightweight planner instantiations without retraining the retriever or generator: a rule-based latency-budget planner and a score-aware context-allocation gate. On a 500-query SQuAD workload, static $k = 3$ achieves the best F1 (0.5720), while increasing k to 5–20 does not improve answer quality. Under a 0.040-s budget, the rule-based CARP planner selects an average depth of 4.292 and achieves F1 = 0.5645, compared with F1 = 0.5624 for static $k = 5$. To test whether adaptation is useful on heterogeneous evidence requirements, we additionally evaluate a frozen score-aware planner on 500 held-out HotpotQA queries after calibration on a disjoint 500-query development split. Relative to static $k = 10$, Score-CARP reduces propagated passages by 11.34% (average $k = 8.866$) while supporting-document recall changes from 0.822 to 0.805 and complete-support retrieval from 0.672 to 0.640. Paired bootstrap 95% confidence intervals for these differences are $[-0.026, -0.009]$ and $[-0.048, -0.018]$, respectively. The results show that adaptive retrieval exposes a measurable quality–compute tradeoff and that retrieval demand varies substantially across multi-hop queries.**

***Index Terms*—Retrieval-augmented generation, adaptive retrieval, compute-aware inference, multi-hop question answering, vector search, FAISS, performance engineering for AI.**

I. Introduction

Retrieval-augmented generation (RAG) improves the factual grounding and updateability of language model outputs by combining parametric models with retrieved non-parametric memory [1], [2]. A typical RAG pipeline embeds a query, retrieves top-ranked passages from a vector index, augments the prompt or context, and produces an answer. This modular design supports open-domain, enterprise, scientific, and financial question answering [8], [9].

A common systems inefficiency is that retrieval depth k is often fixed for every query. Fixed-depth retrieval is easy to deploy, but it assumes that all questions require the same amount of evidence. Simple factoid questions may be answerable from a few passages, whereas comparison, causal, or multi-hop questions may require broader retrieval. Prior work further shows that indiscriminate retrieval can introduce irrelevant evidence [3] and that long contexts are not always used reliably by downstream models [5]. These observations motivate a systems question: *can retrieval depth be adapted to query-specific evidence demand while explicitly controlling inference cost?*

CARP-RAG addresses this question by treating retrieval depth as a control variable rather than a fixed hyperparameter. The framework supports lightweight planners that can operate before retrieval or after a small candidate search. In this paper, we study two concrete instantiations. *Rule-CARP* uses transparent query-complexity indicators and an empirical latency budget to constrain k. *Score-CARP* retrieves a top-20 candidate pool, uses its similarity-score profile together with lightweight query features to estimate whether shallow retrieval is sufficient, and propagates either $k = 3$ or $k = 10$ passages downstream. The latter planner is calibrated on development data only and frozen before held-out evaluation.

The experiments are designed to separate two questions that a single benchmark cannot answer well. SQuAD v1.1 [13] provides a controlled extractive QA workload for measuring latency and answer-quality behavior under different fixed depths and latency budgets. HotpotQA [14] provides heterogeneous, multi-hop evidence requirements and supporting-document supervision for directly testing retrieval-depth adaptation.

The main contributions are:

- We formulate retrieval depth as a compute-aware inference control variable and present CARP-RAG as a lightweight planning layer that does not retrain the underlying retriever or generator.
- We evaluate a transparent rule-based latency-budget planner on 500 SQuAD queries and show that larger retrieval depth does not monotonically improve extractive QA quality.
- We add a held-out HotpotQA evaluation that directly measures heterogeneous retrieval requirements, addressing whether adaptation is useful beyond simple factoid QA.
- We introduce a score-aware adaptive context gate and report frozen held-out performance, query-level failure analysis, planner latency, and paired bootstrap 95% confidence intervals against static $k = 10$.

II. Related Work

A. Retrieval-Augmented Generation and Dense Retrieval

RAG was formalized by Lewis et al. as a framework combining parametric generation with retrieved non-parametric memory for knowledge-intensive tasks [1]. Subsequent surveys characterize RAG as a modular retrieval–augmentation–generation pipeline and identify retrieval quality, context selection, and efficiency as core design concerns [2]. Dense Passage Retrieval showed the effectiveness of neural semantic retrieval for open-domain QA [15], while Sentence-BERT enabled efficient dense representations for semantic similarity search [11]. FAISS provides efficient similarity search over dense vectors [10]. These components make vector retrieval practical but do not determine the appropriate retrieval depth for each query.

B. Adaptive and Reflective Retrieval

Recent methods move beyond fixed retrieval policies. Self-RAG learns when to retrieve and critique evidence using reflection tokens [3]. Adaptive-RAG selects retrieval strategies based on question complexity [4]. RAPTOR retrieves over a hierarchy of recursively summarized text [6], while RAFT trains models to reason with retrieved documents and ignore distractors [7]. In contrast, CARP-RAG focuses on a small systems-level control layer: the retriever and generator remain unchanged, while the planner decides how much retrieved evidence should be propagated under a compute constraint.

C. Benchmarks and Evidence Requirements

BEIR demonstrates that retrieval effectiveness varies substantially across heterogeneous tasks [9], and FinanceBench highlights challenges in enterprise open-book QA where long-context processing can be costly [8]. SQuAD is useful for controlled extractive QA analysis [13], but it does not strongly stress multi-document retrieval. HotpotQA contains multi-hop questions with annotated supporting facts and is therefore better suited for testing whether different queries require different retrieval depths [14]. CARP-RAG uses both workloads for complementary purposes rather than treating one benchmark as representative of all RAG settings.

III. CARP-RAG Method

A. Framework Overview

Let q be a query, V a vector index, and $\mathcal{K} = \{3, 5, 10, 20\}$ the candidate depth set. CARP-RAG chooses a depth k based on a lightweight planning signal and a compute constraint. The framework does not require changes to the embedding model, vector index, or downstream QA model.

B. Rule-Based Query Complexity

The SQuAD planner assigns a complexity score using four transparent indicators: (i) query length of at least 12 words, (ii) comparison terms, (iii) causal or explanatory terms, and (iv) numerical-reasoning phrases. One point is added for each matched indicator. A score of 0 is labeled low complexity, scores of 1–2 are medium, and scores above 2 are high. The corresponding initial depth is

$$k_0(q) = \begin{cases} 3, & C(q) = \text{low}, \\ 5, & C(q) = \text{medium}, \\ 10, & C(q) = \text{high}. \end{cases} \tag{1}$$

This rule is intentionally simple and is reported explicitly to make the planner reproducible.

C. Latency-Budget Planning

For the rule-based implementation, an empirical latency table $\hat{T}(k)$ is calibrated from the static SQuAD runs. Rather than optimizing an unobserved quality function, the implemented planner chooses the largest depth that is both no larger than the complexity-selected depth and feasible under budget B:

$$k^*(q, B) = \max\{k \in \mathcal{K} : k \le k_0(q),\ \hat{T}(k) \le B\}. \tag{2}$$

If the initial depth exceeds the budget, the planner steps down through $10 \to 5 \to 3$ until the estimated latency is feasible. This equation matches the implemented decision rule.

D. Score-Aware Context Allocation

The HotpotQA experiment tests a different CARP instantiation because the primary question is whether retrieval demand can be estimated from the retrieval result itself. A top-20 candidate pool is retrieved once using normalized sentence embeddings and inner-product similarity. From the ranked similarity scores, the planner derives a compact feature vector containing rank scores, mean similarity at several depths, boundary scores, score gaps, score dispersion, and relative score decay. Lightweight query-form features are concatenated to this vector.

A logistic regression gate is trained on the 500-query development partition to predict whether $k = 3$ is sufficient. The binary target is

$$y(q) = \begin{cases} 0, & \text{all support is recovered by } k = 3, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

The decision threshold and deep retrieval depth are selected using development-only out-of-fold evaluation. The frozen configuration uses threshold $\tau = 0.30$ with

$$k_{\text{Score}}(q) = \begin{cases} 3, & P(y = 1 \mid q) < 0.30, \\ 10, & P(y = 1 \mid q) \ge 0.30. \end{cases} \tag{4}$$

No held-out query is used to tune the model, threshold, or depth values. Importantly, Score-CARP does not reduce the already-inexpensive top-20 FAISS candidate lookup in this experiment; it reduces the amount of retrieved evidence propagated to downstream processing.

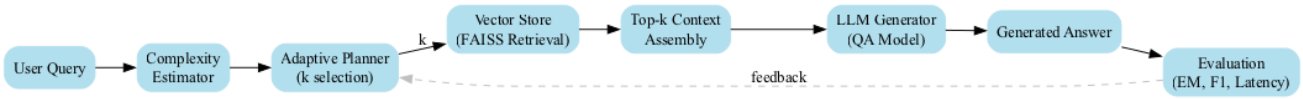


Fig. 1. CARP-RAG architecture. The reported experiments use a rule-based pre-retrieval planner for SQuAD and a score-aware post-candidate-retrieval gate for HotpotQA. Any feedback path is architectural future work and is not used to update the planner online in the reported experiments.

E. Retrieval and Answer Generation

Document chunks are embedded with `sentence-transformers/all-MiniLM-L6-v2` using normalized embeddings. Retrieval uses FAISS `IndexFlatIP`; with normalized embeddings, inner product is equivalent to cosine similarity. In the SQuAD experiment, retrieved chunks are concatenated and passed to `distilbert-base-cased-distilled-squad`, a DistilBERT-based extractive QA model [12]. The model input is limited by the 512-token transformer context, so increasing k can introduce truncation and distractor effects. For HotpotQA, the primary evaluation is retrieval evidence coverage rather than answer generation because the extractive SQuAD model is not a reliable multi-hop answer synthesizer.

F. Planning Procedures

Algorithm 1: Rule-CARP Latency-Budget Planning
Input: query q, budget B, vector index V
1. Compute transparent complexity score $C(q)$.
2. Map $C(q)$ to initial depth $k_0 \in \{3, 5, 10\}$.
3. Set $k^* = \max\{k \in \mathcal{K} : k \leq k_0, \hat{T}(k) \leq B\}$.
4. Embed q and retrieve top-k^* chunks from V.
5. Assemble context, run QA model, and log depth, latency, EM, and F1.

Algorithm 2: Score-CARP Context Allocation
Input: query q, vector index V, frozen gate g, threshold $\tau = 0.30$
1. Embed q and retrieve a top-20 candidate pool.
2. Extract similarity-profile and lightweight query features.
3. Compute $p = g(q)$, the estimated probability that $k = 3$ is insufficient.
4. Propagate $k = 10$ passages if $p \geq \tau$; otherwise propagate $k = 3$.
5. Log selected depth, supporting-document recall, complete-support retrieval, and planner latency.

IV. Experimental Setup

A. SQuAD Workload

The first experiment uses 500 SQuAD v1.1 questions [13]. This workload is used to measure end-to-end extractive QA quality and latency as retrieval depth changes. Static baselines use $k \in \{3, 5, 10, 20\}$. Rule-CARP is evaluated with budgets of 0.034, 0.036, and 0.040 s. Exact match (EM) and normalized token-overlap F1 are computed against reference answers.

B. HotpotQA Development and Held-Out Workload

The second experiment uses 1000 HotpotQA validation examples [14]. Examples 0–499 form the development partition and 500–999 form a held-out test partition. Each partition uses a shared distractor-derived corpus constructed from all paragraphs in that partition; duplicate title–text pairs are removed. This is a controlled shared-corpus evaluation, not the official HotpotQA full-wiki retrieval setting.

The development corpus contains 4,937 unique passages, and the held-out corpus contains 4,930. Supporting-fact annotations are aggregated to document titles. We therefore report *supporting-document recall* and *complete-support retrieval rate*, not sentence-level supporting-fact F1. The score-aware gate and its threshold are selected using development data only and frozen before the held-out run.

C. Baselines and Ablations

For both workloads, static retrieval baselines use $k = 3, 5, 10, 20$. The SQuAD experiment additionally compares the rule-based budget planner. The HotpotQA development study evaluates three adaptive designs: the original rule-based complexity planner, a query-embedding-only learned gate, and the final retrieval-score-aware gate. The held-out result is reported only for the frozen Score-CARP configuration ($k = 3/10$, threshold 0.30); the test split is not used for further tuning.

D. Metrics and Statistical Analysis

For SQuAD, total latency is

$$T_{\text{total}} = T_{\text{retrieval}} + T_{\text{generation}}, \tag{5}$$

and we report average latency, p95, EM, F1, and average k. For HotpotQA, supporting-document recall for query i is

$$R_i = \frac{|G_i \cap D_i(k)|}{|G_i|}, \tag{6}$$

where G_i is the set of gold supporting-document titles and $D_i(k)$ is the set of retrieved titles at depth k. Complete-support retrieval equals 1 when $G_i \subseteq D_i(k)$ and 0 otherwise. We additionally report average propagated passages and the minimum oracle depth in $\{3, 5, 10, 20\}$ required to recover all support. Paired differences between Score-CARP and static $k = 10$ are summarized with 10,000-sample paired bootstrap 95% confidence intervals.

All measurements were collected on the same local Apple M-series workstation. Latency values are intended as controlled relative measurements rather than production-hardware estimates.

V. Results and Discussion

A. SQuAD: Static and Budget-Aware Retrieval

Table I reports the 500-query SQuAD results. Static $k = 3$ provides the highest F1 (0.5720) and the lowest average total latency among the fixed-depth baselines. Increasing k from 3 to 20 does not improve answer quality. This result is consistent with the downstream model's finite context window: additional retrieved chunks can introduce distractors or be truncated rather than contribute useful evidence.

At $B = 0.040$, Rule-CARP reduces average retrieval depth from the static-5 baseline's 5.0 to 4.292 while yielding slightly higher EM and F1 than static $k = 5$. At the strict $B = 0.034$ setting, the latency table makes $k = 3$ the only feasible depth, reproducing the strongest static configuration. These results show that the rule-based planner responds predictably to the configured latency envelope, but they also reveal a limitation: SQuAD itself does not require deeper retrieval often enough to demonstrate strong query-level adaptation.

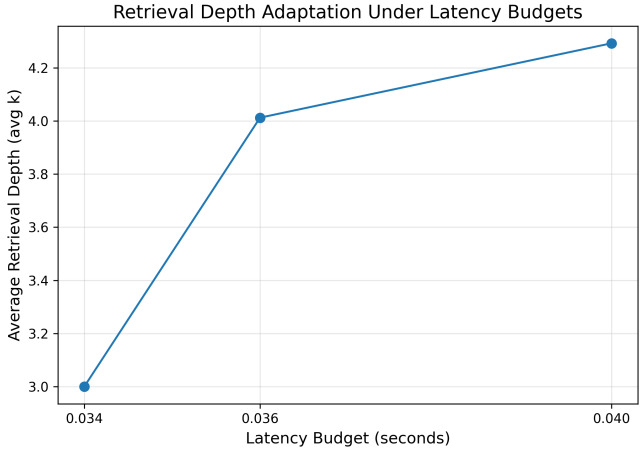


Fig. 2. Average retrieval depth under the SQuAD latency budgets.

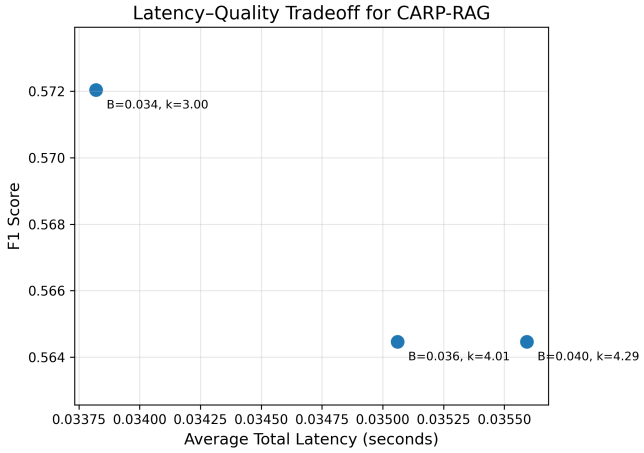


Fig. 3. SQuAD latency–quality tradeoff for the evaluated budget settings.

B. HotpotQA: Evidence-Demand Heterogeneity

The held-out oracle distribution directly demonstrates heterogeneous retrieval requirements. Of 500 queries, 195 (39.0%) recover all supporting documents by $k = 3$, 68 (13.6%) first succeed at $k = 5$, 73 (14.6%) at $k = 10$, and 38 (7.6%) at $k = 20$. The remaining 126 (25.2%) are not completely recovered even at $k = 20$. Thus, a static $k = 3$ policy is sufficient for a substantial subset, but 35.8% require a deeper value within the tested range and another 25.2% exceed that range. This is the retrieval heterogeneity that is largely absent from the SQuAD experiment.

C. Held-Out Score-CARP Tradeoff

Table III compares static baselines with the frozen Score-CARP policy. Static $k = 10$ achieves supporting-document recall 0.822 and complete-support retrieval 0.672. Score-CARP achieves 0.805 and 0.640, respectively, while reducing average propagated passages from 10.0 to 8.866, an 11.34% reduction. Static $k = 20$ obtains the highest evidence coverage, but doubles the propagated context relative to static $k = 10$ and more than doubles it relative to shallow $k = 3$.

The paired bootstrap analysis shows that the quality differences relative to static $k = 10$ are small but statistically distinguishable. Supporting-document recall changes by -0.017 with 95% CI $[-0.026, -0.009]$, and complete-support retrieval changes by -0.032 with 95% CI $[-0.048, -0.018]$. Therefore, the correct interpretation is not that CARP matches static $k = 10$ without loss. Rather, it exposes a measurable compute–quality tradeoff: 11.34% less propagated context in exchange for a 1.7-point recall decrease and a 3.2-point complete-support decrease.

At the query level, both methods retrieve all supporting documents for 320 queries and both fail for 164. Static $k = 10$ succeeds on 16 queries where Score-CARP does not; Score-CARP cannot uniquely succeed because it never propagates more than 10 passages. This makes the tradeoff directly interpretable: context is reduced broadly, while complete evidence is lost on 16 of 500 held-out queries compared with static $k = 10$.

D. Planner Ablation and Generalization

The development experiments show why retrieval-score information is important. A three-class query-embedding gate that attempted to predict $k \in \{3, 10, 20\}$ achieved only 0.4375 out-of-fold accuracy, below the 0.5208 majority-class accuracy, although its balanced accuracy was 0.3989 versus the trivial 0.3333 baseline. A simpler binary query-embedding gate achieved 0.584 accuracy and 0.580 balanced accuracy, but its context-allocation points were not consistently competitive with static baselines. Adding the retrieval-score profile improved binary out-of-fold accuracy to 0.664, balanced accuracy to 0.6575, and macro F1 to 0.6547. The development-selected 3/10 policy at threshold 0.30 then generalized closely to the held-out split: development average $k = 8.936$ and all-support 0.638 versus held-out average $k = 8.866$ and all-support 0.640.

TABLE I
SQUAD RESULTS OVER 500 QUERIES. LATENCIES ARE IN SECONDS.

Method	Avg. k	Ret. Lat.	Gen. Lat.	Total Lat.	P95 Total	EM	F1
Static-3	3.000	0.008238	0.026471	0.034721	0.039411	0.490	0.572038
Static-5	5.000	0.008165	0.027826	0.036006	0.039121	0.480	0.562377
Static-10	10.000	0.008197	0.029191	0.037412	0.042954	0.480	0.562702
Static-20	20.000	0.008244	0.029401	0.037687	0.039812	0.480	0.562702
Rule-CARP, $B = .034$	3.000	0.008198	0.025611	0.033821	0.036232	0.490	0.572038
Rule-CARP, $B = .036$	4.012	0.008171	0.026876	0.035061	0.037660	0.482	0.564461
Rule-CARP, $B = .040$	4.292	0.008196	0.027384	0.035593	0.039473	0.482	0.564461

TABLE II
HELD-OUT HOTPOTQA ORACLE RETRIEVAL-DEPTH DISTRIBUTION ($N = 500$).

Minimum depth	Queries	Share
$k = 3$	195	39.0%
$k = 5$	68	13.6%
$k = 10$	73	14.6%
$k = 20$	38	7.6%
> 20	126	25.2%

TABLE III
HELD-OUT HOTPOTQA RETRIEVAL RESULTS ($N = 500$).

Method	Avg. k	Support Recall	All Support
Static-3	3.000	0.661	0.390
Static-5	5.000	0.742	0.526
Static-10	10.000	0.822	0.672
Static-20	20.000	0.864	0.748
Score-CARP	8.866	0.805	0.640

This ablation supports a narrower claim than simply "complex questions need larger k." Query wording alone is a weak proxy for retrieval demand. The ranked retrieval-score profile provides a more useful runtime signal for deciding whether shallow context is sufficient.

E. Planner Latency

For the held-out Score-CARP run, the classifier decision itself averages 0.000128 s (0.128 ms) with p95 0.000135 s. The top-20 FAISS lookup, measured after precomputing query embeddings, averages 0.000172 s. The ratio between these two very small components is not a meaningful end-to-end overhead percentage because it excludes query embedding and downstream processing. We therefore report the absolute planner decision latency and avoid extrapolating it to production overhead. In the SQuAD end-to-end pipeline, query processing and generation dominate the microsecond-scale FAISS lookup.

F. When Adaptation Succeeds and Fails

Score-CARP recovers all support for every held-out query whose oracle depth is $k = 3$. For oracle-5 and oracle-10 queries, complete-support rates are 0.8676 and 0.9041, respectively. By design, the frozen policy cannot recover queries whose minimum required depth is $k = 20$, and it cannot solve cases that remain incomplete at $k = 20$. These groups account for 32.8% of the held-out set. The result exposes a fundamental policy limitation: conservative context allocation is effective when useful evidence is concentrated within the allowed deep depth, but a binary 3/10 controller cannot serve workloads that frequently require more than 10 passages. A production controller would therefore need either a third deep state, confidence-aware escalation, or a reranking stage.

VI. LIMITATIONS AND THREATS TO VALIDITY

The SQuAD generator is an extractive DistilBERT model with a 512-token input limit. Consequently, the SQuAD experiment should be interpreted as controlled evidence about retrieval-depth and context effects, not as a general evaluation of modern generative LLM RAG. The HotpotQA experiment intentionally evaluates retrieval evidence coverage rather than final answer EM/F1 because the SQuAD-trained extractive model is not an appropriate multi-hop answer synthesizer.

The HotpotQA corpora are split-specific shared distractor-derived corpora rather than the official full-wiki setting. Gold evidence is matched at document-title level, so the reported evidence metric is supporting-document recall rather than supporting-fact recall. Absolute retrieval scores should therefore not be compared directly with official HotpotQA leaderboard results.

The score-aware gate is a lightweight logistic regression model calibrated on 500 development queries. Although the held-out result closely tracks development behavior, broader datasets are needed to assess robustness across domains and corpus scales. The 126 held-out queries unresolved by $k = 20$ also show that retrieval failure and planner failure are distinct: no controller restricted to the same top-20 candidate pool can recover evidence that the retriever does not surface. Finally, latency measurements come from one Apple M-series workstation and are controlled system measurements rather than production service-level estimates.

VII. CONCLUSION

This paper presented CARP-RAG, a compute-aware retrieval planning framework that treats retrieval depth as an adaptive systems parameter. The SQuAD experiment shows that increasing retrieval depth from $k = 3$ to $k = 20$ does not improve extractive QA quality in the evaluated pipeline and that a transparent latency-budget planner can reduce

average depth under explicit timing constraints. The HotpotQA experiment demonstrates why adaptation nevertheless matters: only 39.0% of held-out multi-hop queries are fully supported by $k = 3$, while many require substantially deeper retrieval.

A frozen score-aware CARP policy reduces propagated context by 11.34% relative to static $k = 10$, with supporting-document recall changing from 0.822 to 0.805 and complete-support retrieval from 0.672 to 0.640. These differences are statistically distinguishable, so CARP-RAG should be understood as exposing a controllable quality–compute tradeoff rather than providing free efficiency. The results also show that retrieval-score profiles are more useful for planning than query semantics alone. Future work should evaluate multi-level escalation beyond $k = 10$, reranker-aware planning, larger production-scale corpora, and generative multi-hop models while preserving strict development/test separation.

REFERENCES

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[2] Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023.

[3] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," arXiv:2310.11511, 2023.

[4] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity," in *Proc. NAACL*, 2024. DOI: 10.18653/v1/2024.naacl-long.389.

[5] N. F. Liu et al., "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. DOI: 10.1162/tacl_a_00638.

[6] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval," arXiv:2401.18059, 2024.

[7] T. Zhang et al., "RAFT: Adapting Language Model to Domain Specific RAG," arXiv:2403.10131, 2024.

[8] P. Islam, A. Kannappan, D. Kiela, R. Qian, N. Scherrer, and B. Vidgen, "FinanceBench: A New Benchmark for Financial Question Answering," arXiv:2311.11944, 2023.

[9] N. Thakur, N. Reimers, A. Rueckle, A. Srivastava, and I. Gurevych, "BEIR: A Heterogeneous Benchmark for Zero-Shot Evaluation of Information Retrieval Models," in *NeurIPS Datasets and Benchmarks*, 2021.

[10] J. Johnson, M. Douze, and H. Jégou, "Billion-Scale Similarity Search with GPUs," arXiv:1702.08734, 2017.

[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. EMNLP-IJCNLP*, 2019. DOI: 10.18653/v1/D19-1410.

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," arXiv:1910.01108, 2019.

[13] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proc. EMNLP*, 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[14] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," in *Proc. EMNLP*, 2018, pp. 2369–2380. DOI: 10.18653/v1/D18-1259.

[15] V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," in *Proc. EMNLP*, 2020. DOI: 10.18653/v1/2020.emnlp-main.550.

[16] O. Khattab and M. Zaharia, "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT," in *Proc. SIGIR*, 2020. DOI: 10.1145/3397271.3401075.

[17] O. Khattab et al., "ColBERTv2: Efficient and Effective Retrieval via Lightweight Late Interaction," in *Proc. NAACL*, 2022. DOI: 10.18653/v1/2022.naacl-main.272.