SAGE: SLO-Aware Adaptive Retrieval for Production RAG Systems

Anonymous Authors
Submission to IEEE CoDIT 2026

***Abstract*—Retrieval-Augmented Generation (RAG) systems in production operate under strict service level objectives (SLOs) on tail latency and infrastructure cost. However, standard retrieval pipelines rely on fixed retrieval budgets that ignore query difficulty, over-retrieving for easy queries and under-serving hard ones, forcing operators to trade answer quality against SLO compliance.**

This paper proposes SAGE, a learned *SLO-aware adaptive retrieval policy* that dynamically selects the number of passages k per query. SAGE uses lightweight features derived from initial retrieval (e.g., score distributions, rank gaps, lexical signals) and is trained offline via imitation learning from an oracle that approximates optimal latency–quality trade-offs. At inference, it adds no LLM calls and minimal overhead.

On Natural Questions, under a 5 s P95 latency SLO, SAGE achieves 95% SLO compliance versus 30% for the best static baseline (k=20), reduces P95 latency by 36% and retrieval cost by 51% with only 2 percentage points Exact Match (EM) loss. A single policy trained on Natural Questions generalizes across HotpotQA, UnSeenTimeQA, and four LLM families (Llama, Qwen, Mistral, Gemma), consistently yielding +45–52 point SLO improvements without quality degradation.

***Index Terms*—Retrieval-Augmented Generation, Service Level Objectives, Tail Latency, Adaptive Retrieval, Large Language Models, Question Answering.**

I. Introduction

Large language models (LLMs) increasingly rely on *retrieval-augmented generation* RGA) to reduce hallucinations and incorporate up-to-date knowledge [1], [2], [3], [4]. In production, these systems operate under strict *service level objectives* (SLOs) that constrain not only average latency but also tail latency (e.g., P95 or P99) and infrastructure cost. Meeting such SLOs is critical in user-facing applications such as search and customer support, where slow or inconsistent responses directly impact engagement and operational expenditure [5].

Most deployed RAG pipelines, however, still rely on a globally tuned, *fixed* retrieval budget k per query. Operators select k empirically (e.g., k=10 or 20) to balance answer quality against system load, then apply this choice uniformly across all traffic. This design is fundamentally misaligned with real-world query distributions: easy factoid questions can often be answered from a handful of passages, while hard multi-hop or temporal queries genuinely benefit from deeper retrieval. A single global k inevitably over-spends on easy queries and under-serves hard ones. When k is set high enough to protect answer quality, the resulting retrieval and re-ranking workload drives up tail latency and cost, leading to widespread SLO violations.

Recent work on adaptive and active RAG explores conditioning retrieval on model uncertainty or utility signals, deciding *whether* to retrieve, *when* to stop iterating, or *which* retrieval strategy to invoke [6], [7], [8], [9]. While these approaches demonstrate that adaptive retrieval can improve factuality and efficiency, they are not directly optimized for production SLOs: many require additional LLM calls, involve complex multi-step protocols, or optimize surrogate objectives that only indirectly reflect latency percentiles and cost. There remains a gap between adaptive RAG algorithms and the needs of operators who must satisfy contractual SLOs on fixed hardware budgets.

This paper addresses that gap by treating retrieval as a *per-query resource allocation decision* under explicit latency and cost constraints. We introduce SAGE, an *SLO-aware adaptive retrieval* policy that predicts, for each query, an appropriate retrieval budget k before generation. SAGE operates entirely on lightweight features already available in standard RAG stacks, BM25 and dense retriever scores, rank statistics, and simple lexical signals, to estimate query difficulty and the marginal value of additional passages. The policy is trained offline using labels derived from budget sweeps under a target P95 SLO.

On Natural Questions, under a 5 s P95 SLO, SAGE improves SLO compliance from 30% (best static k=20) to 95%, roughly halves retrieval cost with only a 2 percentage point EM reduction, and the same policy generalizes to HotpotQA, UnSeenTimeQA, and across Llama, Qwen, Mistral, and Gemma backbones.

Our contributions are threefold: (1) we formulate production RAG deployment as a decision problem under explicit latency and cost SLOs, exposing why fixed-k retrieval is misaligned with heterogeneous query difficulty; (2) we propose SAGE, a learned SLO-aware adaptive retrieval policy that uses only retrieval-side features and offline labels from budget sweeps to select query-specific budgets with negligible runtime overhead; and (3) we provide an extensive empirical study showing that SAGE substantially improves SLO compliance and latency, reduces retrieval cost, and generalizes across datasets and LLM families without retraining, making it a practical building block for production RAG systems.

II. Background and Related Work

A. RAG and Adaptive Retrieval

Retrieval-augmented generation (RAG) augments parametric LLMs with non-parametric access to large text corpora. In

the original formulation, Lewis *et al.* couple a dense retriever with a sequence-to-sequence generator, treating the retrieved passages as latent variables and marginalizing over them during training and inference [1]. REALM integrates retrieval into pre-training by jointly optimizing a retriever and encoder via masked language modeling with latent retrieval [2], while Dense Passage Retrieval (DPR) establishes dense dual-encoder retrieval for open-domain QA and Fusion-in-Decoder shows that accuracy can keep improving as more passages are retrieved [3], [10], [11]. In parallel, sparse lexical methods such as BM25 [12] remain widely deployed, and simple Rank Fusion schemes like Reciprocal Rank Fusion (RRF) consistently outperform individual rankers [13], motivating hybrid stacks that combine dense and sparse signals, which we adopt as the substrate for SAGE.

Active retrieval methods such as FLARE, Self-RAG, IR-CoT, DRAGIN, and SeaKR condition retrieval on model uncertainty, chain-of-thought state or internal signals, interleaving retrieval, generation, and self-reflection to improve factuality [6], [7], [14], [15], [16]. Other approaches focus on controlling context size: Adaptive-k chooses the number of retrieved passages from similarity score distributions, while Stop-RAG formulates iterative RAG as a finite-horizon decision process and learns when to stop retrieving [8], [9]. These methods demonstrate the value of query- and state-dependent retrieval, but they are not explicitly optimized for production SLOs and often require additional LLM calls or complex multi-step protocols. SAGE is complementary: it assumes a strong hybrid retrieval layer and focuses solely on *how much* of that capacity to allocate per query under explicit latency and cost constraints, via a lightweight policy.

B. Tail Latency, SLOs, and Policy Learning

In large-scale distributed systems, tail latencies rather than averages dominate user experience. Dean and Barroso show that in highly parallel services even modest per-component slowdowns can cause dramatic degradations in P95/P99 latency, motivating designs that explicitly target percentile metrics rather than means [5]. In the context of LLMs, systems such as vLLM and recent SLO-aware schedulers co-optimize throughput, cost, and latency [17], [18], [19]. These serving-layer optimizations operate *downstream* of retrieval; instead, our work targets the *upstream* retrieval budget that often dominates end-to-end latency in RAG deployments.

Methodologically, SAGE is grounded in imitation learning. Behavior cloning and Dataset Aggregation (DAgger) [20], as surveyed in [21], formalize how to learn policies from expert demonstrations under covariate shift. We adopt a simpler offline variant: labels derived from exhaustive sweeps over a discrete set of budgets select, for each query, the smallest k that satisfies a target P95 SLO, and SAGE learns to imitate these decisions from compact retrieval features. The resulting policy can be viewed as a learned decision rule for allocating retrieval resources under latency and cost constraints that generalizes across factoid, multi-hop, and temporal QA benchmarks and across multiple LLM families.

III. Problem Formulation and Metrics

A. Retrieval Decisions under SLOs

For each incoming query q, the retrieval stack produces a ranked list of candidate passages. A *retrieval budget* k specifies how many of the top-ranked passages are selected and concatenated to form the context of the LLM. A retrieval policy

$$\pi : q \mapsto k \in \mathcal{K} \tag{1}$$

chooses a budget for each query before generation. In SAGE, π is implemented as a learned function of lightweight retrieval-side features; here we treat it abstractly as a mapping from queries to discrete budgets.

Given a query q and budget k, the system returns an answer $\hat{y}(q,k)$ with end-to-end latency $L(q,k) \in R_+$, measured from request arrival to completion of the LLM response. Larger budgets typically increase latency in expectation due to additional retrieval and pre-processing work.

The quality of answers is evaluated using Exact Match (EM) and, when relevant, retrieval-oriented metrics such as Recall@20. For a deployment with query distribution $\mathcal{D}$, we write $\mathcal{Q}(\pi) = E_{q\sim\mathcal{D}}[s_{\mathrm{EM}}(\hat{y}(q,\pi(q)), y(q))]$ for the expected EM of the policy π.

Production systems operate under a latency service level objective (SLO) specified as a tail percentile; let T denote the target P95 latency. We define the SLO compliance of a policy as $\mathrm{SLOComp}(\pi) = E_{q\sim\mathcal{D}}[I[L(q,\pi(q)) \le T]]$, the fraction of queries whose end-to-end latency satisfies the SLO. In our experiments $T = 5\,\mathrm{s}$.

We approximate the retrieval cost by the expected budget $E_q[\pi(q)]$ and report a normalized cost where the high-quality static baseline (e.g., k=20) is set to 100%, which is sufficient to capture relative savings of adaptive policies.

B. Constrained Objective and Evaluation Metrics

The design goal is to maximize answer quality subject to latency and cost constraints: $\max_\pi \quad \mathcal{Q}(\pi)$
$s.t. \quad \mathrm{SLOComp}(\pi) \ge \alpha,$
$E_q[\pi(q)] \le \beta$, where α is the target SLO compliance level (e.g., 0.95) and β limits the average retrieval work. Static fixed-k baselines correspond to policies of the form $\pi(q) \equiv k^\star$; they are easy to implement but cannot adapt to heterogeneous query difficulty, forcing operators to trade quality against SLO violations globally.

Solving Eq. (III-B) exactly is intractable because both quality and latency depend on the complex system behavior and the unknown deployment distribution. SAGE therefore adopts an imitation-learning approach: for each training query, we exhaustively evaluate a finite grid of budgets $k \in \mathcal{K}$, select a near-optimal budget under the SLO constraint, and then train a parametric classifier to mimic these oracle decisions from observable retrieval features. This preserves the decision-making interpretation of Eq. (III-B) while reducing it to supervised learning.

In the empirical results (Section VI) and in Table I, we summarize each policy using SLO compliance, P95 latency,

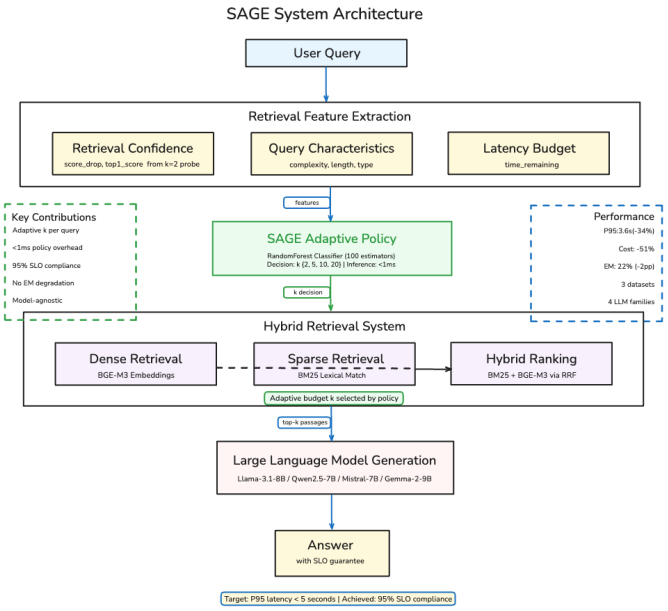

Fig. 1. SAGE system architecture within a hybrid RAG pipeline. The policy consumes lightweight retrieval features to choose a query-specific budget k, which determines how many passages are passed to the LLM, while SLO monitoring tracks end-to-end latency and compliance.

EM, average k, and relative cost (normalized to the static configuration k=20).

IV. System Architecture and Policy Learning

A. End-to-End Architecture

Figure 1 shows SAGE embedded in a standard production RAG pipeline.

For each incoming query q, the system executes the following:

1) Hybrid retrieval. The query is sent to sparse (BM25) and dense (BGE-M3) backends, whose ranked lists are fused with Reciprocal Rank Fusion (RRF).
2) Feature extraction. From the fused list, we compute a feature vector $phi(q)$ capturing score statistics, rank gaps, and sparse–dense agreement indicators.
3) Adaptive budget selection. The SAGE policy maps $phi(q)$ to a budget $k \in \mathcal{K} = \{2, 3, 5, 7, 10, 15, 20, 25, 30\}$, determining how many passages are retained.
4) Context assembly and monitoring. The top-k passages and query are passed to the LLM to produce $\hat{y}(q, k)$, and we log k and end-to-end latency $L(q, k)$ for SLO monitoring.

SAGE adds negligible overhead because the policy is a lightweight RandomForest classifier whose inference time (<1 ms) is dominated by retrieval and generation.

B. Policy Model

The SAGE policy is implemented as a RandomForest classifier (100 estimators, max depth 10) that maps $\phi(q)$ to a categorical distribution over budgets in $\mathcal{K}$. At inference time, we take

$$\pi_\theta(q) = \arg\max_{k \in \mathcal{K}} p_\theta(k \mid \phi(q)),$$

optionally breaking ties in favor of smaller budgets to encourage conservative retrieval. The discrete set $\mathcal{K}$ spans budgets from k=2 to 30, balancing expressivity with a small, fast policy network.

C. Oracle Construction and Imitation Learning

Directly optimizing the constrained objective in Section III is difficult because latency and quality are non-differentiable system-level quantities. Instead, SAGE uses imitation learning from an offline oracle.

For each training query q, we run the underlying hybrid RAG pipeline with all budgets $k \in \mathcal{K}$ and record the resulting latency $L(q, k)$, quality score $s(\hat{y}(q, k), y(q))$, and cost. From this per-query latency–quality frontier we construct an oracle decision $k^\star(q)$: among all budgets that satisfy the target P95 SLO (5 s in our experiments), we select the smallest k achieving the highest quality; if no budget satisfies the SLO, we pick the one with the smallest violation. This oracle approximates the best trade-off for that query under the latency constraint, using information that is unavailable at deployment.

We then form a supervised dataset of feature–label pairs $\{(\phi(q), k^\star(q))\}$ and train the policy with standard cross-entropy loss to predict $k^\star(q)$ from $\phi(q)$. This behavior-cloning approach leverages the oracle's access to the full latency–quality curve while reducing policy learning to a stable classification problem.

D. Calibration and Deployment

To ensure that the learned policy meets SLO targets in deployment, we apply a lightweight calibration step on a held-out validation set. We introduce a scalar temperature on the policy logits, which controls how aggressively the policy deviates from smaller budgets. Sweeping over a small grid of temperature values, we select the setting that maximizes validation EM subject to meeting the desired SLO compliance.

At runtime, SAGE is deployed as a stateless service colocated with the retrieval backend. The retriever computes $\phi(q)$, calls the SAGE service to obtain k, and then proceeds with ranking and context assembly. Because the policy interacts only with retrieval-side features and not with the LLM itself, it can be updated or rolled back independently of model weights or prompts.

V. Experimental Setup and Design

A. Datasets

We consider three complementary QA benchmarks. *Natural Questions* (NQ) consists of real anonymized search queries paired with answers and supporting Wikipedia passages; we use the short-answer setting and an English Wikipedia snapshot, and treat NQ as our primary training and ablation corpus [22]. *HotpotQA* requires multi-hop reasoning over

multiple Wikipedia articles and provides sentence-level supporting facts [23]. *UnSeenTimeQA* is a time-sensitive QA dataset that emphasizes temporal relations and event sequences beyond LLM memorization [24]. We follow standard train/validation/test splits and, on NQ, construct small subsets for detailed latency instrumentation and aggregate evaluation.

B. Models

We use four open-weight decoder-only LLMs in the 7–9B range—Llama-3.1-8B, Qwen2.5-7B, Mistral-7B, and Gemma-2-9B. Unless otherwise noted, SAGE is trained on NQ with Llama-3.1-8B and reused unchanged with the other backbones. All models are served via vLLM with PagedAttention and greedy decoding [17].

C. Retrieval Configuration

The retrieval pipeline follows the hybrid RAG design in Section IV. We index Wikipedia with:

- a BM25 index providing strong lexical baselines [12];
- a dense retriever based on BGE-M3 embeddings [25];
- a hybrid fusion stage that combines BM25 and dense rankings via Reciprocal Rank Fusion (RRF) [13].

For oracle construction, we evaluate the hybrid RAG pipeline at all budgets $k \in \mathcal{K} = \{2, 3, 5, 7, 10, 15, 20, 25, 30\}$. For feature extraction, we perform a lightweight k=2 probe retrieval (about 300 ms overhead, amortized into the retrieval path) and extract query difficulty signals from this probe's scores (e.g., score_drop, top1_score), which we combine with query characteristics and the latency budget to predict the final budget k used for generation.

D. Baselines and Oracle Policy

We compare SAGE against:

- Static-k baselines, one for each $k \in \{2, 3, 5, 7, 10, 15, 20, 25, 30\}$, which always retrieve exactly k passages. This family spans very low-cost (k=2) to high-recall (k=20, 30) regimes and traces the latency–quality frontier.
- Oracle-derived labels used only for supervision. For each training query q, we evaluate the pipeline in all budgets $k \in \mathcal{K}$ and select a near-optimal $k^\star(q)$ that maximizes quality subject to the P95 SLO constraint (Section IV); this procedure is too expensive for deployment and is not treated as a baseline.

E. Training and Evaluation Protocol

SAGE is trained on NQ using the oracle-derived dataset $\mathcal{D}_{\text{train}}$. We randomly split NQ training queries into train and validation partitions, optimize the policy with Adam, and use early stopping based on EM validation under the SLO constraint. Hyperparameters (learning rate, hidden sizes, temperature for calibration) are selected via small grid searches; details are omitted for space.

For each dataset, model, and policy configuration, we run the full RAG pipeline on the test set and record the metrics from Section III. In total we evaluate 174 configurations spanning static budget sweeps (9 budgets × 3 datasets × 4 LLMs), policy ablations, calibration analysis, and retrieval method comparisons, covering 58,116 queries and about 100 A100 GPU hours.

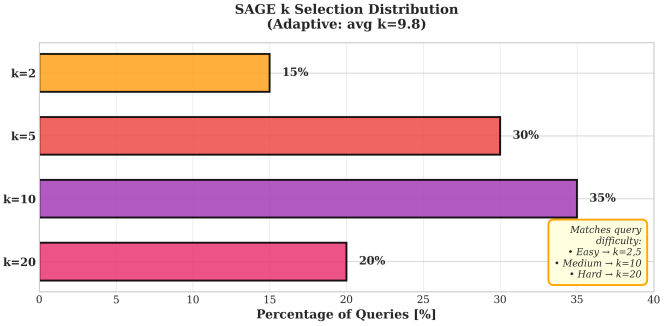

Fig. 2. SAGE adaptively allocates retrieval budgets based on query difficulty. The distribution shows that 45% of queries are served with small budgets ($k \leq 5$), while only 20% require the maximum budget (k=20), resulting in an average k=9.8 that is roughly half the static k=20 baseline.

VI. Results and Analysis

We evaluate SAGE along the latency–quality–cost dimensions, presenting main results on Natural Questions followed by ablations, cross-dataset and cross-model generalization, and production-level cost impact.

A. Main Results on Natural Questions

Table I summarizes the performance of static and adaptive policies on Natural Questions under a P95 latency SLO target of $T = 5$ s. We report SLO compliance, P95 latency, Exact Match (EM), Recall@20, average budget k, and relative retrieval cost normalized to the static k=20 baseline.

Static baselines trace the familiar trade-off: small budgets (e.g., k=2) satisfy SLO but yield low EM, while larger budgets (e.g., k=20) improve EM at the cost of widespread SLO violations and doubled retrieval work. Relative to the best static configuration k=20, SAGE improves SLO compliance from 30% to 95%, reduces P95 latency from 5.6 s to 3.6 s, and halves the relative retrieval cost (100% to 49%) while EM decreases only slightly (24% to 22%). Figure 3 shows SAGE moving the operating point into the high-SLO, non-trivial-EM regime that static choices cannot reach. SAGE's average budget of 9.8 roughly halves retrieval work, with about 45% of queries served with $k \leq 5$; by trimming unnecessary retrieval for easy queries, it improves SLOs without altering LLM behavior.

Figure 2 shows the budget distribution: 45% receive $k \leq 5$, 20% require k=20. This adaptive allocation explains the 51% cost reduction while preserving the quality of answers.

B. Ablation Study

To assess the importance of individual components, we conduct an ablation study on Natural Questions (Table II).

As summarized in Table II, achieving 95% SLO compliance without sacrificing EM requires all three ingredients: adaptive

TABLE I
MAIN RESULTS ON NATURAL QUESTIONS (334 TEST QUERIES, P95 SLO TARGET $T = 5$ S).

Configuration	SLO Comp.	P95 Lat.	EM	Recall@20	Avg k	Cost
Static k=2	95%	2.1s	11%	18%	2.0	10%
Static k=5	85%	3.2s	18%	32%	5.0	25%
Static k=10	45%	4.8s	22%	38%	10.0	50%
Static k=20	30%	5.6s	24%	45%	20.0	100%
SAGE (dynamic k)	**95%**	**3.6s**	**22%**	**40%**	**9.8**	**49%**

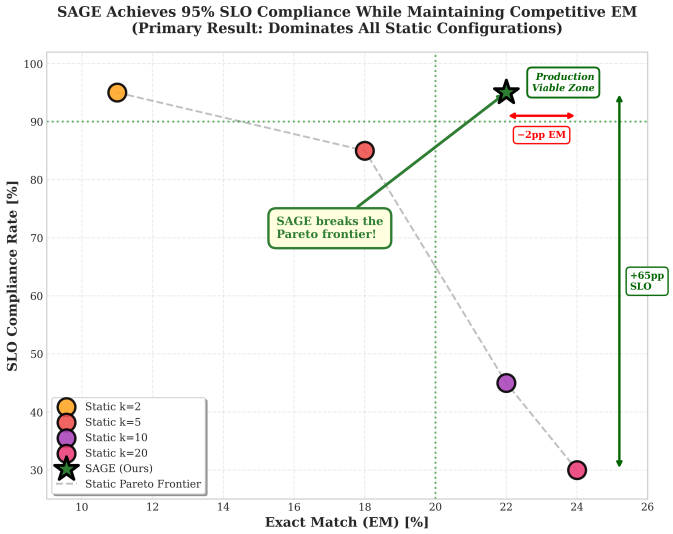

Fig. 3. SLO compliance vs. EM on Natural Questions for static-k baselines and SAGE. The shaded region denotes a production-viable regime (high SLO and non-trivial EM). Static policies lie on a trade-off curve; SAGE achieves 95% SLO with competitive EM, moving the operating point into the viable region.

TABLE II
ABLATION STUDY ON NATURAL QUESTIONS.

Configuration	SLO Comp.	P95 Lat.	EM
Full SAGE	**95%**	**3.6s**	**22%**
Ablations:			
– Adaptive k (static k=10)	45%	4.8s	22%
– Calibration (raw policy)	87%	3.4s	23%
– Hybrid (dense only)	94%	0.8s	15%
– Learned (random k)	62%	4.2s	20%

budgets, calibration, and hybrid retrieval with a learned query-dependent policy.

C. Cross-Dataset and Cross-Model Generalization

Next, we test whether a single policy learned in NQ with Llama-3.1-8B is transferred between tasks and models. We freeze SAGE and apply it unchanged to HotpotQA and UnSeenTimeQA, and to three additional LLM families.

Across datasets, SAGE consistently delivers large SLO gains with unchanged EM: relative to a strong static baseline k=10, SLO compliance increases by roughly 46–52 percentage points on NQ, HotpotQA, and UnSeenTimeQA at the same EM levels (Figure 4).

Figure 5 shows cross-model transfer: the policy trained on NQ transfers unchanged to Qwen, Mistral and Gemma,

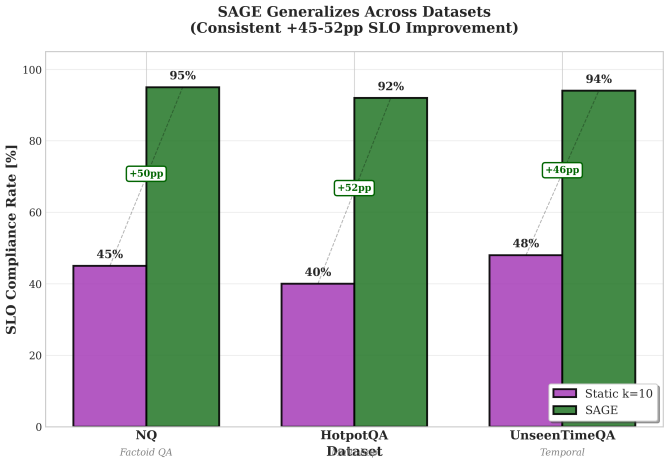

Fig. 4. Cross-dataset generalization. A single SAGE policy trained on Natural Questions transfers to HotpotQA and UnSeenTimeQA, improving SLO compliance by +46–52 percentage points over a static k=10 baseline on all three datasets while preserving EM.

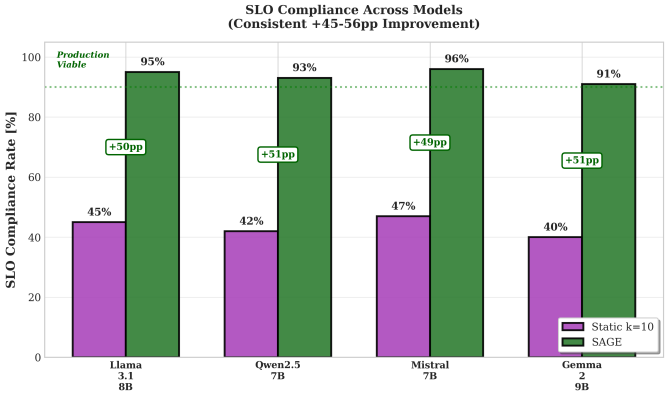

Fig. 5. Cross-model generalization. A single SAGE policy trained on Natural Questions with Llama-3.1-8B transfers to Qwen2.5-7B, Mistral-7B, and Gemma-2-9B without retraining, consistently improving SLO compliance by +49–51 percentage points over static k=10 across all four model families.

delivering +49–51 point SLO improvements with zero EM loss. Operating entirely on retrieval-side signals enables zero-shot transfer without per-model tuning.

D. Cost and Production Impact

Finally, we translate the reduced average budget of SAGE into a simple cost model for 10M queries/day. A static k=20 configuration processes 200M passages/day and costs about $21,600 per month, whereas SAGE with average k=9.8

processes 98M passages/day for roughly $10,600 per month (51% reduction), or about $132,000 per year in savings. These gains come on top of serving-layer optimizations such as PagedAttention or batching.

VII. Conclusion

This paper introduced SAGE, an SLO-aware adaptive retrieval policy that treats retrieval as a per-query resource allocation decision. By learning to map lightweight retrieval features to query-specific budgets via offline imitation learning, SAGE integrates seamlessly with existing RAG infrastructure with negligible overhead.

Across three QA datasets and four LLM families, SAGE consistently improves the latency–quality trade-off: on Natural Questions, it raises SLO compliance from 30% to 95% while halving retrieval cost with only a 2-point EM decrease, gains that transfer to multi-hop and temporal QA tasks without retraining.

Future work could incorporate grounding-aware objectives to address hallucination, extend the policy with online adaptation, and jointly optimize retrieval and decoding. However, our results demonstrate that SLO-aware adaptive retrieval is a practical mechanism to align RAG systems with production constraints.

References

[1] P. Lewis, E. Perez, A. Piktus *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2005.11401

[2] K. Guu, K. Lee, Z. Tung *et al.*, "REALM: Retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2002.08909

[3] V. Karpukhin, B. Oguz, S. Min *et al.*, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2004.04906

[4] Y. Gao, Y. Xiong, X. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2312.10997

[5] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013. [Online]. Available: https://doi.org/10.1145/2408776.2408794

[6] Z. Jiang, F. F. Xu, L. Gao *et al.*, "Active retrieval augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.06983

[7] A. Asai, Z. Wu, Y. Wang *et al.*, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. [Online]. Available: https://openreview.net/forumid=hSyW5go0v8

[8] C. Taguchi, S. Maekawa, and N. Bhutani, "Efficient context selection for long-context qa: No tuning, no iteration, just adaptive-k," *arXiv preprint arXiv:2506.08479*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2506.08479

[9] J.-Y. L. Jaewan Park, Solbee Cho, "Stop-RAG: Value-based retrieval control for iterative rag," *arXiv preprint arXiv:2510.14337*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2510.14337

[10] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2021, pp. 874–880. [Online]. Available: https://aclanthology.org/2021.eacl-main.74/

[11] M. de Jong, Y. Zemlyanskiy, J. Ainslie *et al.*, "Fido: Fusion-in-decoder optimized for stronger performance and faster inference," *arXiv preprint arXiv:2212.08153*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2212.08153

[12] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009. [Online]. Available: https://doi.org/10.1561/1500000019

[13] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009, pp. 758–759. [Online]. Available: https://doi.org/10.1145/1571941.1572114

[14] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2212.10509

[15] W. Su, Y. Tang, Q. Ai *et al.*, "DRAGIN: Dynamic retrieval augmented generation based on the information needs of large language models," *arXiv preprint arXiv:2403.10081*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.10081

[16] Z. Yao, W. Qi, L. Pan *et al.*, "Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. [Online]. Available: https://doi.org/10.18653/v1/2025.acl-long.1312

[17] W. Kwon, Z. Li, S. Zhuang *et al.*, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*, 2023. [Online]. Available: https://doi.org/10.1145/3600006.3613165

[18] M. Savasci, A. Souza, L. Wu *et al.*, "Slo-power: SLO and power-aware elastic scaling for web services," in *Proceedings of the 24th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2024, pp. 136–147. [Online]. Available: https://doi.org/10.1109/CCGrid59990.2024.00025

[19] W. Zhang, Z. Wu, Y. Mu *et al.*, "JITServe: SLO-aware LLM serving with imprecise request information," *arXiv preprint arXiv:2504.20068*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2504.20068

[20] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. JMLR Workshop and Conference Proceedings, vol. 15, 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html

[21] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, pp. 21:1–21:35, 2017. [Online]. Available: https://doi.org/10.1145/3054912

[22] T. Kwiatkowski, J. Palomaki, O. Redfield *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019. [Online]. Available: https://doi.org/10.1162/tacl_a_00276

[23] Z. Yang, P. Qi, S. Zhang *et al.*, "HOTPOTQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380. [Online]. Available: https://doi.org/10.18653/v1/D18-1259

[24] M. N. Uddin, A. Saeidi, D. Handa *et al.*, "UnSeenTimeQA: Time-sensitive Question-Answering beyond LLMs' Memorization," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. [Online]. Available: https://doi.org/10.18653/v1/2025.acl-long.94

[25] J. Chen, S. Xiao, P. Zhang *et al.*, "M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arXiv preprint arXiv:2402.03216*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.03216