

Adaptive Reasoning for Efficient Generative Retrieval

Yingchen Zhang^{1,2,3} Ruqing Zhang^{1,3*} Jiafeng Guo^{1,3} Wenjun Peng⁴
Sen Li⁴ Fuyu Lv⁴ Xueqi Cheng^{1,3}

¹ State Key Laboratory of AI Safety

² School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences

³ Institute of Computing Technology, Chinese Academy of Sciences

⁴ Taobao & Tmall Group of Alibaba

{zhangyingchen23s, zhangruqing, guojiafeng}@ict.ac.cn,

pengwj@mail.ustc.edu.cn, {lisen.lisen, fuyu.lyf}@alibaba-inc.com, cxq@ict.ac.cn

Abstract

Generative retrieval (GR) retrieves documents by directly generating document identifiers (DocIDs) from queries using a sequence-to-sequence model. Recent reasoning-enhanced GR methods improve retrieval effectiveness by generating intermediate reasoning before DocID decoding. However, they typically apply reasoning uniformly to all queries, even though many queries can already be handled effectively through direct DocID decoding. This introduces unnecessary token generation and retrieval latency when reasoning provides little additional benefit. In this work, we propose Generative Retrieval with Adaptive Reasoning (GEAR), a retrieval framework that dynamically allocates reasoning computation based on query-specific needs. Instead of treating reasoning as a fixed pre-retrieval procedure, GEAR learns when to reason, what retrieval-oriented context to generate, and when to transition from free-form generation to constrained DocID decoding. This adaptive behavior is implemented through a learnable retrieval transition policy optimized jointly for retrieval quality and reasoning cost. Experiments on NQ, MS MARCO, and TREC DL 2022 show that GEAR maintains retrieval effectiveness comparable to strong reasoning-enhanced GR methods while substantially reducing reasoning overhead, triggering non-trivial reasoning for only 37.1% of queries and reducing both generated tokens and retrieval latency.

1 Introduction

Generative retrieval (GR) formulates retrieval as sequence generation, where a sequence-to-sequence large language model (LLM) directly generates the identifier of a relevant document given a query (Tay et al., 2022; Bevilacqua et al., 2022; Wang et al., 2022; Zeng et al., 2024b). By replacing explicit index lookup with autoregressive decoding of a discrete document identifier (DocID), GR provides

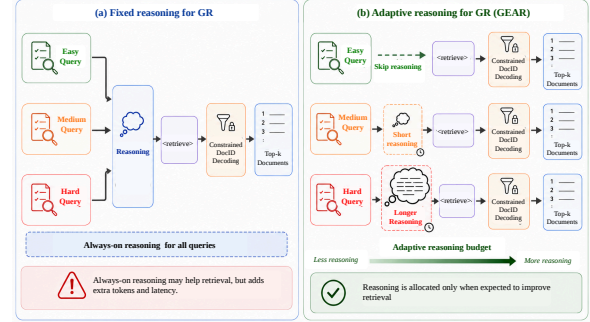


Figure 1: Comparison between fixed reasoning-enhanced GR and our GEAR. Existing methods apply reasoning uniformly regardless of query difficulty. In contrast, GEAR treats reasoning as a query-dependent retrieval resource and dynamically allocates different reasoning budgets based on the expected retrieval benefit of each query.

a unified and end-to-end alternative to traditional sparse and dense retrieval systems (Robertson et al., 2009; Nogueira et al., 2019; Karpukhin et al., 2020; Xiong et al., 2020).

Generative retrieval with intermediate reasoning. Recent work has shown that introducing intermediate reasoning before document identifier (DocID) decoding can improve retrieval effectiveness, particularly for complex or ambiguous queries (Zhang et al., 2025b; Liu et al., 2025; Dong et al., 2026). Instead of decoding DocIDs immediately after receiving a query (Chen et al., 2022b; Tay et al., 2022; Chen et al., 2022a; Zeng et al., 2024a), these methods first generate intermediate natural-language reasoning that clarifies query intent, resolves ambiguity, or surfaces latent semantic constraints before retrieval.

The inefficiency of fixed reasoning. However, as shown in Figure 1 (a), existing reasoning-enhanced GR methods typically apply reasoning uniformly to all queries, regardless of whether additional reasoning is actually necessary. This design overlooks an important property of retrieval workloads: *the utility of reasoning is highly uneven across queries.*

* Corresponding author

Many queries can already be handled effectively through direct DocID decoding, while the benefit of additional semantic analysis varies substantially across queries (Amati et al., 2004; Datta et al., 2024). As a result, fixed reasoning-before-retrieval pipelines often spend considerable generation budget on queries where reasoning provides little or no retrieval gain.

This inefficiency is particularly problematic for retrieval systems. Unlike general text generation tasks, retrieval is highly latency-sensitive and often operates under large-scale query throughput constraints. Even relatively short reasoning traces can substantially increase decoding latency and token cost when invoked for every query. These observations suggest that *the central challenge in reasoning-enhanced GR is not simply how to introduce reasoning, but how to allocate reasoning computation adaptively*.

Adaptive reasoning as retrieval-time computation allocation. In this work, we treat reasoning as a query-dependent retrieval resource rather than a mandatory pre-retrieval procedure. Reasoning should be skipped when direct DocID decoding is sufficient, invoked when additional semantic analysis is beneficial, and terminated once its marginal retrieval benefit no longer justifies its generation cost. This perspective turns reasoning-enhanced retrieval into an adaptive computation problem, where the model must jointly optimize retrieval quality and reasoning efficiency.

However, adaptive reasoning in GR is non-trivial. Reasoning and retrieval correspond to two fundamentally different generation regimes: free-form natural-language generation and constrained DocID decoding over corpus-valid identifiers. An effective GR system must therefore learn not only whether reasoning is needed, but also when and how to transition from unconstrained semantic generation to symbolic retrieval decoding.

The proposed GEAR framework. To address this challenge, we propose *Generative Retrieval with Adaptive Reasoning (GEAR)*, a self-budgeted reasoning framework for efficient generative retrieval. As shown in Figure 1 (b), GEAR learns a retrieval-oriented reasoning policy that dynamically decides whether reasoning is needed, what intermediate retrieval-oriented context to generate, and when to transition from reasoning to retrieval.

For easy queries, GEAR can directly enter DocID decoding without generating reasoning tokens.

For more difficult queries, it generates lightweight retrieval-oriented reasoning that clarifies query intent or highlights useful semantic constraints before retrieval. We implement this behavior through a learnable transition action represented by a special token <retrieve>, whose emission switches the model from free-form generation to constrained DocID decoding. To support this adaptive behavior, GEAR combines mixed DocID-language continued pretraining, difficulty-aware supervised fine-tuning, and self-budgeted reasoning reinforcement learning. The training objective explicitly balances retrieval effectiveness against reasoning cost, encouraging the model to reason only when additional computation improves retrieval quality.

Experiments. Our experimental results on NQ (Kwiatkowski et al., 2019), MS MARCO (Bajaj et al., 2016), and TREC DL 2022 (Craswell et al., 2025) demonstrate that GEAR achieves a strong effectiveness-efficiency trade-off. Many queries can already be handled effectively through direct DocID decoding, while the benefit of additional semantic analysis varies substantially across queries (Amati et al., 2004; Datta et al., 2024). In particular, GEAR triggers non-trivial reasoning for only 37.1% of queries on average, reducing both generated tokens and retrieval latency compared with fixed reasoning-enhanced GR methods. These results show that reasoning need not be applied uniformly to all queries and can instead be allocated selectively while preserving competitive retrieval effectiveness.

2 Related Work

Generative retrieval (GR). Existing GR studies have mainly focused on designing effective document identifiers and improving query-to-DocID generation. Early methods represent the corpus as a differentiable index and train sequence-to-sequence models to map queries to document identifiers (De Cao et al., 2020; Tay et al., 2022). Subsequent work improves identifier construction with textual identifiers (Bevilacqua et al., 2022; Li et al., 2023) or learned discrete identifiers (Wang et al., 2022), and further enhances training and decoding with ranking-aware objectives (Tang et al., 2024; Li et al., 2024b) or planning-ahead generation (Zeng et al., 2024b). Related studies such as LC-Rec (Zheng et al., 2024) and LMIndexer (Jin et al., 2024) further align discrete semantic identifiers with natural-language representations, show-

ing that grounding identifier tokens in language semantics can improve their interaction with language models. Despite these advances, most GR methods still follow a direct decoding paradigm: once a query is given, the model immediately enters constrained DocID generation.

Recent studies have introduced explicit reasoning into GR models. ReasonGR (Dong et al., 2026) targets complex numerical queries in financial documents and introduces multi-step reasoning before retrieval, while R4R (Zhang et al., 2025b) adopts iterative think–retrieve–rethink reasoning to improve retrieval quality. In recommendation settings, OneRec-Think (Liu et al., 2025) incorporates chain-of-thought reasoning to model user preferences and recommendation rationales before item identifier generation. These methods show that intermediate reasoning can improve retrieval effectiveness in complex scenarios. However, they generally treat reasoning as a fixed pre-retrieval step, introducing substantial token and latency overhead that limits practical deployment efficiency.

Adaptive and efficient reasoning. Adaptive reasoning has been explored as a way to reduce unnecessary reasoning computation in LLMs (Sui et al., 2025). Existing methods typically fall into several categories. Some methods decide whether to trigger reasoning based on input difficulty or model uncertainty, so that chain-of-thought reasoning is used only for examples that are likely to benefit from it (Lou et al., 2025; Zhang et al., 2025a). Other methods allocate different reasoning budgets to different inputs, allowing hard questions to receive more thinking tokens while keeping simple questions efficient (Shen et al., 2025; Han et al., 2025). Another line of work studies when to stop reasoning once it has started, for example by terminating generation when intermediate answers become stable or when further reasoning is estimated to bring limited benefit (Li et al., 2024a; Nagle et al., 2026).

These methods show that explicit reasoning should not be uniformly applied to all inputs. However, they are mainly designed for answer generation, where the objective is usually final answer correctness. GR poses a different budget-control problem: a reasoning prefix is useful only if it improves the ranked DocID list produced by constrained decoding. Therefore, GEAR optimizes reasoning behavior with retrieval-grounded feedback, rewarding reasoning prefixes that improve downstream retrieval quality while penalizing un-

necessary tokens.

3 Problem Formulation

Let $\mathcal{D} = \{d_i\}_{i=1}^{|\mathcal{D}|}$ denote a document corpus and $\mathcal{T} = \{(q_i, d_i^+)\}_{i=1}^{|\mathcal{T}|}$ denote training query-document pairs. In GR, each document d is assigned a discrete DocID sequence $c_d = (c_d^1, c_d^2, \dots, c_d^L)$, and the set of corpus-valid DocIDs is $\mathcal{C} = \{c_d \mid d \in \mathcal{D}\}$. Given a query q , retrieval is performed by constrained autoregressive decoding over $\mathcal{C}$ and the generated DocIDs are mapped back to documents.

GEAR extends this formulation by allowing the model to generate an optional retrieval-oriented reasoning prefix before DocID decoding. Let $\mathcal{V}_{\text{nl}}$ and $\mathcal{V}_{\text{doc}}$ denote the natural-language and DocID vocabularies, respectively. We introduce a special transition token `<retrieve>` and define the overall vocabulary as $\mathcal{V} = \mathcal{V}_{\text{nl}} \cup \mathcal{V}_{\text{doc}} \cup \{\text{<retrieve>}\}$. For a query q , GEAR generates a sequence $y = z \circ \text{<retrieve>} \circ c_d$, where z is an optional reasoning prefix and $c_d \in \mathcal{C}$ is a corpus-valid DocID. If `<retrieve>` is generated as the first token, then $z = \emptyset$ and the model performs zero-reasoning retrieval. Otherwise, z serves as retrieval-oriented context for subsequent DocID decoding. The generation probability is factorized as $p_\theta(y \mid q) = p_\theta(z, \text{<retrieve>} \mid q) \cdot p_\theta(c_d \mid q, z, \text{<retrieve>}; \mathcal{C})$, where the second term is decoded under the corpus-valid DocID constraint.

4 Method

As shown in Figure 2, GEAR is trained in three main stages: (i) mixed DocID-language continued pretraining aligns dedicated DocID tokens with natural-language semantics. (ii) difficulty-aware supervised fine-tuning initializes query-dependent trajectories for easy, medium, and hard queries. (iii) self-budgeted reasoning reinforcement learning directly optimizes the trade-off between retrieval quality and reasoning cost. At inference time, GEAR generates natural-language tokens under a vocabulary mask until `<retrieve>` is emitted, and then switches to trie-constrained DocID decoding; if the maximum reasoning budget is reached, the transition is forced. We next describe the three training stages.

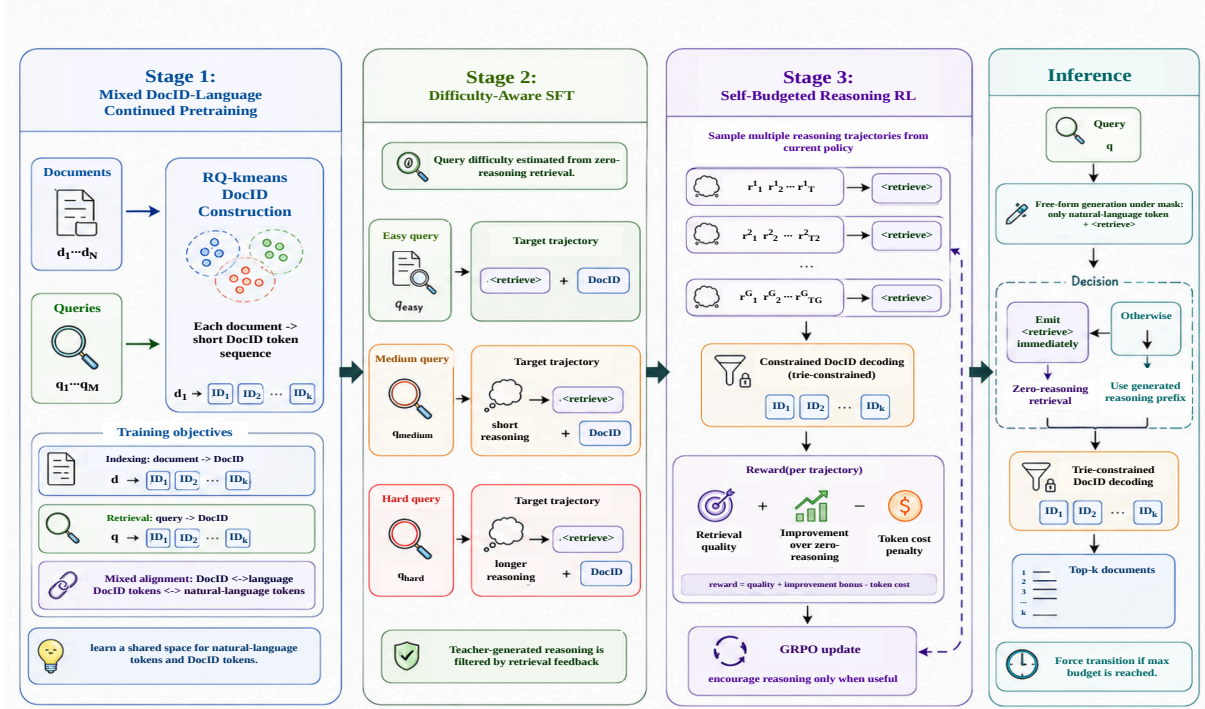


Figure 2: Overview of GEAR. GEAR uses mixed DocID-language continued pretraining, difficulty-aware supervised fine-tuning, and self-budgeted reinforcement learning to learn when to stop reasoning and switch to constrained DocID decoding via `<retrieve>`.

4.1 Mixed DocID-language Continued Pretraining

This stage connects natural-language reasoning with discrete DocID generation. The model is trained to generate DocID tokens for retrieval and to align DocID tokens with natural-language semantics in a shared generation space.

DocID construction. Following prior GR work (Zhang et al., 2023; Zeng et al., 2024a), we construct discrete DocIDs with RQ-kmeans over document representations. Each document d is encoded into a dense vector $\mathbf{h}_d$ and quantized into a length- L sequence of residual code-word indices. The resulting DocID is denoted as $c_d = (c_d^1, c_d^2, \dots, c_d^L)$, where each c_d^j is represented as a dedicated token from $\mathcal{V}_{\text{doc}}$.

DocID generation training. We first train the model to generate DocIDs from both documents and queries. The indexing objective is

$$\mathcal{L}_{\text{ind}} = - \sum_{d \in \mathcal{D}} \log p_{\theta}(c_d | d; \mathcal{C}), \quad (1)$$

and the retrieval objective is

$$\mathcal{L}_{\text{ret}} = - \sum_{(q, d^+) \in \mathcal{T}} \log p_{\theta}(c_{d^+} | q; \mathcal{C}). \quad (2)$$

These objectives teach the model to map documents and queries into the same DocID space (Tay et al.,

2022; Wang et al., 2022), but they do not by themselves ensure that DocID tokens interact well with free-form reasoning tokens.

Mixed training. To bridge this token-space gap, we further construct mixed DocID-language training data. Specifically, we use two types of mixed data: (i) DocID-to-language paired data, where the model recovers document or query text from a DocID, such as reconstructing the corresponding document summary or its associated query; and (ii) continuous DocID-language sequences, where DocID tokens and natural-language tokens appear in the same autoregressive context, such as inserting DocID tokens into document descriptions or query-document matching statements. This exposes the model to settings in which DocIDs serve not only as discrete retrieval labels, but also as semantic anchors that can be interpreted and conditioned on by the language model. These objectives encourage DocID tokens to carry natural-language semantics and allow retrieval-oriented reasoning prefixes to condition subsequent DocID decoding. Detailed construction of the mixed training data is provided in Appendix A.

Overall continued pretraining objective. The overall continued pretraining objective is

$$\mathcal{L}_{\text{mix}} = \mathcal{L}_{\text{ind}} + \mathcal{L}_{\text{ret}} + \mathcal{L}_{\text{align}}, \quad (3)$$

where $\mathcal{L}_{align}$ denotes the combined mixed DocID-language alignment objectives.

4.2 Difficulty-aware SFT

This stage constructs a difficulty-aware SFT dataset to initialize the model with the trajectory format of reasoning followed by `<retrieve>` and with an initial query-dependent reasoning policy.

Data construction. For each training pair $(q, d^+) \in \mathcal{T}$, we first run zero-reasoning retrieval with the mixed-pretrained model:

$$\mathcal{R}_{1:k}^0 = \text{Decode}_k(q, \emptyset; \mathcal{C}), \quad (4)$$

where $z = \emptyset$ means that decoding starts immediately after `<retrieve>`, $\text{Decode}_k(\cdot)$ denotes top- k constrained DocID decoding. Let $\text{rank}_0(q, d^+)$ be the rank of c_{d^+} in $\mathcal{R}_{1:k}^0$, with $\text{rank}_0(q, d^+) = \infty$ if it is not retrieved in the top- k list. Based on $\text{rank}_0(q, d^+)$, we divide queries into easy, medium, and hard groups and construct target trajectories as

$$\begin{aligned} y_{\text{easy}} &= \text{<retrieve>} \circ c_{d^+}, \\ y_{\text{med}} &= z_{\text{short}} \circ \text{<retrieve>} \circ c_{d^+}, \\ y_{\text{hard}} &= z_{\text{long}} \circ \text{<retrieve>} \circ c_{d^+}, \end{aligned} \quad (5)$$

where the three trajectories are used for queries with $\text{rank}_0(q, d^+) \leq r$, $r < \text{rank}_0(q, d^+) \leq k$, and $\text{rank}_0(q, d^+) = \infty$, respectively. Easy queries are trained to skip reasoning, medium queries receive a short retrieval-oriented prefix, and hard queries receive a longer prefix under a larger reasoning budget. For medium and hard queries, we use an offline teacher to generate retrieval-oriented prefixes conditioned only on the query, encouraging the prefix to highlight key entities, semantic constraints, and disambiguating clues without exposing the positive DocID. We then filter the generated prefixes by running constrained DocID decoding with each prefix and retain only those that improve the retrieval score over the zero-reasoning result. Appendix B provides more details.

SFT training. All constructed examples form the difficulty-aware SFT dataset $\mathcal{S}$. We train the model with the standard autoregressive CE objective:

$$\mathcal{L}_{sft} = - \sum_{(q,y) \in \mathcal{S}} \log p_\theta(y | q). \quad (6)$$

Through this stage, the model learns an initial transition policy for emitting `<retrieve>`.

4.3 Self-Budgeted Reasoning RL

We further optimize GEAR with self-budgeted reasoning RL, where the reward directly balances retrieval quality and reasoning cost. Specifically, for each query q , we sample a group of reasoning trajectories from the current policy:

$$\tau_i = z_i \circ \text{<retrieve>}, \quad (7)$$

where z_i is the generated reasoning prefix. The policy trajectory only contains the free-form reasoning prefix and the transition action. Conditioned on $q \circ z_i \circ \text{<retrieve>}$, we run constrained top- k DocID decoding and obtain a ranked list $\mathcal{R}_{1:k}^i = \text{Decode}_k(q, z_i; \mathcal{C})$. We also compute a zero-reasoning baseline $\mathcal{R}_{1:k}^0 = \text{Decode}_k(q, \emptyset; \mathcal{C})$ using the same policy checkpoint. If `<retrieve>` is not emitted within the maximum reasoning budget B_{max} , the transition is forced.

Let $S_i = S(q, \mathcal{R}_{1:k}^i)$ and $S_0 = S(q, \mathcal{R}_{1:k}^0)$, where $S(\cdot)$ is a bounded retrieval metric.¹ We normalize the reasoning length as $\ell_i = |z_i|/B_{\text{max}}$. The reward for trajectory τ_i is defined as

$$\rho_i = S_i + \eta[S_i - S_0]_+ - \lambda\ell_i, \quad (8)$$

where $[x]_+ = \max(x, 0)$, η controls the bonus for improving over zero-reasoning retrieval, and λ controls the normalized reasoning cost. This reward favors reasoning prefixes that improve the downstream DocID ranking, while penalizing unnecessarily long reasoning.

We optimize the reasoning policy with GRPO (Shao et al., 2024). For each query, trajectories in the same sampled group are compared according to their rewards, and the model is updated to increase the likelihood of trajectories with higher relative rewards. We further add KL regularization to the SFT model for stable training. The full optimization objective is provided in Appendix C.

5 Experimental setup

Datasets and Metrics. We evaluate GEAR on three standard document retrieval benchmarks. (i) *NQ* contains 320K query-document pairs and approximately 100K Wikipedia documents, where queries are posed as natural-language questions (Kwiatkowski et al., 2019). (ii) *MS MARCO* consists of real user queries from Bing search with

¹In practice, we use $\text{MRR}@k$ for NQ and MS MARCO, and $\text{nDCG}@k$ for TREC DL.

Method	NQ			MS MARCO			TREC DL	
	Hits@1	Hits@10	MRR@10	Hits@1	Hits@10	MRR@10	nDCG@10	Recall@10
BM25	19.3	29.5	25.6	20.4	27.8	24.1	41.2	12.4
DocT5Query	23.1	36.9	31.3	24.9	31.5	27.0	49.7	19.8
DPR	31.8	40.3	36.4	29.6	35.0	33.5	45.1	18.7
ANCE	35.5	42.8	36.9	31.8	38.6	35.7	52.4	20.1
DSI	24.5	38.1	34.2	23.9	33.2	28.1	23.8	9.3
DSI-QG	32.7	44.0	39.3	30.2	38.5	35.3	36.6	15.4
SEAL	35.2	51.4	46.5	33.0	50.2	42.0	42.8	18.2
MINDER	38.6	56.5	50.0	37.8	53.0	45.6	50.3	19.6
LTRGR	44.1	56.6	54.4	40.9	56.8	48.6	54.1	20.4
TSGen	41.2	58.5	52.3	37.6	49.4	45.4	54.0	18.8
RIPOR	50.5	66.3	59.6	44.2	60.7	50.0	63.5	21.4
PAG	48.1	66.8	61.7	44.0	61.2	51.8	66.2	21.9
ReasonGR	46.2	68.2	60.0	43.9	61.8	51.6	65.4	19.2
R4R	<u>53.8</u>	71.0	68.5	<u>45.7</u>	66.7	53.5	<u>66.8</u>	<u>21.7</u>
GEAR	54.4	<u>70.2</u>	<u>66.8</u>	46.1	<u>64.8</u>	<u>54.0</u>	67.0	21.5

Table 1: Main results on NQ, MS MARCO, and TREC DL 2022. GEAR achieves competitive or state-of-the-art effectiveness while maintaining strong efficiency compared with fixed reasoning-enhanced GR (shown in Figure 3).

human-annotated relevance judgments (Bajaj et al., 2016). Following prior GR work, we use a subset with 360K query-document pairs and 320K documents for training and evaluation. (iii) *TREC DL* 2022 is the document ranking task from the TREC 2022 Deep Learning Track. It is built upon MS MARCO v2 (Craswell et al., 2025).

Following previous studies, we report Hits@ k and MRR@ k on NQ and MS MARCO (Tay et al., 2022; Bevilacqua et al., 2022; Li et al., 2024b). For TREC DL, we report nDCG@ k and Recall@ k (Zeng et al., 2024a,b). In addition to effectiveness metrics, we report the average end-to-end latency per query and the number of generated tokens per query to evaluate retrieval efficiency. We provide detailed latency measurement in D.4.

Baselines. (i) *Standard GR methods*, including DSI (Tay et al., 2022), DSI-QG (Zhuang et al., 2022), SEAL (Bevilacqua et al., 2022), MINDER (Li et al., 2023), LTRGR (Li et al., 2024b), TSGen (Zhang et al., 2024), RIPOR (Zeng et al., 2024a), and PAG (Zeng et al., 2024b). We reproduce these baselines as closely as possible following the descriptions in the original papers, and use the same backbone as GEAR for fair comparison. Detailed implementations are provided in Appendix D.2. (ii) *Adapted reasoning-enhanced GR methods*, including ReasonGR (Dong et al., 2026) and R4R (Zhang et al., 2025b). Since the original methods were developed under different tasks, corpora, and identifier designs, we adapt their core reasoning procedures to our unified GR setting. For controlled comparison, we use the same Qwen3-8B

backbone, dataset splits, beam size, hardware, and evaluation protocol as GEAR whenever applicable. Because the original settings are not directly shared with ours, their absolute reported numbers cannot be exactly reproduced under this unified setting. Detailed adaptations and complete prompts are provided in Appendix D.3. (iii) *Sparse and dense retrieval baselines*, including BM25 (Robertson et al., 2009), DocT5Query (Nogueira et al., 2019), DPR (Karpukhin et al., 2020), and ANCE (Xiong et al., 2020).

Implementation details. We implement GEAR with Qwen3-8B (Yang et al., 2025) as the backbone. For DocID construction, we encode documents with BERT and apply RQ-kmeans (Chen et al., 2010) with four residual quantization levels and a codebook size of 256, yielding a four-token DocID for each document. During inference, GEAR generates natural-language tokens under a vocabulary mask until <retrieve> is emitted or the maximum reasoning budget $B_{\max} = 256$ is reached, and then performs trie-constrained beam search with beam size 10. All experiments are conducted on two NVIDIA A100 GPUs. Full training hyperparameters and implementation details are provided in Appendix D.1.

6 Experimental results

Our experiments answer the following research questions: **(RQ1)** Does GEAR improve retrieval effectiveness compared with existing generative retrieval methods? **(RQ2)** How efficient is GEAR in terms of reasoning cost, latency, and token us-

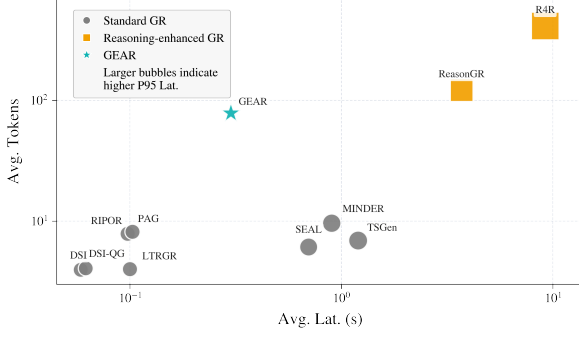


Figure 3: Efficiency comparison of GR methods, where bubble size indicates P95 latency.

age? **(RQ3)** Does GEAR learn to adaptively allocate reasoning across queries of different difficulty? **(RQ4)** How do different design components affect the effectiveness-efficiency trade-off of GEAR?

6.1 RQ1: Main results

We compare GEAR with various baselines. As shown in Table 1, we observe that: (i) GEAR achieves competitive performance across all three datasets, obtaining the best or near-best results on several metrics while remaining close to the strongest baselines on others. (ii) Compared with standard GR methods, GEAR consistently improves retrieval effectiveness, showing that retrieval-oriented reasoning can provide useful context beyond direct query-to-DocID decoding. (iii) Compared with fixed reasoning-enhanced GR methods, GEAR achieves comparable effectiveness without reasoning for every query, indicating that adaptive reasoning preserves retrieval quality while avoiding unnecessary reasoning. These results show that GEAR provides a strong effectiveness-efficiency trade-off.

6.2 RQ2: Efficiency results

We evaluate inference efficiency on all three datasets and aggregate the results using four metrics: non-empty reasoning rate, average generated tokens, average end-to-end latency, and P95 latency. Figure 3 visualizes the latter three metrics, where the x-axis denotes average latency, the y-axis denotes average token count, and bubble size indicates P95 latency (smaller is better). *The full results are reported in Table 4 in Appendix E.* On average, GEAR triggers reasoning for 37.1% of queries, compared with 0% for standard GR and 100% for fixed reasoning-enhanced GR, under the same backbone and hardware settings.

We draw the following observations: (i) Standard GR methods are efficient because they directly generate DocIDs without intermediate reasoning.

Dataset	Group (Ratio)	ZRR	w/o Reason.	w/ Reason.
NQ	Easy (55.7)	90.8	60.3	64.6
	Medium (28.6)	42.9	0.0	58.4
	Hard (15.7)	0.9	0.0	10.9
MS MARCO	Easy (52.3)	89.4	59.5	60.7
	Medium (30.8)	50.8	0.0	43.9
	Hard (16.9)	2.6	0.0	4.9
TREC DL	Easy (57.6)	90.6	65.2	77.6
	Medium (27.8)	36.8	8.7	67.9
	Hard (14.6)	3.0	2.4	23.5

Table 2: Breakdown analysis by query difficulty. “ZRR” denotes the zero-reasoning rate.

(ii) Fixed reasoning-enhanced GR methods introduce substantial overhead. ReasonGR generates 121.5 tokens per query with 3.7s average latency, while R4R generates 412.3 tokens and reaches 9.2s average latency and 15.1s P95 latency. (iii) GEAR substantially reduces the cost of reasoning-enhanced retrieval and effectively controls tail latency, by reasoning only when needed, while maintaining strong retrieval effectiveness. (iv) Methods based on trie-constrained decoding with numerical DocIDs are generally more efficient, such as DSI, because their constraint computation is fast and the DocID length is fixed. In contrast, methods based on FM-index constraints and textual DocIDs, such as SEAL, typically incur higher decoding overhead and thus exhibit lower efficiency. Therefore, GEAR adopts the former implementation. These results show that GEAR improves the effectiveness-efficiency trade-off.

6.3 RQ3: Adaptive reasoning behavior

We analyze how GEAR allocates reasoning across queries of different difficulty levels and how this behavior interacts with the reasoning budget during inference.

Difficulty-wise analysis. We first analyze the performance of GEAR under different query difficulty levels. Following the difficulty estimation strategy used during training, we divide test queries into easy, medium, and hard groups according to the rank position of the relevant document under zero-reasoning retrieval. The performance metrics are Hits@1 for NQ and MS MARCO, and nDCG@10 for TREC DL. “w/o Reason.” reports the performance of the zero-reasoning baseline, and “w/ Reason.” reports the performance of the full GEAR with adaptive reasoning.

As shown in Table 2, we draw the following observations: (i) Easy queries account for more than half of all queries across the three datasets. (ii) GEAR learns to skip reasoning for most easy

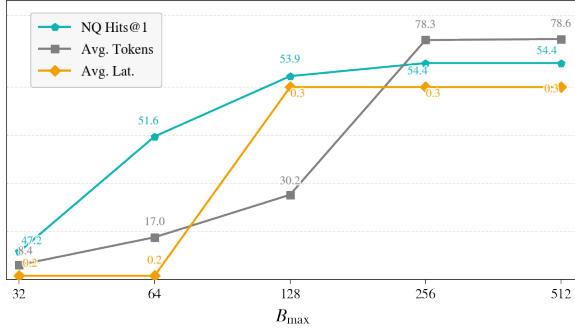


Figure 4: Effect of the maximum reasoning budget on the effectiveness-efficiency trade-off on NQ.

queries, while invoking reasoning more frequently for medium and hard queries. This indicates that GEAR learns to allocate reasoning only when direct retrieval is less reliable. (iii) Adaptive reasoning brings significant improvements on both medium and hard queries. These results show that GEAR not only reduces reasoning cost, but also applies reasoning to the queries that need it most.

Reasoning budget analysis. To investigate the effect of the maximum reasoning budget, we vary $B_{\max}$ from 32 to 512 on NQ and report retrieval effectiveness, generated tokens, and inference latency. As shown in Figure 4, we observe that: (i) A too small budget, especially below 128, limits retrieval effectiveness because reasoning may be prematurely truncated for harder queries. (ii) Increasing $B_{\max}$ beyond 128 brings only marginal gains, indicating that GEAR does not require an excessively large reasoning budget. (iii) The average number of generated tokens and latency do not grow linearly with $B_{\max}$, suggesting that the model often emits `<retrieve>` before consuming the full budget.

6.4 RQ4: Model analysis and ablation

We analyze how different design choices and modeling decisions affect both the effectiveness and efficiency of GEAR.

Ablation study. We conduct ablation experiments on NQ by comparing the full GEAR model with five variants: removing mixed DocID-language training, removing difficulty-aware SFT, removing self-budgeted RL, removing reasoning, and replacing adaptive reasoning with fixed reasoning. As shown in Figure 5, we observe that: (i) Removing any component hurts performance, confirming the importance of the integrated design. (ii) Removing mixed DocID-language training causes a large drop, showing that aligning DocID tokens with

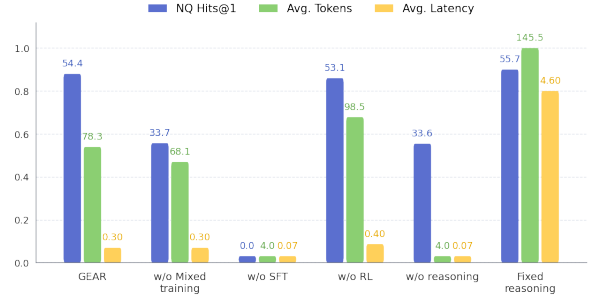


Figure 5: Ablation study on NQ. Removing any component of GEAR degrades performance, highlighting the importance of each design choice.

Method	Hits@1	Hits@10	MRR@10	Avg. Lat.
PRF Expansion	38.1	46.7	42.5	—
Qwen3-8B Rewrite	48.7	65.9	61.3	3.8s
GPT-5 Rewrite	52.3	68.4	67.1	5.7s
GEAR	54.4	70.2	66.8	0.3s

Table 3: Retrieval performance and inference latency of GEAR and query rewriting baselines on NQ.

natural-language semantics is crucial. (iii) Removing difficulty-aware SFT makes the model fail to learn the reasoning-`<retrieve>`-DocID trajectory. (iv) Self-budgeted RL improves both effectiveness and efficiency by encouraging reasoning only when its expected retrieval gain justifies the token cost. (v) Fixed reasoning can improve retrieval effectiveness, but incurs substantially higher inference cost than adaptive reasoning.

Comparison with query rewriting. We compare GEAR with three query reformulation baselines on NQ: pseudo-relevance feedback (PRF) (Lv and Zhai, 2009), which performs a one-round initial retrieval and expands the query using terms extracted from the top-ranked pseudo-relevant documents, and LLM-based query rewriting with Qwen3-8B and GPT-5, respectively. Unlike these methods, which reformulate every query before the final retrieval stage, GEAR integrates adaptive reasoning and DocID decoding into a single generative process. We omit PRF latency because it is implemented with an offline sparse retrieval expansion pipeline and is not directly comparable to neural generation latency. As shown in Table 3, we observe that: (i) LLM-based rewriting improves over PRF expansion but incurs high latency. (ii) GEAR achieves the best Hits@1 and Hits@10, with MRR@10 close to GPT-5 rewriting, indicating that its reasoning prefix provides retrieval-oriented query understanding for DocID generation. (iii) By reasoning only when needed, GEAR avoids the cost of rewriting every query and offers a more efficient alternative to external query rewriting.

7 Conclusion

In this work, we presented GEAR, a generative retrieval framework with adaptive reasoning that allocates computation based on query-specific needs. Unlike prior reasoning-enhanced GR methods that apply reasoning uniformly, GEAR learns when to reason and when to directly perform DocID decoding via a learned <retrieve> transition. GEAR is trained to balance retrieval effectiveness and efficiency through mixed DocID-language pretraining, difficulty-aware supervised fine-tuning, and self-budgeted reinforcement learning. This design enables sparse reasoning that activates intermediate reasoning only when it provides sufficient retrieval benefit. Experiments on NQ, MS MARCO, and TREC DL 2022 demonstrate that GEAR maintains retrieval effectiveness comparable to strong reasoning-enhanced GR methods while substantially reducing reasoning cost, average latency, and tail latency, highlighting the effectiveness of adaptive reasoning for efficient generative retrieval.

Limitations

The limitations of this work are as follows: (i) GEAR relies on relatively large language models with reasoning ability. This is an inherent consequence of introducing explicit reasoning into GR, and makes its training and inference costs still higher than those of lightweight non-reasoning GR methods. Future work needs to explore methods for further reducing the computational cost. (ii) The difficulty estimation used in GEAR is still relatively coarse and may not fully capture the fine-grained reasons behind query difficulty. More fine-grained difficulty modeling could further optimize the allocation of reasoning resources. (iii) GEAR relies on teacher-generated reasoning data during the supervised fine-tuning stage, and the quality of the teacher model can affect the initial reasoning behavior learned by GEAR. Future work may reduce the reliance on external teachers through retrieval-feedback-based data construction or automatically mined reasoning trajectories. (iv) Our experiments mainly focus on standard document retrieval benchmarks with offline evaluation. These benchmarks may not fully reflect more complex real-world search environments, such as multi-turn retrieval or personalized retrieval. We leave these directions for future work.

Ethical Considerations

This work analyzes baselines and our newly proposed method using existing benchmark datasets and reports aggregate results only. It does not collect new user data or involve human subjects.

Acknowledgements

This work was funded by the Strategic Priority Research Program of the CAS under Grants No. XDB0680102, the National Natural Science Foundation of China (NSFC) under Grants No. U25B2076, 62472408 and 62441229, and the National Key Research and Development Program of China under Grants No. 2023YFA1011602. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

- Giambattista Amati, Claudio Carpineto, and Giovanni Romano. 2004. Query difficulty, robustness, and selective application of query expansion. In *European conference on information retrieval*, pages 127–137. Springer.
- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683.
- Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022a. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2184–2189.
- Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022b. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 191–200.
- Yongjian Chen, Tao Guan, and Cheng Wang. 2010. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M Voorhees, and Ian

- Soboroff. 2025. Overview of the trec 2022 deep learning track. *arXiv preprint arXiv:2507.10865*.
- Suchana Datta, Debasis Ganguly, Sean MacAvaney, and Derek Greene. 2024. A deep learning approach for selective relevance feedback. In *European Conference on Information Retrieval*, pages 189–204. Springer.
- Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.
- Steven Dong, Yubao Tang, and Maarten de Rijke. 2026. Multi-step semantic reasoning in generative retrieval. In *European Conference on Information Retrieval*, pages 273–281. Springer.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.
- Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, Suhang Wang, Jiawei Han, and Xianfeng Tang. 2024. Language models as semantic indexers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22244–22259. PMLR.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024a. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.
- Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview identifiers enhanced generative retrieval. *arXiv preprint arXiv:2305.16675*.
- Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024b. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8716–8723.
- Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, and 1 others. 2025. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*.
- Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*.
- Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1895–1898.
- Alliot Nagle, Jakhongir Saydaliev, Dhia Garbaya, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. 2026. Terminator: Learning optimal exit points for early stopping in chain-of-thought reasoning. *arXiv preprint arXiv:2603.12529*.
- Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.
- Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2322–2331.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten De Rijke, Wei Chen, and Xueqi Cheng. 2024. List-wise generative retrieval models via a sequential learning process. *ACM Transactions on Information Systems*, 42(5):1–31.
- Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, and 1 others. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, and 1 others. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024a. Scalable and effective generative information retrieval. In *Proceedings of the ACM Web Conference 2024*, pages 1441–1452.

Hansi Zeng, Chen Luo, and Hamed Zamani. 2024b. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 469–480.

Hailin Zhang, Yujing Wang, Qi Chen, Ruiheng Chang, Ting Zhang, Ziming Miao, Yingyan Hou, Yang Ding, Xupeng Miao, Haonan Wang, and 1 others. 2023. Model-enhanced vector index. *Advances in Neural Information Processing Systems*, 36:54903–54917.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. Adaptthink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 458–468.

Yingchen Zhang, Ruqing Zhang, Jiafeng Guo, Wenjun Peng, Sen Li, and Fuyu Lv. 2025b. Retrieval-in-the-chain: Bootstrapping large language models for generative retrieval. *arXiv preprint arXiv:2510.13095*.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for

differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

A Mixed DocID-language training data

We instantiate $\mathcal{L}_{align}$ with two types of mixed DocID-language objectives.

DocID-to-language paired data. Given a DocID and an instruction prompt, the model is trained to recover the corresponding document-side text or query. For document reconstruction, we use

$$\mathcal{L}_{id2d} = - \sum_{d \in \mathcal{D}} \log p_{\theta}(t_d | c_d, u_d), \quad (9)$$

where t_d is the document-side text and u_d is the instruction prompt. For query reconstruction, we use

$$\mathcal{L}_{id2q} = - \sum_{(q, d^+) \in \mathcal{T}} \log p_{\theta}(q | c_{d^+}, u_q). \quad (10)$$

Continuous DocID-language sequence data. We also construct sequences in which DocID tokens and natural-language tokens appear in the same context, e.g., $s_d = \text{“The main summary of”} \circ c_d \circ \text{“is”} \circ \tilde{t}_d$, where $\tilde{t}_d$ is a short document summary. The language modeling objective is

$$\mathcal{L}_{seq} = - \sum_{s \in \mathcal{M}} \sum_{t=1}^{|s|} \log p_{\theta}(s_t | s_{<t}), \quad (11)$$

where $\mathcal{M}$ denotes the set of constructed continuous DocID-language sequences.

Thus, the alignment objective used in the main text is

$$\mathcal{L}_{align} = \mathcal{L}_{id2d} + \mathcal{L}_{id2q} + \mathcal{L}_{seq}. \quad (12)$$

B Teacher reasoning generation and filtering

For medium and hard training queries, we use an offline teacher to generate retrieval-oriented reasoning prefixes. The teacher is given the query q , but is not given the positive document d^+ and the DocID token sequence.

For medium queries, the teacher generates a concise prefix z_{short} under a budget B_{short} , focusing on key entities, constraints, and disambiguating terms. For hard queries, the teacher generates a longer prefix z_{long} under a budget $B_{long} > B_{short}$, summarizing query intent and expected document semantics.

To reduce noisy or redundant supervision, we filter teacher-generated prefixes with retrieval feedback. Given a candidate prefix z , we run constrained decoding conditioned on $q \circ z \circ \langle \text{retrieve} \rangle$ and obtain $\mathcal{R}_{1:k}^z$. We retain z only if it improves the retrieval score over the zero-reasoning baseline $\mathcal{R}_{1:k}^0$.

C GRPO objective

For each query q , we sample a group of G reasoning trajectories $\{\tau_i\}_{i=1}^G$ from the old policy $\pi_{\theta_{\text{old}}}$ and compute their rewards $\{\rho_i\}_{i=1}^G$ using Eq. 8. GRPO estimates the advantage of each trajectory by normalizing rewards within the sampled group:

$$A_i = \frac{\rho_i - \text{mean}(\{\rho_j\}_{j=1}^G)}{\text{std}(\{\rho_j\}_{j=1}^G)}. \quad (13)$$

The clipped policy optimization objective is

$$\mathcal{L}_{rl} = -\frac{1}{G} \sum_{i=1}^G \min \left(\omega_i(\theta) A_i, \right. \\ \left. \text{clip}(\omega_i(\theta), 1 - \epsilon, 1 + \epsilon) A_i \right), \quad (14)$$

where

$$\omega_i(\theta) = \frac{\pi_{\theta}(\tau_i | q)}{\pi_{\theta_{\text{old}}}(\tau_i | q)} = \frac{\pi_{\theta}(z_i, \langle \text{retrieve} \rangle | q)}{\pi_{\theta_{\text{old}}}(z_i, \langle \text{retrieve} \rangle | q)} \quad (15)$$

is the policy ratio over the reasoning trajectory and transition action.

To stabilize training and keep the reasoning policy close to the SFT model, we add KL regularization with respect to a reference model π_{ref} :

$$\mathcal{L} = \mathcal{L}_{rl} + \beta_{\text{KL}} \text{KL}(\pi_{\theta} \| \pi_{\text{ref}}), \quad (16)$$

where β_{KL} controls the strength of regularization.

D Implementation details

D.1 Implementation details of GEAR

For DocID construction, we encode each document with BERT-base and apply RQ-kmeans over the resulting document representations. We use four residual quantization levels with a codebook size of 256 at each level. Each document is assigned a DocID sequence consisting of four tokens, where each token is represented by a dedicated symbol from the DocID vocabulary $\mathcal{V}_{\text{doc}}$.

All stages of GEAR, including mixed DocID-language continued pretraining, difficulty-aware

SFT, and self-budgeted reasoning RL, are trained only on the corresponding training split. For NQ and MS MARCO, we follow the official dataset splits. For TREC DL, we follow the standard setting that uses the MS MARCO v2 document ranking training data for training and evaluates on the official TREC DL queries. The TREC DL queries and relevance judgments are used only for evaluation, and are not used in continued pretraining, supervised fine-tuning, reinforcement learning, or difficulty estimation.

We implement GEAR based on Qwen3-8B. During training, we use the Transformers library for mixed DocID-language continued pretraining and difficulty-aware SFT, and use the TRL library for self-budgeted reasoning RL. All training stages use bfloat16 precision and the AdamW optimizer. For continued pretraining, we train the model for 1 epoch with a learning rate of 1×10^{-5} . For difficulty-aware SFT, we train the model for 3 epochs with a learning rate of 2×10^{-5} . For difficulty estimation in SFT, we set the retrieval depth to $k = 100$ and the easy-query threshold to $r = 10$. Queries whose positive document is ranked within the top 10 are treated as easy, those ranked between 11 and 100 are treated as medium, and those not retrieved in the top 100 are treated as hard.

For teacher-generated reasoning, we set the short and long reasoning budgets to $B_{\text{short}} = 32$ and $B_{\text{long}} = 96$, respectively. For RL, we use GRPO with a learning rate of 5×10^{-6} , a group size of 4, a clipping coefficient of 0.2, and a KL coefficient of 0.01. In the reward function, we set the improvement-bonus weight to $\eta = 0.5$ and the normalized reasoning-length penalty to $\lambda = 0.1$. Other hyperparameters follow the default settings.

During inference, GEAR first generates natural-language reasoning tokens under a vocabulary mask that disallows DocID tokens. Generation continues until $\langle \text{retrieve} \rangle$ is emitted or the maximum reasoning budget $B_{\text{max}} = 256$ is reached. After that, decoding switches to trie-constrained beam search for DocID generation, with the beam size set to 10 by default. All experiments are conducted on a server equipped with two NVIDIA A100 GPUs.

D.2 Implementation details of standard GR baselines

For standard GR baselines, we reproduce the compared methods under a unified setting. All methods use the same training and test splits as GEAR, and

use the same Qwen3-8B backbone whenever applicable. Since existing generative retrieval methods differ in their identifier designs, we adapt their document identifiers accordingly. For methods with numerical DocIDs, we use the same RQ-kmeans-based DocID construction as GEAR. For methods with textual DocIDs, we use the original document titles on NQ. Since MS MARCO and TREC DL do not provide document titles, we use GPT-5 to generate pseudo titles from document contents and use them to construct textual identifiers. During inference, all GR baselines use the corresponding constrained beam search described in the original papers, such as trie constraints, FM-index constraints, or inverted-index constraints. The beam size is set to 10 for all methods.

D.3 Implementation details of reasoning-enhanced GR baselines

Implementation details of ReasonGR. ReasonGR was originally designed for financial document retrieval on FinQA, and therefore does not fully match our experimental setting. We adapt it to our benchmarks with Qwen3-8B as follows: (i) The original ReasonGR constructs textual DocIDs by extracting keywords with KeyBERT and prefixing them with the company name and year. Since such metadata is not available in our datasets, we follow the standard GR setting with textual DocIDs and use GPT-5 to generate pseudo titles from document contents and use them as textual DocIDs. (ii) The reasoning training data in ReasonGR is obtained from model-generated trajectories. For a fair comparison, we also use GPT-5 as the teacher model to generate reasoning trajectories, following the same teacher-generation setup as GEAR.

The adapted ReasonGR differs from GEAR in three main aspects: (i) It performs reasoning for every query. (ii) Inference is carried out in two stages: the model first freely generates a reasoning prefix, and then the generated reasoning is concatenated with the query as context for constrained DocID decoding. (iii) ReasonGR contains only indexing/retrieval SFT and reasoning SFT, without difficulty-aware reasoning allocation or self-budgeted reinforcement learning.

Implementation details of R4R. R4R is a prompt-based think–retrieve–rethink framework. We implement it following the procedure described in the original paper. Specifically, we use Qwen3-8B and alternate between two decoding modes: free-

Method	Reason. Rate	Avg. Tokens	Avg. Lat. (s)	P95 Lat. (s)
DSI	0.0	4.0	0.06	0.08
DSI-QG	0.0	4.0	0.06	0.07
SEAL	0.0	6.1	0.7	0.9
MINDER	0.0	9.6	0.9	1.1
LTRGR	0.0	4.0	0.1	0.1
TSGen	0.0	6.9	1.2	1.4
RIPOR	0.0	8.0	0.1	0.1
PAG	0.0	8.0	0.1	0.1
ReasonGR	100.0	121.5	3.7	5.0
R4R	100.0	412.3	9.2	15.1
GEAR	37.1	78.3	0.3	2.7

Table 4: Efficiency comparison.

form generation and constrained DocID generation. Given a query, the model first freely generates an initial reasoning step, then performs constrained generation to produce a DocID, and then freely generates a revised reasoning step based on the intermediate retrieval result.

D.4 Latency measurement protocol

We measure the inference efficiency of all GR methods under the same hardware and software environment. All methods are evaluated on a server equipped with two NVIDIA A100 GPUs. End-to-end latency includes all online computation required to return the top- k retrieval results for a query. For GEAR, this includes masked free-form reasoning generation, emitting <retrieve> or forcing the transition at $B_{\max}$, trie-constrained DocID decoding, and mapping the generated DocIDs back to documents. For standard GR baselines, latency includes direct constrained DocID generation and document mapping. For reasoning-enhanced GR baselines, latency includes all reasoning, retrieval, and rethinking steps used before producing the final ranked list. We set the measurement batch size to 1 and the beam size for constrained decoding to 10.

For each method, we report the average number of generated tokens per query, the average end-to-end latency per query, and the P95 end-to-end latency. The generated-token count includes all tokens generated during online inference, including reasoning tokens, transition tokens, and DocID tokens. For methods with multiple reasoning or retrieval rounds, tokens from all rounds are counted. P95 latency is computed over all evaluation queries and reflects tail latency.

E Numerical results of efficiency experiment

The numerical results corresponding to Figure 3 in the main text are shown in Table 4.

F Examples of generation

We provide two representative examples to illustrate how GEAR allocates reasoning budgets at inference time. For a simple query, GEAR emits <retrieve> immediately and performs zero-reasoning retrieval. For a more difficult query, GEAR first generates a retrieval-oriented reasoning prefix and then switches to constrained DocID decoding.

Case 1: Zero-reasoning retrieval.

Query: Who wrote the novel The Fault in Our Stars?

Generated trajectory: <retrieve>
[DocID]

Case 2: Retrieval with reasoning.

Query: What treaty ended the War of 1812 between the United States and Britain?

Generated trajectory: The query asks for the formal agreement that ended the War of 1812. The relevant document should describe the peace settlement between the United States and Britain. A useful clue is that the treaty was signed in Ghent and is commonly known as the Treaty of Ghent. Therefore, the target document should focus on the historical treaty that concluded the conflict.
<retrieve> [DocID]