LatentRAG: Latent Reasoning and Retrieval for Efficient Agentic RAG

Yijia Zheng **Marcel Worring**
University of Amsterdam, Amsterdam, the Netherlands
{y.zheng, m.worring}@uva.nl

https://github.com/zhengyijia/LatentRAG

Abstract

Single-step retrieval-augmented generation (RAG) provides an efficient way to incorporate external information for simple question answering tasks but struggles with complex questions. Agentic RAG extends this paradigm by replacing single-step retrieval with a multi-step process, in which the large language model (LLM) acts as a search agent that generates intermediate thoughts and subqueries to iteratively interact with the retrieval system. This iterative process incurs substantial latency due to the autoregressive generation of lengthy thoughts and subqueries. To address this limitation, we propose **LatentRAG**, a novel framework that shifts both reasoning and retrieval from discrete language space to continuous latent space. Unlike existing explicit methods that generate natural language thoughts or subqueries token-by-token, LatentRAG produces latent tokens for thoughts and subqueries directly from the hidden states in a single forward pass. We align LLMs with dense retrieval models in the latent space, enabling retrieval over latent subquery tokens and supporting end-to-end joint optimization. To improve transparency and encourage semantically meaningful latent representations, we incorporate a parallel latent decoding mechanism that translates latent tokens back into natural language. Extensive experiments on seven benchmark datasets show that LatentRAG achieves performance comparable to explicit agentic RAG methods while reducing inference latency by approximately **90%**, substantially narrowing the latency gap with traditional single-step RAG.

1 Introduction

Large language models (LLMs) have demonstrated strong capabilities in answering complex questions [1–3], but these capabilities are fundamentally bounded by their static internal knowledge [4, 5]. Solely relying on internal knowledge limits their performance on questions that require up-to-date information or proprietary knowledge [6, 7] and increases the risk of hallucinations [8, 9]. To improve both the factuality and transparency of LLM-generated outputs, retrieval-augmented generation (RAG) [10, 11] retrieves question-relevant information from an external retrieval system to augment the LLM inputs [12, 13]. Traditional RAG methods provide an efficient way to access external knowledge, but their single-step retrieval design limits their effectiveness on complex questions that require iterative reasoning and retrieval [14, 15].

Motivated by the success of tool-using LLM agents [16, 17], recent agentic RAG approaches [18, 19] replace traditional single-step retrieval with a multi-step agentic search process that alternates between generation and retrieval. In this process, the LLM acts as a search agent and iteratively decides what to retrieve. At each iteration, the agent generates a thought via chain-of-thought (CoT) reasoning [20] and then produces the next action, which can be either a subquery for the next retrieval step or the final answer. Each generated subquery is used to retrieve relevant documents. Unlike static single-step

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

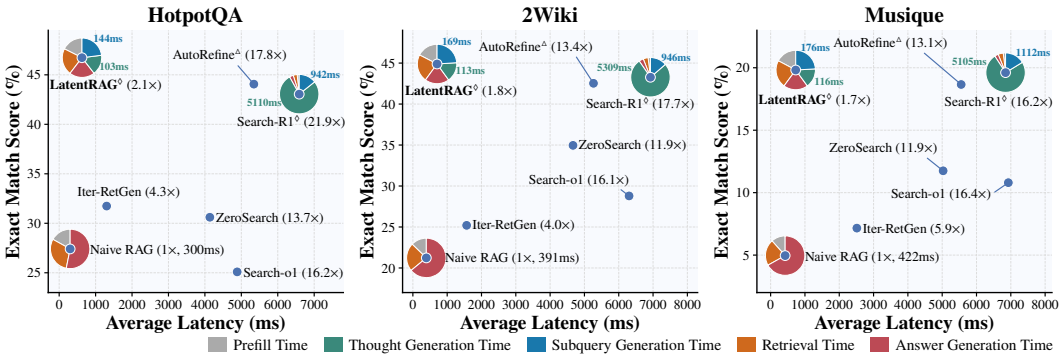


Figure 1: **Performance and latency comparison across multi-hop QA datasets.** The parenthesized numbers indicate the latency ratio relative to Naive RAG. LatentRAG achieves performance comparable to competitive agentic RAG methods such as Search-R1 and AutoRefine, while maintaining latency on par with naive single-step RAG. Search-R1 incurs excessive latency from thought and subquery generation, whereas LatentRAG substantially reduces the time spent in these two stages, leading to the efficiency gains. Detailed stage-wise latency breakdowns are provided in Appendix E.3.

retrieval in traditional RAG, this multi-step agentic search process enables complex questions to be decomposed and effectively solved step by step [21, 22]. Although agentic RAG methods demonstrate strong performance on tasks with complex questions [15, 23], they incur substantial latency due to the additional multi-step interactions [24, 25].

To identify the latency bottlenecks of agentic RAG, we measure the average inference time across different stages for both naive single-step RAG and agentic RAG methods. As shown in Fig. 1, on multi-hop question answering (QA) datasets, the total inference time of a representative agentic RAG method, Search-R1 [19], requires 16–22$\times$ the inference time of naive RAG. This overhead is primarily driven by the thought and subquery generation stages, which together account for approximately 90% of the total latency. Both stages involve autoregressive token-by-token generation of long outputs, where each output token depends on previously generated tokens, leading to multiple sequential LLM forward passes with limited parallelism. In contrast, prefill, retrieval, and final answer generation take far less time than the other two stages. The inference time comparison indicates that the latency bottlenecks of agentic RAG lie in the thought and subquery generation stages.

To reduce the thought and subquery generation latency in agentic RAG, we draw inspiration from another technique called latent reasoning. Latent reasoning [26, 27] is an efficient reasoning paradigm that performs reasoning within the continuous hidden states of the LLM, also referred to as latent tokens, without explicitly generating discrete language tokens. Compared to explicit reasoning, latent reasoning avoids allocating computation to non-semantic tokens that are produced solely for linguistic fluency [28, 26]. Furthermore, continuous latent tokens allow the LLM to directly generate high-level semantic representation, avoiding the inefficiency of explicit token-by-token generation and thereby enabling more parallelizable computation [29–31]. Although latent reasoning offers a promising avenue for enhancing reasoning efficiency, its application to agentic RAG remains unexplored.

In this work, we pioneer the integration of latent reasoning into the agentic RAG paradigm and, more importantly, propose a latent retrieval mechanism. Unlike generation-only tasks studied in prior work on latent reasoning [26, 32], agentic RAG requires the LLM to emit explicit subquery tokens to invoke external retrieval. This explicit token generation not only incurs significant decoding overhead but also prevents gradient propagation, thereby hindering direct optimization of the LLM using retrieval signals. To overcome these limitations, we investigate whether latent tokens generated by an LLM can effectively serve as subqueries for retrieval. This introduces **two challenges**. **(1) Data scarcity:** Training retrieval models typically requires large-scale paired data, often comprising hundreds of millions of query–document pairs [33, 34]. In contrast, agentic RAG systems are commonly developed under a training setup that provides only tens of thousands of question–answering pairs, without explicit supervision on the ground-truth documents for intermediate subqueries [19, 35]. This data scarcity makes it difficult to learn effective retrieval capability using conventional training paradigms for retrieval models. **(2) Transparency:** Latent tokens inherently obscure the intermediate thoughts and subqueries, which is particularly problematic for agentic RAG, as lengthy and redundant

retrieved documents make answer verification and evidence attribution [36, 37] time-consuming without explicit intermediate steps.

To address the aforementioned challenges, we introduce **LatentRAG**, an efficient agentic RAG framework that conducts reasoning and retrieval in the latent space. Specifically, we feed a sequence of special thought and subquery tokens into the LLM and use the corresponding last hidden states as latent thought and subquery tokens, respectively. These latent tokens are obtained in a single forward pass, enabling parallel computation and avoiding the inefficiency of autoregressive generation. To address challenge (1), we align the LLM with a pretrained dense retrieval model in the latent space. The latent subquery tokens are used as inputs to the retrieval model to generate latent subquery embeddings. We then minimize the KL divergence between the similarity distribution over documents induced by latent subquery embeddings and that induced by natural language subquery embeddings. This design enables fully differentiable end-to-end joint optimization of the LLM and the retrieval model. To address challenge (2) and encourage the latent tokens to capture meaningful semantics, we incorporate a parallel latent decoding mechanism that converts latent tokens into natural language thoughts and subqueries. During inference, this latent decoding process is optional, enabling a trade-off between transparency and efficiency. Since this latent decoding process depends only on the latent tokens, all thoughts and subqueries across different steps can be decoded in parallel, reducing the latency of the decoding process. Our main contributions are summarized as follows:

- We introduce LatentRAG, a novel agentic RAG framework that performs reasoning and retrieval in the latent space, reducing the latency overhead of explicit thought and subquery generation.
- We propose a latent-space alignment objective that jointly optimizes the LLM and retrieval model, enabling latent tokens to serve as effective retrieval queries while supporting end-to-end training.
- We incorporate a parallel decoding mechanism that translates latent tokens into explicit thoughts and subqueries, improving transparency while remaining more efficient than explicit agentic RAG.

Extensive experiments on seven benchmark datasets show that LatentRAG achieves performance comparable to explicit agentic RAG methods, with relative performance differences of less than 5%, while significantly reducing latency overhead by approximately **90%** on average, approaching the latency of traditional single-step RAG.

2 Related Work

Agentic RAG. Recent advances in RAG have shifted beyond traditional single-step methods [10, 11] toward agentic RAG approaches [23, 21, 15], which perform multi-step retrieval by iteratively generating intermediate thoughts and subqueries. Early agentic RAG methods [14, 38, 39, 18] primarily rely on prompting strategies to enable LLMs to interact with retrieval systems. To improve the retrieval ability of LLMs, Self-RAG [40] and AutoRAG [41] construct synthetic training data from RAG benchmark datasets for supervised finetuning (SFT). Some methods [24, 42, 43] further introduce mechanisms to balance internal knowledge and external retrieval, enabling LLMs to retrieve only when internal knowledge is insufficient. To mitigate the reliance on supervised training data and promote more flexible search strategies, a growing line of work [19, 44–46] formulates agentic RAG as a Markov decision process, where LLMs learn an optimal decision policy to interact with the retrieval system via reinforcement learning (RL). Recent RL-based approaches further incorporate fine-grained intermediate reward functions [47–50] and explore parallel retrieval strategies [51, 25, 52]. As discussed in the introduction, all these existing methods require generating long sequences of thoughts and subqueries in the language space, leading to substantial latency. In contrast to existing approaches, we explore performing reasoning and retrieval in the latent space, avoiding long textual thought and subquery generation and achieving significant efficiency gains.

Latent Reasoning. Latent reasoning [29, 53] reduces the latency overhead of explicit chain-of-thought (CoT) reasoning [20] by operating in the continuous hidden states of LLMs, but existing work primarily focuses on generation-only tasks [32, 26] without external retrieval. Early research explores adding filler tokens to enable LLMs to allocate more computation within the hidden states before generating outputs [32, 54]. Coconut [26] proposes an autoregressive latent reasoning paradigm, where each latent token, *i.e.*, a generated hidden state, is recursively fed back into the LLM to generate the next latent token. While the training process of Coconut is supervised only by the final answer, subsequent methods [55, 28, 56, 57] further leverage information generated by explicit CoT

as intermediate supervision to improve the training process. To enhance semantic consistency and address the distributional mismatch between the latent token space and the model input space, recent approaches [58–60] constrain latent representations as mixtures of the language token embeddings. Some methods [61, 62] introduce lightweight assistant models to generate latent tokens, thereby improving efficiency while avoiding disruption to the capabilities of the base LLM. Latent reasoning has been extended to practical applications, including retrieval. CLaRa [63] leverages latent reasoning to compress retrieved information in single-step RAG, while a concurrent work, LaSER [64], develops a dense retrieval model based on latent reasoning. Despite the rapid advancement of latent reasoning, its application to agentic RAG introduces several challenges as discussed in the introduction, leaving this area largely unexplored. In this paper, we pioneer the integration of latent reasoning into agentic RAG and further propose a latent retrieval mechanism, significantly reducing latency overhead.

3 Preliminaries

Following the standard setting in prior RAG research [19, 25], we study the question-answering (QA) task defined as follows. Given a question q, the objective is to generate an answer a by retrieving the necessary information from a large corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, where each d_i represents a document. To simplify notation, for each symbol that represents natural language text (e.g., d_i), we use the same symbol to denote its token sequence.

LLMs are widely used for solving the QA task. An LLM maps an input token sequence to an output sequence through two stages: *prefill* and *decoding*. In the prefill stage, all input tokens are processed in parallel to compute the key-value (KV) cache. In the decoding stage, output tokens are generated autoregressively, where each token is produced based on the KV cache of the input tokens and previously generated tokens. Due to autoregressive dependencies, the decoding stage can only generate output tokens in a token-by-token manner, leading to substantial latency for long outputs.

RAG systems augment LLMs with information retrieved from an external retrieval system. Two types of retrieval models are widely adopted [15, 65]: sparse retrieval models, which rely on exact token-level matches, and dense retrieval models, which encode the query and documents into continuous embeddings and select top-k documents based on cosine similarity. Dense retrieval models capture deeper semantic similarity than sparse retrieval models, leading to superior performance on RAG benchmarks [66, 67]. Thus, in this work, we focus on dense retrieval models.

Agentic RAG methods perform multi-step generation and retrieval, as shown in Fig. 2(a). At each iteration, the LLM generates a reasoning thought and a corresponding subquery, which is then used to retrieve relevant information from an external retrieval system. Formally, at iteration t, the historical *interaction trajectory* is denoted as a sequence:

$$\mathcal{I}_t = (\tau_0, s_0, c_0, \ldots, \tau_{t-1}, s_{t-1}, c_{t-1}), \tag{1}$$

where τ_i represents the i-th thought, s_i is the i-th generated subquery, and c_i comprises the contents of the top-k documents retrieved using s_i. Conditioned on the question q and the interaction trajectory $\mathcal{I}_t$, the agent first performs reasoning by producing the next thought τ_t and subsequently generates the next action output o_t, denoted jointly as $(\tau_t, o_t) = g_{\text{LLM}}(q, \mathcal{I}_t; \theta_{\text{LLM}})$, where θ_{LLM} represents the parameters of the LLM. The action output o_t is either a new subquery s_t for further retrieval or a final answer a, distinguished according to the output text format specified by the particular method.

4 Methodology

LatentRAG adopts a similar procedure to traditional explicit agentic RAG described in Sec. 3, where the LLM agent iteratively generates thoughts and subqueries, and the subqueries are then used to retrieve relevant information. Unlike explicit agentic RAG methods that generate thoughts and subqueries in the language space, LatentRAG operates in the latent space and only produces latent tokens, *i.e.*, the last-layer hidden states, for thoughts and subqueries (Sec. 4.1). The latent subquery tokens are then used as inputs to retrieve relevant documents (Sec. 4.2). To improve transparency, the latent thought and subquery tokens can be decoded into natural language via the latent decoding process (Sec. 4.3). The model is trained with a joint objective that combines losses from different components (Sec. 4.4). The overall framework is shown in Fig. 2.

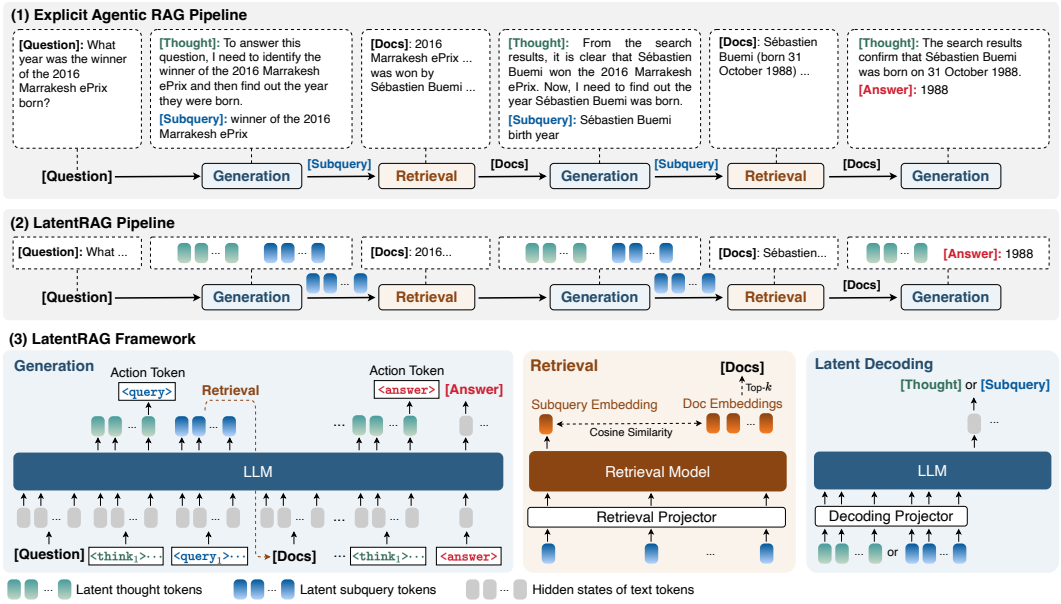


Figure 2: (1) Traditional explicit agentic RAG methods alternate between generation and retrieval, producing natural language thoughts and subqueries at each generation step to iteratively retrieve relevant documents. (2) LatentRAG only produces latent thought and subquery tokens at each generation step, and the latent subquery tokens are used for retrieval. (3) LatentRAG contains three components: **Generation** (Sec. 4.1), **Retrieval** (Sec. 4.2), and **Latent Decoding** (Sec. 4.3).

4.1 Generation with Latent Tokens

We replace the explicit thoughts τ_t and subqueries s_t in Eq. 1 with sequences of special tokens τ_t^ℓ and s_t^ℓ, respectively. Here $\tau_t^\ell = (\texttt{<think}_1\texttt{>}, \ldots, \texttt{<think}_m\texttt{>})$ denotes a sequence of m special thought tokens, and $s_t^\ell = (\texttt{<query}_1\texttt{>}, \ldots, \texttt{<query}_n\texttt{>})$ denotes a sequence of n special subquery tokens. At iteration t, the interaction trajectory is denoted as a sequence:

$$\mathcal{I}_t^\ell = (\tau_0^\ell, s_0^\ell, c_0, \ldots, \tau_{t-1}^\ell, s_{t-1}^\ell, c_{t-1}). \tag{2}$$

The special tokens serve as latent computation slots, allowing the LLM to allocate additional internal computation without generating explicit natural language thoughts and subqueries. During the prefill stage, the special tokens are processed in parallel, producing their last hidden states H_t^τ and H_t^s, which are referred to as latent thought and subquery tokens, respectively.

Given the question q and the interaction trajectory $\mathcal{I}_t^\ell$, we append the input with the special thought tokens τ_t^ℓ and let the LLM decode an action token from the last latent thought token:

$$\alpha_t = g_{\text{LLM}}(q, \mathcal{I}_t^\ell, \tau_t^\ell; \theta_{\text{LLM}}), \tag{3}$$

where $\alpha_t \in \{\texttt{<query>}, \texttt{<answer>}\}$ represents whether to proceed with retrieval by generating a subquery or to terminate by producing the final answer. If $\alpha_t = \texttt{<query>}$, we append the special tokens s_t^ℓ to the input sequence $(q, \mathcal{I}_t^\ell, \tau_t^\ell)$ and let the LLM generate the latent subquery tokens:

$$H_t^s = f_{\text{LLM}}(s_t^\ell; q, \mathcal{I}_t^\ell, \tau_t^\ell, \theta_{\text{LLM}}), \tag{4}$$

where f_{LLM} denotes a single forward pass of the LLM. The obtained latent subquery tokens H_t^s are used to retrieve relevant top-k documents, which constitute the retrieved content c_t (described in Sec. 4.2). If $\alpha_t = \texttt{<answer>}$, the model continues to generate the final answer a.

LatentRAG is trained via supervised finetuning using interaction trajectories constructed from existing explicit agentic RAG methods. Specifically, we replace each natural language thought τ_t and subquery s_t with the corresponding special token sequences τ_t^ℓ and s_t^ℓ. The token sequences within these trajectories are formatted according to a predefined prompt template (see Appendix C) and then provided as input to the LLM. The LLM is optimized to generate the correct action token α_t and the final answer a using the standard cross-entropy loss, denoted as $\mathcal{L}_{\text{gen}}$.

4.2 Latent Retrieval

We use the generated latent subquery tokens H_t^s to retrieve relevant content c_t. Since these latent tokens reside in the output space of the LLM and are not directly compatible with the input space of the retrieval model, we add a lightweight projector module $\mathrm{Proj}_{\mathrm{ret}}$ to bridge the two spaces. The projector is composed of a bidirectional self-attention layer and a position-wise feed-forward network (FFN) layer. The projected latent subquery tokens are fed into a trainable retrieval model to obtain the latent subquery embedding:

$$\boldsymbol{v}_{s_t^\ell} = f_{\mathrm{ret}}(\mathrm{Proj}_{\mathrm{ret}}(H_t^s); \theta_{\mathrm{ret}}). \tag{5}$$

Here θ_{ret} denotes the parameters of the retrieval model, which are initialized from a pretrained model and will be optimized during the finetuning process. Since ground truth documents are not available in our setting, we train the model to produce latent subquery embeddings that approximate the retrieval behavior induced by the corresponding natural language subqueries. Specifically, each natural language subquery s_t in the trajectory is encoded using a reference retrieval model to produce a reference embedding $\boldsymbol{v}'_{s_t}$:

$$\boldsymbol{v}'_{s_t} = f_{\mathrm{ret}}(s_t; \theta'_{\mathrm{ret}}), \tag{6}$$

where the reference retrieval model is initialized from the same pretrained model as the trainable one, but its parameters θ'_{ret} remain frozen during the finetuning process. The reference embeddings are used to retrieve top-k documents from the corpus, which are treated as pseudo-relevant documents.

To learn subquery embeddings that align with relevant documents, a common practice is to use the InfoNCE loss [68], which pulls query embeddings closer to positive documents while pushing them away from negatives. However, in our setting, pseudo-relevant documents are not ground-truth annotations and may contain substantial noise. In addition, unlike large-scale dense retrieval pretraining settings that rely on hundreds of millions of labeled query–document pairs [33, 34], agentic RAG is typically trained with only tens of thousands of samples [19, 35]. Such data noise and scarcity make the standard InfoNCE objective less well-suited to the agentic RAG setting.

To better leverage the prior knowledge encoded in the pretrained retrieval models, we introduce a retrieval objective based on Kullback–Leibler (KL) divergence. Specifically, for each subquery s_t and each candidate document d_i, we compute the following cosine similarity-based probabilities using the reference subquery embedding and the corresponding latent subquery embedding:

$$p_i(s_t) = \frac{\exp(\cos(\boldsymbol{v}'_{s_t}, \boldsymbol{v}_{d_i})/\beta)}{\sum_{j=1}^{N_d} \exp(\cos(\boldsymbol{v}'_{s_t}, \boldsymbol{v}_{d_j})/\beta)}, \qquad q_i(s_t) = \frac{\exp(\cos(\boldsymbol{v}_{s_t^\ell}, \boldsymbol{v}_{d_i})/\beta)}{\sum_{j=1}^{N_d} \exp(\cos(\boldsymbol{v}_{s_t^\ell}, \boldsymbol{v}_{d_j})/\beta)}, \tag{7}$$

where β is the temperature parameter that controls the sharpness of the distribution. N_d is the number of candidate documents, where the candidate set consists of all in-batch pseudo-relevant documents. The retrieval loss function is defined as the KL divergence between both distributions:

$$\mathcal{L}_{\mathrm{ret}} = \frac{1}{\|\mathcal{B}_s\|} \sum_{s_t \in \mathcal{B}_s} \sum_{i=1}^{N_d} p_i(s_t) \log \frac{p_i(s_t)}{q_i(s_t)}, \tag{8}$$

where $\mathcal{B}_s$ denotes all the subqueries in a training batch. An alternative objective is to directly align $\boldsymbol{v}_{s_t^\ell}$ and $\boldsymbol{v}'_{s_t}$ by minimizing cosine distance. However, our ablation experiments show that it yields lower performance compared to the KL objective.

4.3 Latent Decoding

To improve transparency of the decision-making process and enhance latent representation learning, we introduce a latent decoding objective. The key idea is to optimize the LLM to reconstruct the corresponding natural language sequences directly from the generated latent tokens.

We add projector modules Proj_τ and Proj_s to map latent thought and subquery tokens into the LLM input space, respectively. The projector modules follow the same structure as the projector introduced in Sec 4.2. The projected latent thought tokens or latent subquery tokens are then fed into the LLM to decode the corresponding natural language thought τ_t or subquery s_t:

$$\tau_t = g_{\mathrm{LLM}}(\mathrm{Proj}_\tau(H_t^\tau); \theta_{\mathrm{LLM}}), \qquad s_t = g_{\mathrm{LLM}}(\mathrm{Proj}_s(H_t^s); \theta_{\mathrm{LLM}}). \tag{9}$$

Table 1: **Overall results with different retrieval models.** **Green** and **Red** values respectively indicate relative percentage improvement and degradation compared to the corresponding baseline. Green and yellow shaded cells indicate the best and second best across all settings. Results show that our method achieves substantial efficiency gains with limited or even no performance degradation.

Methods	NQ		TriviaQA		PopQA		HotpotQA		2wiki		Musique		Bamboogle		Average	
	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓	EM(%)↑	Lat.(ms)↓
Direct Infer	14.76	257	42.48	158	14.50	175	16.68	179	21.89	191	3.19	239	11.20	200	17.81	200
Qwen3-Embedding-0.6B																
Naive RAG	25.84	452	53.32	334	35.72	297	27.41	300	21.23	391	4.96	422	11.20	314	25.67	359
Iter-RetGen	31.33	1,884	56.91	1,241	40.29	1,297	31.74	1,303	25.20	1,567	7.16	2,513	19.20	1,461	30.26	1,609
Search-o1	22.33	4,178	44.53	4,102	31.25	3,422	25.09	4,886	28.79	6,301	10.80	6,928	29.60	4,741	27.48	4,937
R1-Searcher	36.34	7,569	56.10	7,894	38.37	7,426	43.34	9,306	46.99	8,861	17.83	10,063	34.40	7,491	39.05	8,373
ZeroSearch	34.07	4,194	54.67	3,866	40.72	3,484	30.61	4,136	34.96	4,667	11.75	5,027	32.80	4,166	34.23	4,220
DeepRAG	33.16	2,847	55.48	2,860	39.15	2,873	33.44	3,232	43.92	3,758	13.20	4,149	32.00	2,848	35.76	3,224
Search-R1$^{\diamond}$	43.93	3,553	**61.70**	5,198	**47.31**	3,588	43.05	6,589	43.27	6,925	19.61	6,846	38.40	4,904	42.47	5,372
LatentRAG$^{\diamond}$	**46.29**	491(**-86.2%**)	59.85	478(**-90.8%**)	46.63	501(**-86.0%**)	**46.73**	626(**-90.5%**)	**44.89**	704(**-89.8%**)	**19.82**	730(**-89.3%**)	**40.00**	623(**-87.3%**)	**43.46**(**+2.3%**)	**593**(**-89.0%**)
AutoRefine$^{\triangle}$	44.35	4,782	**63.09**	4,223	47.19	4,397	44.06	5,344	42.53	5,264	18.66	5,553	**39.20**	4,224	42.73	4,827
LatentRAG$^{\triangle}$	**45.73**	409(**-91.4%**)	60.18	400(**-90.5%**)	**47.42**	422(**-90.4%**)	**47.16**	528(**-90.1%**)	**46.14**	607(**-88.5%**)	**20.73**	639(**-88.5%**)	**39.20**	581(**-86.2%**)	**43.79**(**+2.5%**)	**512**(**-89.4%**)
e5-base-v2																
Search-R1$^{\diamond}$	48.03	3,301	**64.59**	5,409	**46.02**	3,403	44.71	6,263	**42.13**	6,662	**20.19**	6,416	**43.20**	4,709	**44.12**	5,166
LatentRAG$^{\diamond}$	**49.03**	453(**-86.3%**)	61.30	410(**-92.4%**)	43.04	470(**-86.2%**)	**46.41**	588(**-90.6%**)	38.61	687(**-89.7%**)	18.87	656(**-89.8%**)	39.20	558(**-88.2%**)	42.35(**-4.0%**)	**546**(**-89.4%**)
AutoRefine$^{\triangle}$	48.20	4,303	**65.30**	3,853	**47.12**	4,135	45.13	5,133	41.90	4,695	**21.80**	5,583	**48.00**	4,045	**45.35**	4,535
LatentRAG$^{\triangle}$	**49.86**	369(**-91.4%**)	62.32	370(**-90.4%**)	43.88	407(**-90.2%**)	**46.27**	464(**-91.0%**)	**41.98**	552(**-88.2%**)	20.94	548(**-90.2%**)	40.80	519(**-87.2%**)	43.72(**-3.6%**)	**462**(**-89.8%**)
jina-embeddings-v5-text-nano																
Search-R1$^{\diamond}$	45.79	3,381	**63.16**	5,120	46.54	3,639	44.31	6,161	42.95	6,758	20.40	6,177	**44.80**	4,579	43.99	5,116
LatentRAG$^{\diamond}$	**47.37**	456(**-86.5%**)	61.25	394(**-92.3%**)	**46.72**	426(**-88.3%**)	**47.66**	540(**-91.2%**)	**45.06**	641(**-90.5%**)	**22.26**	632(**-89.8%**)	43.20	592(**-87.1%**)	**44.79**(**+1.8%**)	**526**(**-89.7%**)
AutoRefine$^{\triangle}$	45.98	4,730	**64.73**	3,903	46.84	4,109	44.79	4,962	42.95	4,879	20.52	5,614	**43.20**	4,053	44.14	4,607
LatentRAG$^{\triangle}$	**47.59**	368(**-92.2%**)	61.54	365(**-90.7%**)	**47.82**	383(**-90.7%**)	**48.10**	467(**-90.6%**)	**45.24**	539(**-89.0%**)	**22.92**	540(**-90.4%**)	40.80	514(**-87.3%**)	**44.86**(**+1.6%**)	**454**(**-90.1%**)
harrier-oss-v1-270m																
Search-R1$^{\diamond}$	44.40	3,510	**62.80**	4,983	**47.14**	3,351	44.88	6,270	44.16	6,864	18.62	6,613	**40.00**	4,944	**43.14**	5,219
LatentRAG$^{\diamond}$	**46.15**	485(**-86.2%**)	60.40	444(**-91.1%**)	45.41	497(**-85.2%**)	**47.27**	614(**-90.2%**)	**44.99**	698(**-89.8%**)	**19.90**	708(**-89.3%**)	34.40	636(**-87.1%**)	42.65(**-1.1%**)	**583**(**-88.8%**)
AutoRefine$^{\triangle}$	43.80	4,457	**64.32**	3,899	**47.68**	4,153	45.40	4,703	43.31	4,910	20.15	5,511	**43.20**	4,360	**43.98**	4,570
LatentRAG$^{\triangle}$	**45.68**	389(**-91.3%**)	60.98	391(**-90.0%**)	45.92	409(**-90.2%**)	**46.74**	507(**-89.2%**)	**44.58**	602(**-87.7%**)	**20.56**	579(**-89.5%**)	35.20	559(**-87.2%**)	42.81(**-2.7%**)	**491**(**-89.3%**)
F2LLM-v2-330M																
Search-R1$^{\diamond}$	43.35	3,717	**61.04**	5,394	**44.92**	3,620	41.99	6,228	41.53	6,765	**18.54**	6,690	**36.00**	5,183	41.05	5,371
LatentRAG$^{\diamond}$	**45.54**	484(**-87.0%**)	58.56	432(**-92.0%**)	44.21	448(**-87.6%**)	**44.58**	593(**-90.5%**)	**43.02**	679(**-90.0%**)	18.37	667(**-90.0%**)	**36.00**	602(**-88.4%**)	**41.47**(**+1.0%**)	**558**(**-89.6%**)
AutoRefine$^{\triangle}$	43.82	4,483	**62.53**	4,204	**45.17**	4,271	42.63	4,992	41.58	5,101	19.03	5,412	**40.80**	4,381	**42.22**	4,692
LatentRAG$^{\triangle}$	**45.32**	403(**-91.0%**)	58.83	386(**-90.8%**)	44.72	417(**-90.2%**)	**44.74**	499(**-90.0%**)	**44.08**	583(**-88.6%**)	**19.82**	566(**-89.5%**)	36.80	552(**-87.4%**)	42.04(**-0.4%**)	**487**(**-89.6%**)

The prompts used to format these inputs are provided in Appendix C. The decoding process is optimized using the standard cross-entropy loss between the generated sequence and the corresponding natural language target. This results in two decoding losses: a thought decoding loss $\mathcal{L}_{\text{dec}}^{\tau}$ and a subquery decoding loss $\mathcal{L}_{\text{dec}}^{s}$. The latent decoding loss is the combination of both terms:

$$\mathcal{L}_{\text{dec}} = \mathcal{L}_{\text{dec}}^{\tau} + \mathcal{L}_{\text{dec}}^{s}. \tag{10}$$

During inference, this latent decoding process is optional, allowing the LLM agent to perform reasoning and retrieval entirely in the latent space for efficiency. When required, latent tokens can be decoded into natural language for transparency. Since each decoding process depends only on its corresponding latent tokens, all thoughts and subqueries across multiple steps can be decoded in parallel, thus reducing the latency of generating these natural language sequences.

4.4 Overall Training Objective

The overall training objective is defined as a weighted combination of the generation loss, retrieval loss, and latent decoding loss:

$$\mathcal{L} = \mathcal{L}_{\text{gen}} + \lambda_{\text{ret}}\mathcal{L}_{\text{ret}} + \mathcal{L}_{\text{dec}}, \tag{11}$$

where λ_{ret} controls the relative scale of the retrieval loss. We do not introduce additional scaling factors for $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{dec}}$ since both are derived from the standard LLM cross-entropy objective and thus have comparable magnitudes.

5 Experiments

In this section, we conduct experiments to evaluate LatentRAG in terms of effectiveness, efficiency, scalability, and adaptability. Additional experiments and analyses are provided in Appendix E.

5.1 Experimental Setup

Datasets. We evaluate LatentRAG using seven common benchmark QA datasets, comprising three general QA datasets (NQ [69], TriviaQA [70], and PopQA [71]) and four multi-hop QA datasets

(HotpotQA [72], 2wiki [73], Musique [74], and Bamboogle [75]). We use the 2018 Wikipedia dump [76] as the corpus for retrieval. More details of the datasets can be found in Appendix D.

Baselines. We compare LatentRAG against a diverse set of baselines covering direct inference (Direct Infer), traditional single-step RAG (Naive RAG [10]), prompt-based agentic RAG (Iter-RetGen [77], Search-o1 [18]), and training-based agentic RAG (R1-Searcher [45], ZeroSearch [78], DeepRAG [24], Search-R1 [19], AutoRefine [35]).

Implementation details. Following previous works [19, 35], we adopt Qwen2.5-7B [79] as the default LLM for all methods. For training-based baselines, we utilize their released model weights to ensure the faithful reproduction of their reported performance. Training trajectories are constructed from a combined training set of NQ and HotpotQA using Search-R1 and AutoRefine. Variants trained on trajectories generated by Search-R1 and AutoRefine are denoted as **LatentRAG**$^{\diamondsuit}$ and **LatentRAG**$^{\triangle}$, respectively. To reduce computational costs, we conduct main experiments using lightweight retrieval models with fewer than 1B parameters, which are among the top-performing models on the MTEB leaderboard [80] and cover diverse model architectures, including Qwen3-Embedding-0.6B [33], e5-base-v2 [34], jina-embeddings-v5-text-nano [81], harrier-oss-v1-270m[1], and F2LLM-v2-330M [82]. Unless otherwise specified, we use Qwen3-Embedding-0.6B as the default retriever. To evaluate the trade-off between performance and latency, we report the exact match (EM) score [19] and the average latency per question. Latency is measured on a single NVIDIA H100 GPU with 94 GB memory by default. More implementation details are in Appendix B.

5.2 Main Results

Overall performance and latency. As shown in Table 1, advanced agentic RAG methods such as Search-R1 and AutoRefine achieve superior performance over naive single-step RAG, but incur substantially higher latency, with an average overhead of around 15× the latency in single-step RAG. This latency gap is more pronounced on multi-hop QA datasets. In contrast, LatentRAG trained on trajectories from Search-R1 and AutoRefine achieves comparable performance, with relative differences within 5%, while significantly reducing latency by approximately 90%. This advantage holds consistently across diverse retrieval models. Fig. 1 shows that LatentRAG significantly reduces latency in thought and subquery generation.

Compared to other retrieval models, we observe a relatively large performance drop when using e5-base-v2. To investigate the source of this discrepancy, we analyze the embedding spaces of different retrieval models. As shown in Fig. 4 in the Appendix, e5-base-v2 exhibits severe anisotropy [83, 84], indicating that the embeddings produced by the model are highly concentrated within a narrow cone on a hypersphere. This skewed distribution makes it difficult for the LLM to adapt to the retrieval space. More analysis is provided in Appendix E.1.

Latent decoding efficiency. Latent decoding is an option for improving transparency at the cost of additional latency. To quantify this overhead, Table 2 reports the latency of LatentRAG with and without latent decoding. Latent decoding increases the overall latency of LatentRAG by approximately 4–5×. Nevertheless, it still reduces latency by 63.3% and 47.4% compared to Search-R1 and AutoRefine, respectively. The efficiency gain stems from the removal of sequential dependencies, enabling parallel decoding across steps. The actual speedup is bounded by the longest sequence in the batch, which determines the number of decoding steps required. We report the *max length ratio* in Table 2, defined as the fraction of tokens in the longest thought or subquery sequence over the total decoding length. A higher ratio indicates a more imbalanced distribution of sequence lengths. In particular, LatentRAG$^{\triangle}$ exhibits a larger max length ratio, which explains its less pronounced efficiency gains. Further analysis is provided in Appendix E.2, along with case studies of decoded examples in Appendix E.10.

Table 2: **Latency with and without decoding.**

Methods	Lat. (ms)$^{\downarrow}$	Max Len. Ratio(%)
Search-R1$^{\diamondsuit}$	5,372	37.74
LatentRAG$^{\diamondsuit}$ w/o decoding	593(**-89.0%**)	–
LatentRAG$^{\diamondsuit}$ w/ decoding	1,970(**-63.3%**)	33.70
AutoRefine$^{\triangle}$	4,827	43.14
LatentRAG$^{\triangle}$ w/o decoding	512(**-89.4%**)	–
LatentRAG$^{\triangle}$ w/ decoding	2,540(**-47.4%**)	42.13

[1] https://huggingface.co/microsoft/harrier-oss-v1-270m

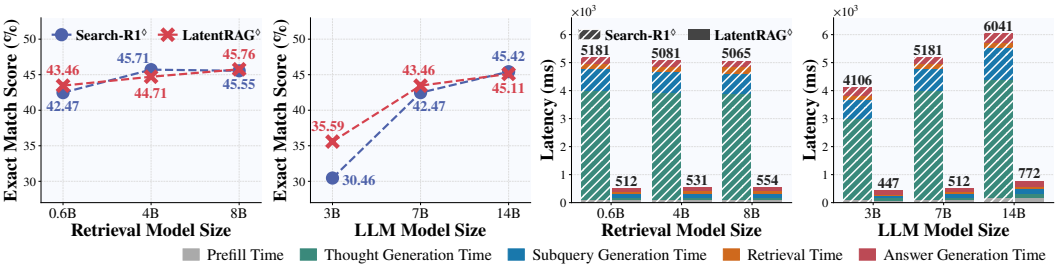


Figure 3: **Performance and latency results across different retrieval model and LLM sizes.**

Table 3: **Cross-retriever adaptation.** This table reports exact match scores (%). LatentRAG adapted from Qwen3 to Jina, finetuning only the retrieval projector, achieves performance comparable to that of the variant trained with the Jina retrieval model from scratch.

Methods	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
Search-R1$^{\diamond}$ (Jina)	45.79	**63.16**	46.54	44.31	42.95	20.40	**44.80**	43.99
LatentRAG$^{\diamond}$ (Jina)	47.37	61.25	46.72	**47.66**	**45.06**	**22.26**	43.20	**44.79** (+1.8%)
LatentRAG$^{\diamond}$ (Qwen3→Jina)	**47.62**	61.25	**46.89**	**47.66**	44.51	21.47	41.60	44.43 (+1.0%)

Scaling model size. We study scalability along two orthogonal dimensions. For retrieval model scaling, we evaluate Qwen3-Embedding-0.6B, 4B, and 8B [33] with a fixed 7B LLM. For LLM scaling, we evaluate Qwen2.5-3B, 7B, and 14B with a fixed Qwen3-Embedding-0.6B retrieval model. Larger retrieval models produce higher-dimensional embeddings, resulting in a substantially larger index that cannot fit on a single GPU. To ensure a fair comparison across different model sizes, we use three H100 GPUs for retrieval deployment and one for the LLM across all scaling experiments.

As shown in Fig. 3, performance improves with increasing model size along both dimensions. Scaling the retrieval model introduces negligible latency overhead, as the retrieval process can be efficiently parallelized. In contrast, scaling the LLM leads to substantial latency increases for SearchR1 due to increased decoding time. Our method achieves comparable performance across all settings and yields improvements in the 3B LLM setting while significantly reducing inference latency.

Cross-retriever adaptation. Since LatentRAG requires aligning the representation spaces of the LLM and retrieval model, switching to a different retrieval model requires adaptation. We investigate whether LatentRAG can be adapted to a new retrieval model without finetuning the LLM or the retrieval model. Specifically, we adapt LatentRAG trained with Qwen3-Embedding-0.6B (Qwen3) to jina-embeddings-v5-text-nano (Jina) by finetuning only the retrieval projector. As shown in Table 3, the adapted model achieves performance similar to that of LatentRAG trained from scratch with Jina. Both variants achieve a slight relative improvement over the Search-R1 baseline. These results demonstrate that finetuning only the retrieval projector is sufficient for cross-retriever adaptation.

We further investigate the data efficiency of cross-retriever adaptation. Specifically, we examine whether adaptation can be achieved with limited training data or even using only synthetic data. We first randomly sample 5,000 queries from the original training set to construct adaptation trajectories, corresponding to less than 10% of the original training data, and compare the adapted model with a model trained from scratch using the same amount of data. We then consider a more challenging setting where adaptation uses only 5,000 synthetic queries, with no real query data. The synthetic queries are generated by the base model Qwen2.5-7B from randomly sampled documents in the Wikipedia corpus. The prompt used for query generation is provided in Appendix C. Since each query is generated from a single document, these synthetic queries are generally simple and differ substantially from the complex multi-hop queries in the evaluation datasets.

As shown in Table 4, using only 5,000 training trajectories, LatentRAG trained from scratch exhibits substantially worse performance than the variant trained using the full data, suggesting that 5,000 training trajectories are insufficient to learn effective retrieval policies from scratch. In contrast, LatentRAG adapted from Qwen3 to Jina achieves performance comparable to that of the full-data variant, demonstrating that cross-retriever adaptation can be achieved with relatively limited training data compared to training from scratch. More importantly, even when using only 5,000 synthetic

Table 4: **Data efficiency of cross-retriever adaptation.** This table reports exact match scores (%). When using a limited amount of training data or even only synthetic data, LatentRAG adapted from Qwen3 to Jina achieves significantly better performance than the model trained from scratch.

Methods	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
LatentRAG$^{\diamond}$ (Jina, full data)	47.37	**61.25**	46.72	**47.66**	**45.06**	**22.26**	**43.20**	**44.79**
LatentRAG$^{\diamond}$ (Qwen3→Jina, full data)	**47.62**	**61.25**	**46.89**	**47.66**	44.51	21.47	41.60	44.43
LatentRAG$^{\diamond}$ (Jina, **5,000 real data**)	35.12	47.84	34.61	33.99	26.79	10.67	22.40	30.20
LatentRAG$^{\diamond}$ (Qwen3→Jina, **5,000 real data**)	**46.93**	**58.85**	**45.29**	**46.06**	**43.21**	**19.36**	**40.80**	**42.93**
LatentRAG$^{\diamond}$ (Jina, **5,000 synthetic data**)	25.18	40.13	30.07	20.65	15.93	3.56	7.20	20.39
LatentRAG$^{\diamond}$ (Qwen3→Jina, **5,000 synthetic data**)	**45.12**	**57.62**	**44.66**	**43.92**	**43.71**	**17.63**	**39.20**	**41.69**

queries, the adapted model substantially outperforms the model trained from scratch, demonstrating that cross-retriever adaptation remains effective even when real training data are unavailable.

5.3 Ablation Studies

We conduct ablation studies on key design choices to validate their effectiveness. Specifically, we replace the KL-based retrieval objective in Eq. 8 with two alternative choices: (i) a cosine loss, which directly minimizes the cosine distance between the latent subquery embedding $\boldsymbol{v}_{s_t^\ell}$ and the corresponding reference subquery embedding $\boldsymbol{v}'_{s_t}$, and (ii) a standard InfoNCE loss [68], which is widely used for training retrieval models. We further consider two ablation settings: (iii) removing the pretrained retrieval model and relying solely on the LLM to produce subquery embeddings, and (iv) removing the latent decoding loss in Eq. 10 during training. We report the average EM score as well as two retrieval related metrics: (a) retrieval success rate, defined as the proportion of successful iterative retrievals where the retrieved documents contain the ground truth answer, and (b) retrieval overlap, defined as the proportion of documents retrieved by Search-R1 that are also retrieved by our model.

Table 5: **Ablation study results.** Overlap indicates the proportion of documents retrieved by Search-R1 that are also retrieved by each model.

Methods	EM(%)$^{\uparrow}$	Success(%)$^{\uparrow}$	Overlap(%)
Search-R1$^{\diamond}$	42.47	60.15	100.00
LatentRAG$^{\diamond}$(w/ KL loss)	**43.46**	**61.27**	59.41
- (i) Cosine loss	42.55	60.76	68.31
- (ii) InfoNCE loss	41.86	58.60	47.08
- (iii) w/o retriever	41.85	59.07	50.92
- (iv) w/o decoding	40.61	60.64	57.38

As shown in Table 5, LatentRAG with the proposed KL-based objective achieves better EM score and retrieval success rate compared to the cosine and InfoNCE alternatives. The cosine loss yields the highest retrieval overlap ratio, indicating closer imitation of the teacher model Search-R1. However, its performance is lower than that of the KL-based variant, suggesting that overly aligning with the teacher model may limit model capacity and lead to suboptimal performance. Removing the pretrained retrieval model also degrades the performance, highlighting the importance of the inductive bias provided by the pretrained retrieval model. Finally, removing the latent decoding loss leads to performance degradation, suggesting that latent decoding not only improves transparency at inference time, but also facilitates the learning of latent representations during training.

6 Conclusion

In this paper, we propose LatentRAG, an efficient agentic RAG framework that shifts both reasoning and retrieval from discrete language space to continuous latent space. Experiments show that LatentRAG achieves performance comparable to existing agentic RAG methods while reducing latency by approximately 90%. To improve transparency, the latent tokens can be optionally decoded into natural language with additional latency overhead, while still achieving an overall 40–60% reduction in latency compared to the baselines. Experiments across different model scales demonstrate the general applicability of LatentRAG. Cross-retriever experiments further demonstrate the adaptability of LatentRAG to new retrieval models, recovering approximately 95% of the performance by finetuning only the retrieval projector with a small amount of training data or even only synthetic data. Additional experiments in Appendix E further demonstrate the effectiveness and efficiency of LatentRAG under different settings, including its superior performance on long-horizon datasets and its latency advantages over baseline methods with vLLM optimization and speculative decoding.

Acknowledgments and Disclosure of Funding

This work is supported by the AI4Intelligence project with file number KICH1.VE01.20.011, partly financed by the Dutch Research Council (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO). This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant number EINF-15515. The authors declare no competing interests.

References

[1] Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and Philip S Yu. Large language models in law: A survey. *AI Open*, 2024.

[2] Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, et al. A survey on large language models for mathematical reasoning. *ACM Comput. Surv.*, 2025.

[3] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. Toward expert-level medical question answering with large language models. *Nat. Med.*, 2025.

[4] Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Qipeng Guo, Xiangkun Hu, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Xuming Hu, Zehan Qi, Wenyang Gao, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. Survey on factuality in large language models. *ACM Comput. Surv.*, 2025.

[5] Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Nenkov Georgiev, Rocktim Jyoti Das, and Preslav Nakov. Factuality of large language models: A survey. In *EMNLP*, 2024.

[6] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 2024.

[7] Mingyang Wang, Alisa Stoll, Lukas Lange, Heike Adel, Hinrich Schütze, and Jannik Strötgen. Bring your own knowledge: A survey of methods for LLM knowledge expansion. *arXiv preprint arXiv:2502.12598*, 2025.

[8] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 2023.

[9] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 2025.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

[11] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *ICML*, 2020.

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[13] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *ACM Trans. Inf. Syst.*, 2025.

[14] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL*, 2023.

[15] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint arXiv:2501.09136*, 2025.

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[17] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

[18] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *EMNLP*, 2025.

[19] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *COLM*, 2025.

[20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.

[21] Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, et al. Towards agentic RAG with deep reasoning: A survey of RAG-reasoning systems in LLMs. In *Findings of EMNLP*, 2025.

[22] Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved LLM agents. *arXiv preprint arXiv:2505.15117*, 2025.

[23] Minhua Lin, Zongyu Wu, Zhichao Xu, Hui Liu, Xianfeng Tang, Qi He, Charu Aggarwal, Xiang Zhang, and Suhang Wang. A comprehensive survey on reinforcement learning-based agentic search: Foundations, roles, optimizations, evaluations, and applications. *arXiv preprint arXiv:2510.16724*, 2025.

[24] Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. DeepRAG: Thinking to retrieve step by step for large language models. In *ICLR*, 2026.

[25] Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. RAG-R1: Incentivizing the search and reasoning capabilities of LLMs through multi-query parallelism. In *AAAI*, 2026.

[26] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *COLM*, 2025.

[27] Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*, 2025.

[28] Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

[29] Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*, 2025.

[30] Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R Costa-jussà, David Dale, et al. Large concept models: Language modeling in a sentence representation space. *arXiv preprint arXiv:2412.08821*, 2024.

[31] Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. LLM pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*, 2025.

[32] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *ICLR*, 2024.

[33] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[34] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[35] Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. Search and refine during think: Facilitating knowledge refinement for improved retrieval-augmented reasoning. In *NeurIPS*, 2025.

[36] Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. Model internals-based answer attribution for trustworthy retrieval-augmented generation. In *EMNLP*, 2024.

[37] João Eduardo Batista, Emil Vatai, and Mohamed Wahib. SAFE: Improving LLM systems using sentence-level in-generation attribution. *arXiv preprint arXiv:2505.12621*, 2025.

[38] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *EMNLP*, 2023.

[39] Zhao Xinjie, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. ReAgent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop QA. In *EMNLP*, 2025.

[40] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*, 2024.

[41] Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouš Eibich. AutoRAG: Automated framework for optimization of retrieval augmented generation pipeline. *arXiv preprint arXiv:2410.20878*, 2024.

[42] Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. Unified active retrieval for retrieval augmented generation. In *Findings of EMNLP*, 2024.

[43] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *NAACL*, 2024.

[44] Mingyang Chen, Linzhuang Sun, Tianpeng Li, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, et al. ReSearch: Learning to reason with search for LLMs via reinforcement learning. In *NeurIPS*, 2025.

[45] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

[46] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *EMNLP*, 2025.

[47] Yutao Xie, Nathaniel Thomas, Nicklas Hansen, Yang Fu, Li Erran Li, and Xiaolong Wang. TIPS: Turn-level information-potential reward shaping for search-augmented LLMs. In *ICLR*, 2026.

[48] Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. HiPRAG: hierarchical process rewards for efficient agentic retrieval augmented generation. In *ICLR*, 2026.

[49] Fengji Zhang, Xinyao Niu, Chengyang Ying, Guancheng Lin, Zhongkai Hao, Zhou Fan, Chengen Huang, Jacky Keung, Bei Chen, and Junyang Lin. A^2Search: Ambiguity-aware question answering with reinforcement learning. In *ICLR*, 2026.

[50] Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. R-Search: Empowering LLM reasoning with search via multi-reward reinforcement learning. *arXiv preprint arXiv:2506.04185*, 2025.

[51] Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. ParallelSearch: Train your LLMs to decompose query and search sub-queries in parallel with reinforcement learning. *arXiv preprint arXiv:2508.09303*, 2025.

[52] Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026.

[53] Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026.

[54] Jacob Pfau, William Merrill, and Samuel R Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *COLM*, 2024.

[55] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *EMNLP*, 2025.

[56] Jianwei Wang, Ziming Wu, Fuming Lai, Shaobing Lian, and Ziqian Zeng. SynAdapt: Learning adaptive reasoning in large language models via synthetic continuous chain-of-thought. *arXiv preprint arXiv:2508.00574*, 2025.

[57] Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *ICLR*, 2026.

[58] Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *NeurIPS*, 2025.

[59] Yufa Zhou, Yixiao Wang, Xunjian Yin, Shuyan Zhou, and Anru R Zhang. The geometry of reasoning: Flowing logics in representation space. In *ICLR*, 2026.

[60] Jingcheng Deng, Liang Pang, Zihao Wei, Shichen Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Latent reasoning in LLMs as a vocabulary-space superposition. *arXiv preprint arXiv:2510.15522*, 2025.

[61] Yinhan He, Wendy Zheng, Yaochen Zhu, Zaiyi Zheng, Lin Su, Sriram Vasudevan, Qi Guo, Liangjie Hong, and Jundong Li. SemCoT: Accelerating chain-of-thought reasoning through semantically-aligned implicit tokens. In *NeurIPS*, 2025.

[62] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *ACL*, 2025.

[63] Jie He, Richard He Bai, Sinead Williamson, Jeff Z Pan, Navdeep Jaitly, and Yizhe Zhang. CLaRa: Bridging retrieval and generation with continuous latent reasoning. *arXiv preprint arXiv:2511.18659*, 2025.

[64] Jiajie Jin, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, Yutao Zhu, and Zhicheng Dou. LaSER: Internalizing explicit reasoning into latent space for dense retrieval. *arXiv preprint arXiv:2603.01425*, 2026.

[65] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In *KDD*, 2024.

[66] Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. In *WWW*, 2025.

[67] Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. CRUD-RAG: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Trans. Inf. Syst.*, 2025.

[68] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[69] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *TACL*, 2019.

[70] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

[71] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

[72] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

[73] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *COLING*, 2020.

[74] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *TACL*, 2022.

[75] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of EMNLP*, 2023.

[76] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[77] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of EMNLP*, 2023.

[78] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*, 2025.

[79] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[80] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *EACL*, 2023.

[81] Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation. *arXiv preprint arXiv:2602.15547*, 2026.

[82] Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. F2LLM-v2: Inclusive, performant, and efficient embeddings for a multilingual world. *arXiv preprint arXiv:2603.19223*, 2026.

[83] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *EMNLP*, 2020.

[84] Wenxuan Zhou, Bill Yuchen Lin, and Xiang Ren. IsoBN: Fine-tuning BERT with isotropic batch normalization. In *AAAI*, 2021.

[85] Ziyi Xia, Kun Luo, Hongjin Qian, Siqi Bao, and Zheng Liu. Open data synthesis for deep research. In *ICLR*, 2026.

[86] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

[87] Haotian Chen, Xin Cong, Shengda Fan, Yuyang Fu, Ziqin Gong, Yaxi Lu, Yishan Li, Boye Niu, Chengjun Pan, Zijun Song, et al. AgentCPM-Explore: Realizing long-horizon deep exploration for edge-scale agents. *arXiv preprint arXiv:2602.06485*, 2026.

[88] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[89] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

[90] nostalgebraist. Interpreting GPT: the logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020.

A Broader Impacts and Limitations

Broader impacts. This work proposes an efficient agentic RAG framework that performs both reasoning and retrieval in the latent space. The proposed approach could be applied to a wide range of information retrieval scenarios, such as legal or clinical question answering [1, 3], and improve overall efficiency in these tasks. More broadly, this work raises a fundamental question: **Should information retrieval be redesigned for AI agents?** Humans naturally use words to search for information, and traditional retrieval systems therefore typically rely on text queries. However, as retrieval systems are increasingly used by AI agents rather than directly by humans, it is worth reconsidering this text-centric paradigm. Both AI agents and retrieval systems operate on continuous latent representations, making it possible for AI agents to search without first translating their internal latent representations into text. This opens up new opportunities to rethink information retrieval in the era of agentic AI and develop retrieval systems better suited to AI agents.

Limitations and future work. Our method relies on supervised finetuning over trajectories generated by existing agentic RAG methods, and its performance is therefore partly bounded by the quality of the training data. This hinders the model from directly learning an optimal retrieval policy through interactions with the retrieval system. Nevertheless, our approach yields strong and efficient initial models that serve as an effective foundation for future research. Future work could investigate reinforcement learning to further improve the SFT-trained model by encouraging exploration and exploitation. Additionally, while LatentRAG takes a first step toward reducing agentic RAG latency through latent reasoning and retrieval, it has not yet been integrated with existing inference optimization frameworks. Future work could explore adapting frameworks such as vLLM to LatentRAG to enable efficient batched inference and improve overall throughput in practical deployments.

B Implementation Details

Training data construction. As described in the main paper, we combine the training sets from NQ and HotpotQA to construct a unified training dataset. We then build training trajectories using interaction data generated by Search-R1 and AutoRefine on this unified training dataset. Each trajectory consists of the question, intermediate reasoning thoughts, subqueries, retrieved document chunks, and the final generated answer. AutoRefine introduces an additional refinement stage to improve the initially retrieved documents. To maintain a consistent trajectory format with Search-R1, we merge the refinement text into the reasoning thoughts. We retain only those trajectories that produce a correct final answer for training. To facilitate finer-grained control over different components in the generated trajectories, we introduce a set of special tokens to explicitly mark structural elements in the output, such as `<Answer>...</Answer>`. In contrast, these tags are typically tokenized into multiple subword units in Search-R1 and AutoRefine. This difference may introduce minor variations in generation time. However, its impact is negligible compared to the overall latency reduction achieved by our framework.

Computing resources & parallelization strategies. For training, we optimize LatentRAG on a single compute node equipped with two NVIDIA H100 GPUs, each with 94 GB of memory. Each training job takes about 24 to 48 hours to complete. To reduce GPU memory consumption, we enable gradient checkpointing to minimize the storage of intermediate activations. For distributed training, we adopt DeepSpeed ZeRO1[2], which shards the optimizer states across GPU devices while keeping gradients and model parameters fully replicated. This design avoids additional communication overhead associated with parameter and gradient sharding, thereby maintaining efficient data-parallel training. To address the imbalance in trajectory lengths, we implement a binned batching strategy. Specifically, we partition trajectories into 200 bins according to their lengths and construct each batch by sampling from a single bin. This binned batching strategy ensures that samples within each batch have similar sequence lengths and therefore reduces padding overhead and improves computational efficiency. We use bfloat16 precision and FlashAttention-2[3] during training. We adopt LoRA[4] with a rank of 16 for parameter-efficient finetuning, which significantly reduces the number of trainable parameters, thereby lowering memory and computational costs.

[2] https://github.com/deepspeedai/DeepSpeed

[3] https://github.com/dao-ailab/flash-attention

[4] https://github.com/microsoft/LoRA

For evaluation, we conduct experiments on a single NVIDIA H100 GPU with 94 GB of memory by default. We deploy the retrieval system using Faiss[5] on the GPU with half-precision indexing and load the LLM on the same device. To ensure a fair comparison across different methods, we measure both LLM prefill and decoding latency using the standard forward pass implemented in Hugging Face Transformers[6]. For scaling experiments, larger retrieval models produce higher-dimensional document embeddings that exceed the memory capacity of a single GPU. For example, the index built from Qwen3-Embedding-8B occupies approximately 160 GB even with float16 precision. To accommodate this, for all the scaling experiments, we deploy the retrieval system across three H100 GPUs, while using a separate H100 GPU to serve the LLM. This setup ensures sufficient GPU resources for both retrieval and generation, allowing us to report latency under sufficient computational resources, where system bottlenecks are minimized.

Hyperparameters. We finetune the model using LoRA with rank 16 and scaling factor 64, applied to all projection weights. The model is trained for 5 epochs with a learning rate of 1×10^{-4}. The maximum trajectory length is capped at 3000 tokens. For the KL divergence loss, we set the target distribution based on the similarity scores between queries and documents. Specifically, we select the temperature factor that makes the cumulative probability of the top-3 retrieved documents approach 0.5. In practice, this corresponds to setting the temperature to $\beta = 0.03$ in most cases, while the detailed setting for each case can be found in our source code. The loss weight for the retrieval objective is set to $\lambda_{\text{ret}} = 1$. For the retrieval model, we remove dropout to reduce noise in the target distribution, while for the LLM we apply a dropout rate of 0.1. We use $m = 4$ thought tokens for each thought generation step and $n = 16$ subquery tokens for each subquery generation step. The training batch size is set to 16. The model is optimized using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. For retrieval loss calculation, we retrieve the top-16 documents as pseudo-relevant documents and combine them with in-batch negative samples, *i.e.*, the pseudo-relevant documents from other subqueries within the same batch, to form a candidate document set, which will be used to compute the document probability distribution.

Evaluation metrics & measurement protocol. We adopt exact match (EM) as the primary performance evaluation metric. The EM score measures whether the final predicted answer exactly matches the ground-truth answer. Before evaluation, both predicted and ground-truth answers are normalized by removing articles (e.g., a, an, the), stripping whitespace, removing punctuation, and converting all text to lowercase. For all retrieval-based methods, we retrieve the top-3 documents per query. The maximum number of retrieval iterations is set to 4. For efficiency, we report latency, which captures the end-to-end response time from receiving a query to generating the final answer. We sample the first 100 questions from each dataset to estimate latency. To enable fine-grained latency analysis, we report a breakdown of the latency across different stages, including prefill, thought generation, subquery generation, retrieval, and answer generation.

Following prior work [19, 35], we use Qwen2.5 Instruct for inference-based methods due to its stronger instruction-following capabilities. For training-based baselines, we adopt checkpoints released in the original papers that are based on the Base variant of Qwen2.5, which demonstrated better performance in prior work compared to the Instruct variant under training-based settings [19]. We also initialize and finetune our model from Qwen2.5 Base for a fair comparison.

For stage-wise latency measurement, the embedding time of a natural language subquery is included in the retrieval stage. For our method, to reduce the number of vectors transmitted to the retrieval system, we generate subquery embeddings on the model side from the latent tokens and pass only the resulting embedding vector to the retrieval system. Therefore, the embedding computation time is attributed to the subquery generation stage instead of the retrieval stage. This design leads to higher measured subquery generation time and lower retrieval time for our method. However, this difference does not affect the computation of the overall latency.

C Prompt Templates

In this section, we provide all prompt templates used in our framework. Double curly braces {{ $\cdots$ }} denote runtime placeholders. Prompt C.1 presents the template for latent thought and subquery

[5] https://github.com/facebookresearch/faiss
[6] https://huggingface.co/docs/transformers

generation. The latent thought and subquery tokens are derived from the hidden states at the positions of the corresponding special tokens. An action token is predicted based on the final thought token. If the action token is `<answer>`, the special subquery tokens in the prompt template are replaced with the answer token `<answer>` to trigger the answer generation process. Prompt C.2 and Prompt C.3 present the templates for latent thought and subquery decoding, respectively. Prompt C.4 presents the template used to generate synthetic questions from Wikipedia passages. These synthetic questions are used in the cross-retriever adaptation experiments without using real query data.

Prompt C.1. Prompt template for thought and subquery generation

Answer the following question by reasoning step by step and retrieving necessary information at each step:
{{QUESTION}}
<think$_1$>...<think$_m$><query$_1$>...<query$_n$>
<information>
Doc 1 {{TOP-1_DOCUMENT}}
Doc 2 {{TOP-2_DOCUMENT}}
Doc 3 {{TOP-3_DOCUMENT}}
</information>
...
<think$_1$>...<think$_m$><query$_1$>...<query$_n$>

Prompt C.2. Prompt template for latent thought decoding

Decode the thought based on the latent representation: {{LATENT_THOUGHT_TOKENS}}

Prompt C.3. Prompt template for latent subquery decoding

Decode the subquery based on the latent representation: {{LATENT_SUBQUERY_TOKENS}}

Prompt C.4. Prompt template for generating synthetic queries

Generate a question answer pair based on the given passage. The answer should be short (e.g., a single entity, a short phrase, or "yes"/"no"). Respond with a JSON object that contains a question and an answer, following the JSON schema below:

```
{
    "type": "object",
    "properties": {
        "question": {
            "type": "string",
        },
        "answer": {
            "type": "string",
        }
    },
    "required": ["question", "answer"]
}
```

Passage:

{{PASSAGE}}

D Dataset Description

We conduct our experiments on seven benchmark QA datasets, following previous works [19, 35]. These datasets include three general QA datasets (NQ [69], TriviaQA [70], and PopQA [71]) and four multi-hop QA datasets (HotpotQA [72], 2wiki [73], Musique [74], and Bamboogle [75]). Instead of using the original documents provided by each dataset as the retrieval corpus, we follow [66] and

Table 6: **Summary of datasets.**

	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle
# Train	79,168	–	–	90,447	–	–	–
# Test	3,610	11,313	14,267	7,405	12,576	2,417	125

Table 7: **Average latency (ms) with and without latent decoding across all datasets.**

Methods	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
Search-R1$^{\diamond}$	3,553	5,198	3,588	6,589	6,925	6,846	4,904	5,372
LatentRAG$^{\diamond}$ w/o decoding	491(**-86.2%**)	478(**-90.8%**)	501(**-86.0%**)	626(**-90.5%**)	704(**-89.8%**)	730(**-89.3%**)	623(**-87.3%**)	593(**-89.0%**)
LatentRAG$^{\diamond}$ w/ decoding	1,654(**-53.5%**)	1,855(**-64.3%**)	1,634(**-54.5%**)	2,110(**-68.0%**)	2,248(**-67.5%**)	2,285(**-66.6%**)	2,002(**-59.2%**)	1,970(**-63.3%**)
AutoRefine$^{\triangle}$	4,782	4,223	4,397	5,344	5,264	5,553	4,224	4,827
LatentRAG$^{\triangle}$ w/o decoding	409(**-91.4%**)	400(**-90.5%**)	422(**-90.4%**)	528(**-90.1%**)	607(**-88.5%**)	639(**-88.5%**)	581(**-86.2%**)	512(**-89.4%**)
LatentRAG$^{\triangle}$ w/ decoding	3,706(**-22.5%**)	2,403(**-43.1%**)	2,576(**-41.4%**)	2,571(**-51.9%**)	2,102(**-60.1%**)	2,300(**-58.6%**)	2,125(**-49.7%**)	2,540(**-47.4%**)

adopt a more challenging and realistic setting by using the full Wikipedia 2018 dump [76] as the corpus. The corpus contains 21,015,324 chunked documents, making retrieval significantly more difficult due to its large scale and diverse content. For training, we use the dataset splits provided by FlashRAG[7] and train our models on the training sets of NQ and HotpotQA. We evaluate all methods on the test sets (or development sets when test sets are unavailable) of the seven benchmark datasets. Table 6 summarizes the number of QA pairs in each dataset. Bamboogle is a very small dataset containing only 125 samples, which may lead to higher evaluation variance and less stable performance estimates compared to the other benchmark datasets.

E More Experimental Results

E.1 Embedding Space Analysis of Retrieval Models

In Table 1 of the main paper, compared to other retrieval models, our method exhibits a relatively larger performance drop when using e5-base-v2. To further investigate the source of this discrepancy, we analyze the differences in the geometric properties of the embedding space across different retrieval models. Specifically, for each retrieval model, we generate ℓ_2-normalized embeddings for the entire Wikipedia corpus. We then compute the mean direction of all document embeddings produced by that model. Next, we measure the cosine similarity and the angular distance between this mean direction and each document embedding and visualize their respective distributions. Any of these distributions skewed toward higher cosine similarities (or lower angles) indicates that the embeddings are concentrated around the mean direction rather than being uniformly distributed over the hypersphere, thus reflecting a stronger anisotropy [83, 84] in the embedding space.

As shown in Fig. 4, the embeddings generated by e5-base-v2 exhibit extremely high cosine similarity and low angular deviation with respect to the mean direction, demonstrating severe anisotropy. This suggests that the embeddings are highly concentrated around a narrow region of the hypersphere, rather than being well spread out. As a result, small approximation errors in the embedding space may lead to completely different retrieval outputs, making it challenging to train a model to faithfully approximate the behavior of the original retrieval model. Moreover, such a skewed distribution may force the LLM to deviate from its original representation geometry to adapt to this skewed concentrated space, which could negatively affect the performance of the LLM.

E.2 Latent Decoding Efficiency Analysis

As discussed in the main paper, latent decoding improves transparency at the cost of additional latency. A good property of our method is that the decoding of thoughts and subqueries is conditionally independent given the latent tokens. This property allows parallel decoding across different steps, which is more efficient than the sequential generation in existing agentic RAG methods.

To quantify the effect of reduced latency enabled by our parallel decoding strategy, we report detailed latency measurements across multiple datasets and compare them with baseline methods. As shown

[7] https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets

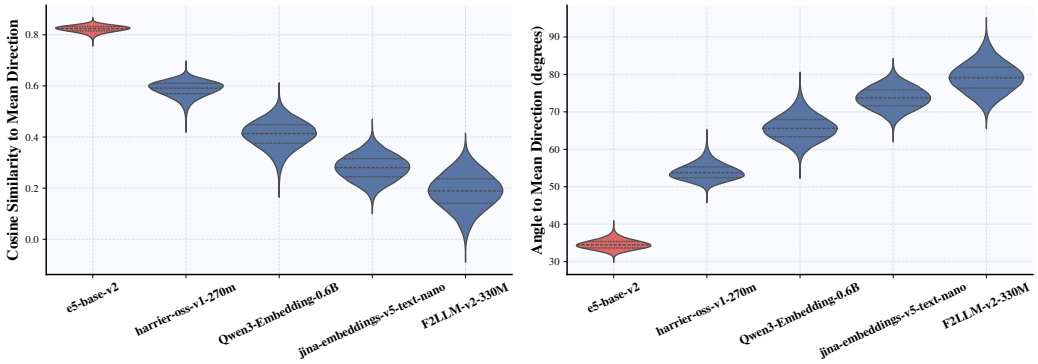


Figure 4: **Distribution of cosine similarity and angle between document embeddings and their mean direction.** We visualize distributions using violin plots. In contrast to other retrieval models, e5-base-v2 yields embeddings with extremely high cosine similarity and small angular deviation, indicating collapse into a narrow cone of the hypersphere and severe anisotropy.

in Table 7, using latent decoding increases latency by approximately 4–5× compared to the setting without latent decoding. Nevertheless, compared to corresponding baseline methods, our method with latent decoding reduces overall latency by approximately 23-68% across different datasets.

The efficiency gains from parallel decoding are more pronounced when sequence lengths are balanced, as this reduces padding overhead and avoids unnecessary computation. To characterize the impact of sequence length imbalance, we define the max length ratio as the ratio between the token count of the longest thought or subquery sequence and the total token count within a decoding batch. A higher max length ratio indicates a more imbalanced batch, where a single long sequence dominates most tokens. Such imbalance reduces the efficiency gains of parallel decoding due to increased padding overhead and more LLM forward passes. As shown in Fig. 5, the latency reduction percentage decreases as the max length ratio increases. This trend indicates that the effectiveness of parallel decoding is strongly associated with the degree of sequence length balance. Nevertheless, across different datasets with varying max length ratios, our method with latent decoding consistently achieves significant latency reductions, demonstrating the effectiveness of the parallel decoding strategy.

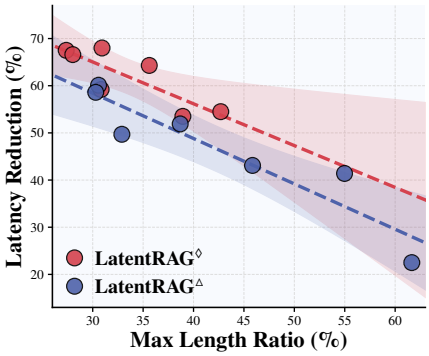


Figure 5: **Latency reduction using batch latent decoding vs. max length ratio.** Lower max length ratios are associated with higher latency reduction ratios. Each data point corresponds to the results on each dataset.

E.3 Detailed Stage-wise Latency Comparison

Table 8 shows the detailed stage-wise latency breakdown when using the Qwen3-Embedding-0.6B retrieval model. Compared to naive single-step RAG, Search-R1 and AutoRefine introduce significant latency overhead. The average latency across all datasets is approximately 15× that of naive RAG. This overhead mainly comes from the thought and subquery generation stages, which together account for about 90% of the total latency. In contrast, our method, trained on trajectories generated by Search-R1 and AutoRefine, significantly reduces the overall latency by approximately 90% compared to the corresponding baseline.

E.4 Impact of Trajectory Quality on Model Performance

To investigate the effect of trajectory quality on model performance, we train the same model using trajectories generated by LLMs of different sizes. As shown in Fig. 3 in the main paper, larger LLMs consistently achieve better performance, suggesting that they tend to produce higher-quality

Table 8: **Detailed stage-wise latency breakdowns (ms) using the Qwen3-Embedding-0.6B retrieval model.** Red and green highlights indicate the thought and subquery generation time of the baselines and LatentRAG, respectively. Search-R1 and AutoRefine incur significantly higher latency in the thought and subquery generation stages compared to our method.

Methods	Stages	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
Naive RAG	Prefill	51 (11.21%)	50 (14.93%)	49 (16.63%)	51 (16.94%)	50 (12.68%)	49 (11.54%)	49 (15.47%)	50 (13.85%)
	Retrieval	93 (20.54%)	93 (27.88%)	90 (30.49%)	90 (29.79%)	91 (23.21%)	91 (21.50%)	91 (28.80%)	91 (25.41%)
	Answer Gen.	309 (68.24%)	191 (57.19%)	157 (52.88%)	160 (53.26%)	251 (64.11%)	283 (66.96%)	175 (55.73%)	218 (60.74%)
	Total	452	334	297	300	391	422	314	359
Search-R1$^{\diamond}$	Prefill	101 (2.85%)	85 (1.64%)	87 (2.43%)	111 (1.68%)	131 (1.90%)	130 (1.89%)	108 (2.21%)	108 (2.00%)
	Thought Gen.	2,453 (69.04%)	3,919 (75.40%)	2,676 (74.58%)	5,110 (77.56%)	5,309 (76.67%)	5,105 (74.56%)	3,571 (72.82%)	4,021 (74.84%)
	Subquery Gen.	650 (18.30%)	701 (13.49%)	488 (13.60%)	942 (14.30%)	946 (13.66%)	1,112 (16.25%)	779 (15.88%)	803 (14.94%)
	Retrieval	166 (4.68%)	148 (2.84%)	154 (4.28%)	216 (3.28%)	282 (4.07%)	290 (4.23%)	212 (4.33%)	210 (3.90%)
	Answer Gen.	183 (5.14%)	345 (6.63%)	184 (5.12%)	210 (3.18%)	256 (3.70%)	210 (3.06%)	234 (4.77%)	231 (4.31%)
	Total	3,553	5,198	3,588	6,589	6,925	6,846	4,904	5,372
LatentRAG$^{\diamond}$	Prefill	95 (19.33%)	90 (18.85%)	86 (17.25%)	110 (17.52%)	123 (17.45%)	126 (17.26%)	104 (16.64%)	105 (17.67%)
	Thought Gen.	84 (17.01%)	81 (16.98%)	84 (16.77%)	103 (16.51%)	113 (16.01%)	116 (15.87%)	102 (16.43%)	98 (16.45%)
	Subquery Gen.	102 (20.86%)	100 (20.96%)	103 (20.52%)	144 (23.00%)	169 (23.96%)	176 (24.10%)	138 (22.08%)	133 (22.43%)
	Retrieval	96 (19.47%)	103 (21.52%)	110 (21.88%)	138 (22.06%)	159 (22.53%)	166 (22.66%)	129 (20.72%)	129 (21.65%)
	Answer Gen.	115 (23.32%)	104 (21.69%)	118 (23.59%)	131 (20.91%)	141 (20.05%)	147 (20.11%)	150 (24.12%)	129 (21.80%)
	Total	491(**-86.2%**)	478(**-90.8%**)	501(**-86.0%**)	626(**-90.5%**)	704(**-89.8%**)	730(**-89.3%**)	623(**-87.3%**)	593(**-89.0%**)
AutoRefine$^{\triangle}$	Prefill	87 (1.81%)	82 (1.94%)	83 (1.89%)	96 (1.80%)	115 (2.18%)	115 (2.07%)	99 (2.34%)	97 (2.00%)
	Thought Gen.	3,858 (80.67%)	3,063 (72.52%)	3,478 (79.11%)	3,987 (74.60%)	3,678 (69.87%)	3,879 (69.86%)	2,955 (69.94%)	3,557 (73.69%)
	Subquery Gen.	525 (10.97%)	777 (18.40%)	512 (11.65%)	890 (16.66%)	1,013 (19.24%)	1,119 (20.15%)	757 (17.91%)	799 (16.55%)
	Retrieval	127 (2.65%)	123 (2.92%)	141 (3.20%)	173 (3.23%)	223 (4.24%)	228 (4.10%)	190 (4.50%)	172 (3.56%)
	Answer Gen.	186 (3.90%)	178 (4.21%)	182 (4.14%)	198 (3.71%)	235 (4.46%)	213 (3.83%)	224 (5.30%)	202 (4.19%)
	Total	4,782	4,223	4,397	5,344	5,264	5,553	4,224	4,827
LatentRAG$^{\triangle}$	Prefill	80 (19.45%)	77 (19.31%)	75 (17.65%)	93 (17.66%)	104 (17.07%)	107 (16.74%)	97 (16.74%)	90 (17.63%)
	Thought Gen.	70 (17.05%)	70 (17.46%)	72 (17.09%)	90 (16.98%)	100 (16.53%)	105 (16.42%)	96 (16.53%)	86 (16.80%)
	Subquery Gen.	76 (18.67%)	76 (18.98%)	81 (19.10%)	111 (21.09%)	133 (21.90%)	140 (21.86%)	125 (21.41%)	106 (20.67%)
	Retrieval	71 (17.37%)	72 (17.98%)	76 (17.94%)	104 (19.65%)	125 (20.56%)	145 (22.71%)	117 (20.07%)	101 (19.77%)
	Answer Gen.	112 (27.46%)	105 (26.26%)	119 (28.22%)	130 (24.63%)	145 (23.94%)	142 (22.27%)	147 (25.25%)	129 (25.12%)
	Total	409(**-91.4%**)	400(**-90.5%**)	422(**-90.4%**)	528(**-90.1%**)	607(**-88.5%**)	639(**-88.5%**)	581(**-86.2%**)	512(**-89.4%**)

Table 9: **Effect of trajectory quality.** EM scores (%)$^{\uparrow}$ are reported for LatentRAG$^{\diamond}$-3B trained on trajectories generated by Search-R1$^{\diamond}$ models of different sizes (3B, 7B, and 14B). **Green** values show gains obtained by training with trajectories from larger models, compared to the 3B setting.

Methods	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
LatentRAG$^{\diamond}$-3B/3B	39.09	54.44	43.04	38.00	37.34	12.45	24.80	35.59
LatentRAG$^{\diamond}$-3B/7B	43.99(**+12.5%**)	56.77(**+4.3%**)	45.62(**+6.0%**)	43.36(**+14.1%**)	41.80(**+11.9%**)	17.25(**+38.6%**)	**37.60**(**+51.6%**)	40.91(**+14.9%**)
LatentRAG$^{\diamond}$-3B/14B	**44.90**(**+14.9%**)	**57.37**(**+5.4%**)	**45.90**(**+6.6%**)	**44.23**(**+16.4%**)	**44.86**(**+20.1%**)	**17.54**(**+40.9%**)	32.80(**+32.3%**)	**41.09**(**+15.4%**)

interaction trajectories. Therefore, we use trajectories generated by Search-R1 based on Qwen2.5 models of different scales to train our method with the Qwen2.5-3B model. As shown in Table 9, LatentRAG models trained on trajectories generated by the 7B and 14B models yield an average improvement of approximately 15% over the variant trained on trajectories generated by the 3B model. These improvements indicate that our method benefits significantly from higher-quality training trajectories, highlighting the importance of the quality of the model used for trajectory generation.

E.5 Frozen-Retriever Experiments

In our main experiments in the main paper, the retrieval model is not frozen to maintain the general flexibility of our framework for adapting to the retrieval alignment objective. In this section, we explore whether latent representations can be understood by a text-based retrieval model when only the projector is optimized, without finetuning the retrieval model. This setting isolates the contribution of retrieval model finetuning and provides a more controlled comparison with the baselines.

As shown in Table 10, the frozen-retriever variant of LatentRAG achieves similar performance to the fully finetuned variant across all retrieval models evaluated, and both variants exhibit relative performance differences of less than 5% compared to the Search-R1 baseline. This indicates that retrieval model finetuning is not a determining factor in the comparable performance of LatentRAG.

E.6 Evaluation on Long-Horizon QA Tasks

To evaluate whether LatentRAG can be applied to complex long-horizon QA tasks that require extensive multi-step reasoning and retrieval, we conduct additional experiments on two long-horizon

Table 10: **Exact match score (%) with and without retrieval model finetuning.** The full-finetuning and frozen-retriever variants of LatentRAG achieve similar performance, with limited or even no performance degradation compared to Search-R1.

Methods	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle	Average
Qwen3-Embedding-0.6B								
Search-R1$^{\diamond}$	43.93	**61.70**	**47.31**	43.05	43.27	19.61	38.40	42.47
LatentRAG$^{\diamond}$ (full-finetuning)	**46.29**	59.85	46.63	**46.73**	**44.89**	19.82	40.00	43.46 (+2.3%)
LatentRAG$^{\diamond}$ (frozen-retriever)	46.18	59.37	46.65	46.25	44.82	**21.22**	**40.80**	**43.61** (+2.7%)
e5-base-v2								
Search-R1$^{\diamond}$	48.03	**64.59**	**46.02**	44.71	**42.13**	**20.19**	**43.20**	**44.12**
LatentRAG$^{\diamond}$ (full-finetuning)	**49.03**	61.30	43.04	**46.41**	38.61	18.87	39.20	42.35 (-4.0%)
LatentRAG$^{\diamond}$ (frozen-retriever)	48.37	61.11	42.65	45.77	38.92	18.29	40.80	42.27 (-4.2%)
jina-embeddings-v5-text-nano								
Search-R1$^{\diamond}$	45.79	**63.16**	46.54	44.31	42.95	20.40	**44.80**	43.99
LatentRAG$^{\diamond}$ (full-finetuning)	47.37	61.25	**46.72**	**47.66**	45.06	**22.26**	43.20	**44.79** (+1.8%)
LatentRAG$^{\diamond}$ (frozen-retriever)	**48.12**	61.09	46.26	47.60	**45.42**	22.01	39.20	44.24 (+0.6%)
harrier-oss-v1-270m								
Search-R1$^{\diamond}$	44.40	**62.80**	**47.14**	44.88	44.16	18.62	**40.00**	**43.14**
LatentRAG$^{\diamond}$ (full-finetuning)	**46.15**	60.40	45.41	**47.27**	**44.99**	**19.90**	34.40	42.65 (-1.1%)
LatentRAG$^{\diamond}$ (frozen-retriever)	46.12	60.64	45.81	**47.27**	44.62	19.16	35.20	42.69 (-1.0%)
F2LLM-v2-330M								
Search-R1$^{\diamond}$	43.35	**61.04**	**44.92**	41.99	41.53	**18.54**	**36.00**	41.05
LatentRAG$^{\diamond}$ (full-finetuning)	45.54	58.56	44.21	**44.58**	43.02	18.37	**36.00**	**41.47** (+1.0%)
LatentRAG$^{\diamond}$ (frozen-retriever)	**46.29**	59.40	44.87	44.38	**43.59**	18.20	32.00	41.25 (+0.5%)

Table 11: **Long-horizon experiments.** LatentRAG achieves the best performance while significantly reducing latency and TTFT compared to the baselines.

Methods	Infoseek-Eval					BrowseComp-Plus				
	Acc.(%)$^{\uparrow}$	CEM(%)$^{\uparrow}$	Lat.(ms)$^{\downarrow}$	TTFT(ms)$^{\downarrow}$	# Steps	Acc.(%)$^{\uparrow}$	CEM(%)$^{\uparrow}$	Lat.(ms)$^{\downarrow}$	TTFT(ms)$^{\downarrow}$	# Steps
AgentCPM	46.33	38.67	165.88	131.48	21.61	14.70	14.10	468.46	438.63	29.10
Text-SFT	44.00	38.33	155.38	114.70	21.25	14.94	13.13	355.12	314.41	28.98
LatentRAG	53.00	45.00	44.84	4.61	19.39	16.14	16.27	46.31	6.80	28.78

QA benchmarks, Infoseek-Eval [85] and BrowseComp-Plus [86], both of which provide curated corpora for constructing local retrieval systems.

Experimental Setup. We use AgentCPM-4B [87] as the teacher model, which is developed for long-horizon tasks. We construct training trajectories using the Infoseek-Train dataset [85] and train LatentRAG based on the Qwen3-4B-Thinking-2507 model [88]. We further train a text-based SFT model using the same training trajectories as LatentRAG, denoted as Text-SFT. Text-SFT follows the same design as LatentRAG, except that it operates in the text space rather than the latent space, which serves as a controlled baseline for isolating the effect of latent reasoning and retrieval.

Metrics. Following the evaluation protocol of BrowseComp-Plus, we report accuracy measured by Qwen3-32B as the judge model. We also report the cover exact match score (CEM), a rule-based metric that measures whether the ground-truth answer is covered by the generated answer. Since models on long-horizon QA tasks often generate lengthy answers with detailed explanations, answer generation can account for a substantial portion of the total latency, making the efficiency gains from multi-step retrieval less apparent. Therefore, in addition to average end-to-end latency, we report the time to first token (TTFT) as an additional efficiency metric, which measures the time from receiving a request to generating the first answer token. By excluding the subsequent answer generation phase, TTFT directly captures the efficiency gains from reducing reasoning and retrieval overhead. TTFT also better reflects the perceived waiting time before a user receives the response.

Results. As shown in Table 11, all methods take about 20 steps on Infoseek-Eval and 30 steps on BrowseComp-Plus, demonstrating that these tasks require substantially more iterative reasoning and retrieval than the traditional QA benchmarks. LatentRAG achieves the best performance on both

datasets while reducing the overall latency of AgentCPM by approximately 73% on Infoseek-Eval and 90% on BrowseComp-Plus. As LatentRAG primarily accelerates the reasoning and retrieval stages before answer generation, it achieves substantial TTFT reductions compared to AgentCPM, with a reduction of approximately 96% on InfoSeek-Eval and 98% on BrowseComp-Plus.

E.7 Latency Comparison using Optimized Serving Frameworks

In our main experiments in the main paper, we measure the latency using the standard forward pass implemented in Hugging Face Transformers[8]. Beyond the standard Hugging Face Transformers implementation, recent serving frameworks such as vLLM[9], TensorRT-LLM[10], and SGLang[11] are widely adopted to optimize the inference process. However, existing serving frameworks are not optimized for the latent retrieval workflow and cannot be directly applied to our method. Specifically, vLLM and TensorRT-LLM do not support accessing hidden states, and SGLang only supports full hidden state extraction, which introduces unnecessary overhead since latent retrieval only requires hidden states corresponding to the special subquery tokens (around 2% of the full sequence). Therefore, we evaluate latency with the standard HuggingFace Transformers forward pass for the main experiments to ensure a fair comparison, which is consistent with the evaluation protocols commonly adopted by previous works on latent reasoning [26, 61].

Apart from the standard decoding implementation, most serving frameworks also implement speculative decoding [89], which is widely adopted to mitigate the autoregressive bottleneck and reduce latency in traditional text-based decoding. Speculative decoding incorporates a small draft model to quickly generate multiple candidate tokens, which are then verified by the target model in a single forward pass. However, this approach requires the draft model to closely match the target model to achieve a high acceptance rate. In complex agentic RAG scenarios, a smaller draft model may struggle to generate outputs that are consistent with the target model. Moreover, speculative decoding introduces additional computational overhead from draft generation and verification. In contrast, LatentRAG requires only a single forward pass per retrieval step without this additional overhead.

To further assess whether the latency advantage of LatentRAG persists under optimized serving, we additionally compare LatentRAG, without any serving-specific optimization, against the baseline method Search-R1 served with vLLM. This provides a conservative comparison for LatentRAG, as Search-R1 benefits from optimized serving while LatentRAG does not. Furthermore, beyond the standard inference implementation in vLLM, we also evaluate its speculative decoding implementation to investigate whether it can further reduce the latency gap between Search-R1 and LatentRAG. We evaluate speculative decoding with two draft models: the base Qwen2.5-0.5B model and a finetuned variant, denoted as SFT-Qwen2.5-0.5B, which is further finetuned on Search-R1 trajectories to improve the speculative acceptance rate.

Our evaluation focuses on minimizing per-request latency under sufficient computational resources, rather than optimizing throughput under resource constraints through batching. While throughput under resource constraints can be improved by allocating additional computational resources, reducing per-request latency remains a more challenging problem due to the sequential dependencies inherent in autoregressive generation. Therefore, our evaluation focuses on the fundamental latency lower bound of individual requests under sufficient computational resources, with all experiments conducted at a batch size of 1. Nevertheless, LatentRAG also reduces inference computational costs and may offer throughput benefits. However, a rigorous evaluation of throughput requires integrating our method with advanced batch inference frameworks, which we leave as future work.

As shown in Table 12, Search-R1 with vLLM serving significantly reduces inference latency compared to the HF Transformers implementation of Search-R1. However, even without any serving optimization, LatentRAG still achieves lower latency than Search-R1 with vLLM serving. Furthermore, applying speculative decoding with Qwen2.5-0.5B as the draft model does not reduce Search-R1 latency in our setting. Instead, it increases latency due to the additional draft generation and verification. Using the finetuned model SFT-Qwen2.5-0.5B as the draft model improves the speculative acceptance rate (from 32.48% to 69.61% when using 10 speculative tokens), but the

[8] https://huggingface.co/docs/transformers
[9] https://github.com/vllm-project/vllm
[10] https://github.com/NVIDIA/TensorRT-LLM
[11] https://github.com/sgl-project/sglang

Table 12: **Latency comparison with vLLM.** P95 and P99 denote the 95th and 99th percentile tail latencies, respectively. LatentRAG, even without any serving optimization, achieves lower latency than Search-R1 with vLLM serving.

Methods	Draft Model	# Spec. Tokens	Latency (ms)↓	P95 (ms)↓	P99 (ms)↓
LatentRAG (HF Transformers)	–	–	593 (1.0×)	821	886
Search-R1 (HF Transformers)	–	–	5,372 (9.1×)	9,737	16,843
Search-R1 (vLLM)	–	–	1,506 (2.5×)	2,454	2,899
Search-R1 (vLLM + Spec.)	Qwen2.5-0.5B	3	2,129 (3.6×)	3,222	3,733
Search-R1 (vLLM + Spec.)	Qwen2.5-0.5B	5	2,788 (4.7×)	4,279	4,742
Search-R1 (vLLM + Spec.)	Qwen2.5-0.5B	10	4,546 (7.7×)	6,737	7,408
Search-R1 (vLLM + Spec.)	SFT-Qwen2.5-0.5B	3	1,738 (2.9×)	2,850	3,337
Search-R1 (vLLM + Spec.)	SFT-Qwen2.5-0.5B	5	2,048 (3.5×)	3,425	3,912
Search-R1 (vLLM + Spec.)	SFT-Qwen2.5-0.5B	10	2,750 (4.6×)	4,597	5,490

Table 13: **Average token counts and number of forward passes per question.** **(in)** and **(out)** denote input and output tokens, respectively. Due to autoregressive token-by-token generation, output tokens incur more forward passes and thus higher latency.

Methods	Token Counts					# Forward Passes
	Thought	Subquery	Answer	Others	Total	
Search-R1$^{\diamond}$	121.8 **(out)**	37.9 **(out)**	9.6 **(out)**	1325.5 **(in)**	1325.5 **(in)** + 169.4 **(out)**	169.4
LatentRAG$^{\diamond}$ w/o decoding	13.8 **(in)**	39.0 **(in)**	5.8 **(out)**	1222.5 **(in)**	1275.2 **(in)** + 5.8 **(out)**	**11.7**
LatentRAG$^{\diamond}$ w/ decoding	13.8 **(in)** + 117.7 **(out)**	39.0 **(in)** + 29.3 **(out)**	5.8 **(out)**	1436.2 **(in)**	1489.0 **(in)** + 152.8 **(out)**	**52.8**

overall latency still underperforms the standard vLLM inference. The P95 and P99 tail latencies show similar trends, with LatentRAG achieving the lowest latency.

E.8 Influence of Latent Token Numbers

To investigate the impact of latent token numbers, we vary the number of latent thought tokens m and the number of subquery tokens n and evaluate the exact match scores under different configurations. As shown in Fig. 6, performance remains relatively stable across different settings. It increases slightly at first and reaches a peak when using 4 thought tokens and 16 subquery tokens for each step, followed by a decline as the token numbers continue to increase. This suggests that while additional latent tokens can provide more expressive capacity and increase performance, excessive tokens may introduce redundancy. Therefore, in our experiments, we set $m = 4$ and $n = 16$.

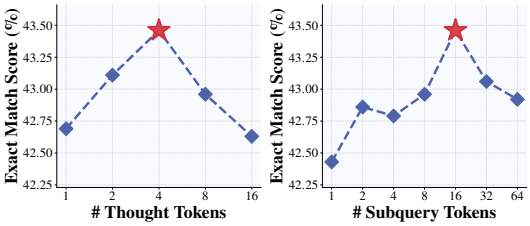


Figure 6: **performance under different numbers of latent thought and subquery tokens.**

E.9 Average Token Counts and Number of Forward Passes

To analyze token usage efficiency, we report the average token counts per question. We distinguish between input and output tokens, as output tokens are generated autoregressively and cannot be fully parallelized, typically incurring higher latency and being more costly in practice. For example, in the OpenAI API pricing[12], output tokens are typically priced about 6× higher than input tokens. We also report the number of forward passes per question. The number of forward passes corresponds to how many sequential LLM forward computations are required, which typically relates to the overall latency under sufficient hardware resources. Moreover, this latency cannot be easily reduced by simply scaling up GPU computational resources as it is fundamentally constrained by sequential dependencies in the generation process.

As shown in Table 13, Search-R1 generates substantially more output tokens due to explicit thought and subquery generation, which in turn leads to a large number of LLM forward passes and explains

[12] https://openai.com/api/pricing/

its high latency reported in the main paper. In contrast, our method directly computes latent thought or subquery tokens by feeding a sequence of special tokens in parallel, which requires only a single forward pass per thought or subquery. As a result, without latent decoding, our method requires less than 5% of the output tokens compared to Search-R1, significantly reducing the number of forward passes and thereby achieving substantially lower latency, as reported in the main paper. As an option to improve transparency at the cost of additional latency, latent decoding increases the number of output tokens in our method to a level comparable to Search-R1. However, since in our method the thought and subquery sequences across different steps are conditionally independent given the latent tokens, these sequences can be decoded in parallel, which significantly reduces the number of LLM forward passes. Moreover, the decoding process depends only on the latent tokens rather than attending to the full interaction history, which can further reduce computational overhead in practice. Consequently, even with a comparable number of output tokens, our method with latent decoding still only requires much fewer forward passes and achieves higher efficiency.

E.10 Case Studies

To qualitatively analyze the behavior of our method, we present several case studies of the reasoning and retrieval processes of LatentRAG.

Success case analysis. As shown in Success Case 1 and 2, our method successfully learns the reasoning and retrieval patterns of the respective baseline models. Models trained on trajectories from different baselines generate thoughts and subqueries that are similar to those of the original models. For instance, the decoded thoughts of LatentRAG$^{\diamondsuit}$ are able to capture the refinement structure in the reasoning process of AutoRefine. In Success Case 3, although both models arrive at the correct answer after a sequence of reasoning and retrieval steps, they exhibit redundant retrieval in the final stage. This suggests that undesirable behaviors of the teacher model may also be learned by our method, highlighting the importance of trajectory quality as discussed in Appendix E.4.

Failure case analysis. As shown in Failure Case 1 and 2, although the reasoning and retrieval processes of our method are correct, the model sometimes fails to produce fully consistent outputs, leading to incorrect answers under exact match evaluation. This might indicate that latent representations facilitate the learning of abstract concepts but are less effective for precise lexical output. Nevertheless, our method maintains competitive performance while significantly reducing overall latency by 90%, highlighting the value of latent reasoning and retrieval in agentic RAG. Future research can further investigate how to balance the use of latent representations and precise contextual information for accurate answer generation.

LogitLens analysis. To investigate what information is encoded in each latent thought or subquery token, we leverage LogitLens [90] to analyze the generated latent tokens. LogitLens projects hidden states into the vocabulary space using the unembedding matrix of the LLM, enabling inspection of the token-level information encoded in the hidden states. Fig. 7 & 8 present the top-5 predicted language tokens by logits for each latent token. Although we do not explicitly constrain latent tokens to align with the LLM vocabulary space, the model still distributes these latent representations around semantically related vocabulary regions. In particular, the decoded vocabulary tokens from the thought and subquery tokens of the first step are closely related to the first subquery, while those from later steps gradually shift toward vocabulary regions associated with the second subquery and eventually the final answer. This progression indicates that the latent representations evolve as the reasoning process proceeds, with different latent tokens capturing information relevant to different stages of the reasoning and retrieval process. Additionally, unlike natural language tokenization, which typically represents text through a fixed subword decomposition that may split semantic units across multiple tokens, a latent token can encode the whole semantic concept, such as *Christianity Today* or *William Goldman*. These findings suggest that performing reasoning and retrieval in the latent space may provide more flexibility and expressivity than operating in natural language space.

Success Case 1: Search-R1$^{\diamond}$ vs. LatentRAG$^{\diamond}$

[Question]	What is the place of birth of Clara Novello's father? *(Ground truth answer: London)*
Search-R1$^{\diamond}$:	
[Thought]	I need to find the place of birth of Clara Novello's father. I'll search for it.
[Subquery]	Clara Novello's father
[Docs]	**Chunk_11245233 [Clara Novello]**: Ivor Novello carried on this family name. Clara Novello Clara Anastasia Novello (10 June 1818 – 12 March 1908) was an acclaimed soprano, the fourth daughter of Vincent Novello, a musician and music publisher, and his wife, Mary Sabilla Hehl. Her acclaimed soprano and pure style made her one of the greatest vocalists, alike in opera, oratorio and on the concert stage, from 1833 onwards. In 1843 she married Count Gigliucci, and retired in 1861. Charles Lamb wrote a poem ("To Clara N.") in her praise. She was born in Oxford Street, London, on 10 June 1818, the fourth daughter ...
[Thought]	I found out that Clara Novello's father is Vincent Novello. Now I need to find the place of birth of Vincent Novello.
[Subquery]	place of birth of Vincent Novello
[Docs]	**Chunk_10809160 [Vincent Novello]**: Vincent Novello (6 September 1781 – 9 August 1861), English musician, son of an Italian who married an English wife, was born in London. He was a chorister and organist, but he is best known for bringing to England many works now considered major standards, and with his son he created a major publishing house. As a boy, Novello was a chorister at the Sardinian Embassy Chapel in Duke Street, Lincoln's Inn Fields, where he learnt the organ from Samuel Webbe; and from 1796 to 1822 he became in succession organist of the Sardinian, Spanish (in Manchester Square) and Portuguese ...
[Thought]	I found out that the place of birth of Vincent Novello is London. Now I can provide the answer.
[Answer]	London
LatentRAG$^{\diamond}$:	
[Thought (dec)]	I need to find the place of birth of Clara Novello's father. I'll search for it.
[Subquery (dec)]	Clara Novello's father
[Docs]	**Chunk_11245233 [Clara Novello]**: Ivor Novello carried on this family name. Clara Novello Clara Anastasia Novello (10 June 1818 – 12 March 1908) was an acclaimed soprano, the fourth daughter of Vincent Novello, a musician and music publisher, and his wife, Mary Sabilla Hehl. Her acclaimed soprano and pure style made her one of the greatest vocalists, alike in opera, oratorio and on the concert stage, from 1833 onwards. In 1843 she married Count Gigliucci, and retired in 1861. Charles Lamb wrote a poem ("To Clara N.") in her praise. She was born in Oxford Street, London, on 10 June 1818, the fourth daughter ...
[Thought (dec)]	I found out that Clara Novello's father is Vincent Novello. Now I need to find his place of birth.
[Subquery (dec)]	Vincent Novello place of birth
[Docs]	**Chunk_10809160 [Vincent Novello]**: Vincent Novello (6 September 1781 – 9 August 1861), English musician, son of an Italian who married an English wife, was born in London. He was a chorister and organist, but he is best known for bringing to England many works now considered major standards, and with his son he created a major publishing house. As a boy, Novello was a chorister at the Sardinian Embassy Chapel in Duke Street, Lincoln's Inn Fields, where he learnt the organ from Samuel Webbe; and from 1796 to 1822 he became in succession organist of the Sardinian, Spanish (in Manchester Square) and Portuguese ...
[Thought (dec)]	I found out that the place of birth of Vincent Novello is London. Now I can provide the answer.
[Answer]	London

Success Case 2: AutoRefine$^{\Delta}$ vs. LatentRAG$^{\Delta}$

[Question]	What is the place of birth of Clara Novello's father? *(Ground truth answer: London)*
AutoRefine$^{\Delta}$:	
[Thought]	To answer this question, I need to find out who Clara Novello's father is and then determine his place of birth.
[Subquery]	Clara Novello's father
[Docs]	**Chunk_11245233 [Clara Novello]**: Ivor Novello carried on this family name. Clara Novello Clara Anastasia Novello (10 June 1818 – 12 March 1908) was an acclaimed soprano, the fourth daughter of Vincent Novello, a musician and music publisher, and his wife, Mary Sabilla Hehl. Her acclaimed soprano and pure style made her one of the greatest vocalists, alike in opera, oratorio and on the concert stage, from 1833 onwards. In 1843 she married Count Gigliucci, and retired in 1861. Charles Lamb wrote a poem ("To Clara N.") in her praise. She was born in Oxford Street, London, on 10 June 1818, the fourth daughter ...
[Thought]	<refine>From the documents, I learned that the father of Clara Novello was Vincent Novello. </refine>
[Subquery]	Vincent Novello's place of birth
[Docs]	**Chunk_10809160 [Vincent Novello]**: Vincent Novello (6 September 1781 – 9 August 1861), English musician, son of an Italian who married an English wife, was born in London. He was a chorister and organist, but he is best known for bringing to England many works now considered major standards, and with his son he created a major publishing house. As a boy, Novello was a chorister at the Sardinian Embassy Chapel in Duke Street, Lincoln's Inn Fields, where he learnt the organ from Samuel Webbe; and from 1796 to 1822 he became in succession organist of the Sardinian, Spanish (in Manchester Square) and Portuguese ...
[Thought]	<refine>From the documents, I learned that Vincent Novello was born in London. </refine>
[Answer]	London
LatentRAG$^{\Delta}$:	
[Thought (dec)]	To answer this question, I need to find out who Clara Novello's father is and then determine his place of birth.
[Subquery (dec)]	Clara Novello's father
[Docs]	**Chunk_11245233 [Clara Novello]**: Ivor Novello carried on this family name. Clara Novello Clara Anastasia Novello (10 June 1818 – 12 March 1908) was an acclaimed soprano, the fourth daughter of Vincent Novello, a musician and music publisher, and his wife, Mary Sabilla Hehl. Her acclaimed soprano and pure style made her one of the greatest vocalists, alike in opera, oratorio and on the concert stage, from 1833 onwards. In 1843 she married Count Gigliucci, and retired in 1861. Charles Lamb wrote a poem ("To Clara N.") in her praise. She was born in Oxford Street, London, on 10 June 1818, the fourth daughter ...
[Thought (dec)]	<refine>From the documents, I learned that the father of Clara Novello was Vincent Novello. </refine>
[Subquery (dec)]	What is the place of birth of Vincent Novello
[Docs]	**Chunk_10809160 [Vincent Novello]**: Vincent Novello (6 September 1781 – 9 August 1861), English musician, son of an Italian who married an English wife, was born in London. He was a chorister and organist, but he is best known for bringing to England many works now considered major standards, and with his son he created a major publishing house. As a boy, Novello was a chorister at the Sardinian Embassy Chapel in Duke Street, Lincoln's Inn Fields, where he learnt the organ from Samuel Webbe; and from 1796 to 1822 he became in succession organist of the Sardinian, Spanish (in Manchester Square) and Portuguese ...
[Thought (dec)]	<refine>From the documents, I learned that Vincent Novello was born in London. </refine>
[Answer]	London

Success Case 3: Search-R1$^{\diamond}$ vs. LatentRAG$^{\diamond}$

[Question]	Which film was released first, The Beginning Or The End or Hold Back The Night? *(Ground truth answer: The Beginning or the End)*
Search-R1$^{\diamond}$:	
[Thought]	I need to determine which film, The Beginning or The End or Hold Back The Night, was released first. I'll search for the release years of both films.
[Subquery]	The Beginning or The End
[Docs]	**Chunk_15135042 [The Beginning Was the End]**: The Beginning Was the End is a 1971 pseudo-scientific book written by ... (Irrelevant retrieval results) ...
[Thought]	I found out that The Beginning or The End is a book. Now I need to find the release year of the film Hold Back The Night and compare it to the release year of the film based on the book The Beginning or The End.
[Subquery]	Hold Back The Night release year
[Docs]	**Chunk_15896613 [Hold Back the Night]**: Hold Back the Night is a 1956 American war film about the Korean War based on the 1951 novel by Pat Frank, who had been a war correspondent in Korea. The film was directed by Allan Dwan; his third film with John Payne and his third film about the United States Marine Corps, the others being "Abroad with Two Yanks" (1944) and "Sands of Iwo Jima" (1949). The film tells the story in flashbacks of a bottle of scotch carried by a World War II Marine lieutenant and Korean War captain, Sam MacKenzie. His girl Anne ...
[Thought]	I found out that Hold Back The Night is a film released in 1956. Now I need to find the release year of the film based on the book The Beginning or The End.
[Subquery]	The Beginning or The End release year
[Docs]	**Chunk_10601403 [The Beginning or the End]**: seldom rises above cheery imbecility." According to MGM records, "The Beginning or the End" was made on a budget of $2,632,000 (), but earned $1,221,000 () in the United States and Canada and $721,000 () elsewhere, resulting in a loss to the studio of $1,596,000 (). The Beginning or the End is a 1947 American docudrama film about the development of the atomic bomb in World War II, directed by Norman Taurog, starring Brian Donlevy and Hume Cronyn, and released by Metro-Goldwyn-Mayer. The film dramatizes the creation of the atomic bomb in the Manhattan Project ..
[Thought]	I found out that The Beginning or the End is a film released in 1947. Now I can compare the release years of both films:
[Subquery]	Release year of Hold Back The Night (Redundant retrieval)
[Docs]	**Chunk_15896613 [Hold Back the Night]**: ... ...
[Thought]	I found out that Hold Back The Night was released in 1956. Now I can compare it to the release year of The Beginning or the End, which I found to be 1947.
[Answer]	The Beginning or the End
LatentRAG$^{\diamond}$:	
[Thought (dec)]	I need to determine which film, The Beginning or The End or Hold Back the Night, was released first. I'll search for the release years of both films.
[Subquery (dec)]	The Beginning Or The End
[Docs]	**Chunk_10601403 [The Beginning or the End]**: seldom rises above cheery imbecility." According to MGM records, "The Beginning or the End" was made on a budget of $2,632,000 (), but earned $1,221,000 () in the United States and Canada and $721,000 () elsewhere, resulting in a loss to the studio of $1,596,000 (). The Beginning or the End is a 1947 American docudrama film about the development of the atomic bomb in World War II, directed by Norman Taurog, starring Brian Donlevy and Hume Cronyn, and released by Metro-Goldwyn-Mayer. The film dramatizes the creation of the atomic bomb in the Manhattan Project ..
[Thought (dec)]	I found out that The Beginning or The End is a 1947 American documentary film. Now I need to find the release year of Hold Back the Night.
[Subquery (dec)]	Hold Back The Night
[Docs]	**Chunk_15896613 [Hold Back the Night]**: Hold Back the Night is a 1956 American war film about the Korean War based on the 1951 novel by Pat Frank, who had been a war correspondent in Korea. The film was directed by Allan Dwan; his third film with John Payne and his third film about the United States Marine Corps, the others being "Abroad with Two Yanks" (1944) and "Sands of Iwo Jima" (1949). The film tells the story in flashbacks of a bottle of scotch carried by a World War II Marine lieutenant and Korean War captain, Sam MacKenzie. His girl Anne ...
[Thought (dec)]	I found out that Hold Back the Night was released in 1956. Now I can compare the release years of both films. The Beginning or The End was released in 1947, and Hold Back the Night was released in 1956.
[Subquery (dec)]	release year of The Beginning of the End (Redundant retrieval)
[Docs]	**Chunk_10601403 [The Beginning or the End]**: ... ..
[Thought (dec)]	I found out that The Beginning or The End (1947 film) was released in 1947. Now I can compare the release years of both films.
[Answer]	The Beginning or the End

Failure Case 1: Search-R1$^{\diamond}$ vs. LatentRAG$^{\diamond}$

[Question]	Who lead the enquiry into the Iraq war? *(Ground truth answer: Sir John Chilcot)*
Search-R1$^{\diamond}$:	
[Thought]	I need to find out who led the enquiry into the Iraq war. I'll search for it.
[Subquery]	who led the enquiry into the Iraq war
[Docs]	**Chunk_6411881 [Iraq Inquiry]**: The Iraq Inquiry (also referred to as the Chilcot Inquiry after its chairman, Sir John Chilcot) was a British public inquiry into the nation's role in the Iraq War. The inquiry was announced in 2009 by Prime Minister Gordon Brown and published in 2016 with a public statement by Chilcot. On 6 July 2016, Sir John Chilcot announced the report's publication, more than seven years after the inquiry was announced. Usually referred to as the Chilcot report by the news media, the document stated that at the time of the invasion of Iraq in 2003, Saddam Hussein did
[Thought]	I found out that the Iraq Inquiry, also referred to as the Chilcot Inquiry, was led by Sir John Chilcot. Now I can provide the answer. ...
[Answer]	Sir John Chilcot
LatentRAG$^{\diamond}$:	
[Thought (dec)]	I need to find out who led the inquiry into the Iraq War. I'll search for it.
[Subquery (dec)]	who led the inquiry into the Iraq War
[Docs]	**Chunk_6411881 [Iraq Inquiry]**: The Iraq Inquiry (also referred to as the Chilcot Inquiry after its chairman, Sir John Chilcot) was a British public inquiry into the nation's role in the Iraq War. The inquiry was announced in 2009 by Prime Minister Gordon Brown and published in 2016 with a public statement by Chilcot. On 6 July 2016, Sir John Chilcot announced the report's publication, more than seven years after the inquiry was announced. Usually referred to as the Chilcot report by the news media, the document stated that at the time of the invasion of Iraq in 2003, Saddam Hussein did ...
[Thought (dec)]	I found out that the Iraq War was led by Sir John Chilworth. Now I can provide the answer.
[Answer]	Sir John Chilcott

Failure Case 2: Search-R1$^{\diamond}$ vs. LatentRAG$^{\diamond}$

[Question]	What was the name of the dog in the novel "Three Men in a Boat"? *(Ground truth answer: Montmorency)*
Search-R1$^{\diamond}$:	
[Thought]	I need to find the name of the dog in the novel "Three Men in a Boat". I'll search for it.
[Subquery]	name of the dog in the novel "Three Men in a Boat"
[Docs]	**Chunk_8650768 [Three Men in a Boat]**: is how undated it appears to modern readers – the jokes have been praised as fresh and witty. The three men are based on Jerome himself (the narrator Jerome K. Jerome) and two real-life friends, George Wingrave (who would become a senior manager at Barclays Bank) and Carl Hentschel (the founder of a London printing business, called Harris in the book), with whom Jerome often took boating trips. The dog, Montmorency, is entirely fictional but, "as Jerome admits, developed out of that area of inner consciousness which, in all Englishmen, contains an element of the dog". The trip is ...
[Thought]	I found out that the dog in the novel "Three Men in a Boat" is named Montmorency. Now I can provide the answer.
[Answer]	Montmorency
LatentRAG$^{\diamond}$:	
[Thought (dec)]	I need to find the name of the dog in the novel "Three Men in a Boat". I'll search for it.
[Subquery (dec)]	what was the name of the dog in the novel "Three Men in a Boat"
[Docs]	**Chunk_8650768 [Three Men in a Boat]**: is how undated it appears to modern readers – the jokes have been praised as fresh and witty. The three men are based on Jerome himself (the narrator Jerome K. Jerome) and two real-life friends, George Wingrave (who would become a senior manager at Barclays Bank) and Carl Hentschel (the founder of a London printing business, called Harris in the book), with whom Jerome often took boating trips. The dog, Montmorency, is entirely fictional but, "as Jerome admits, developed out of that area of inner consciousness which, in all Englishmen, contains an element of the dog". The trip is ...
[Thought (dec)]	I found out that the name of the dog in the novel "Three Men in a Boat" is Montmorencie. Now I can provide the answer.
[Answer]	Montmoreiras

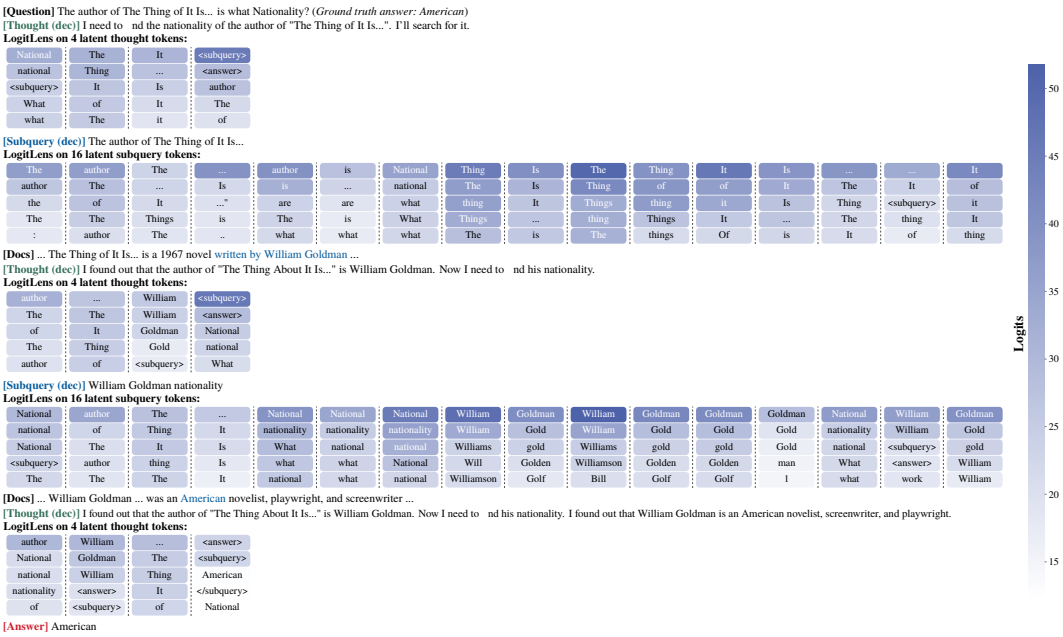


Figure 7: **LogitLens Case Study 1 on LatentRAG$^{\diamondsuit}$.** Latent thought and subquery tokens in the first step align with tokens related to the first subquery, *The author of The Thing of It Is...*, while those in the second step shift toward tokens related to the second subquery, *William Goldman nationality*. A latent token can encode the whole semantic concept, such as *The Thing of It Is...* or *William Goldman*.

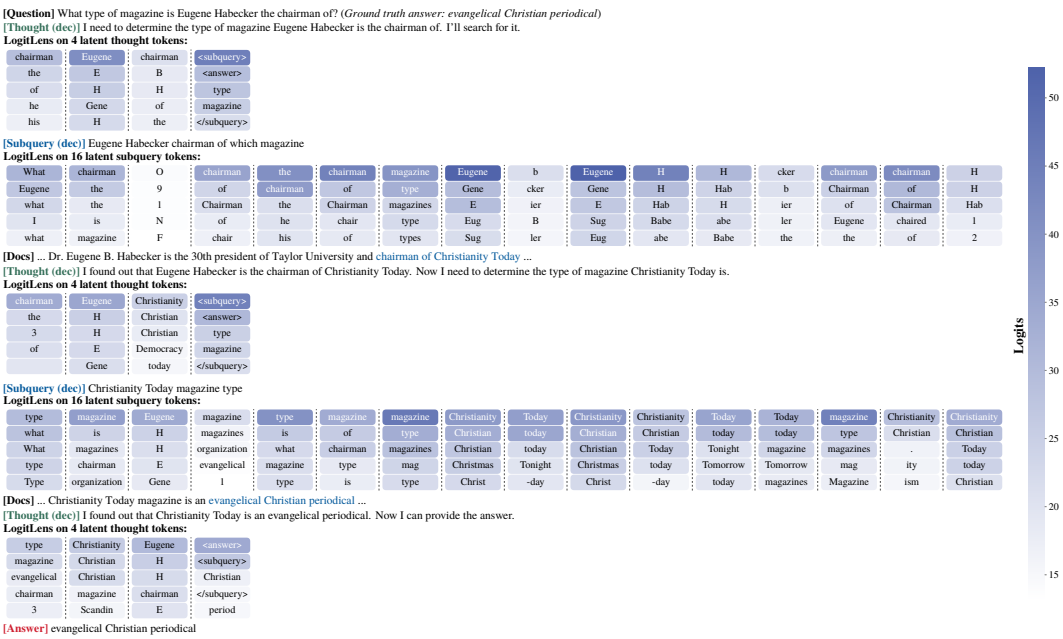


Figure 8: **LogitLens Case Study 2 on LatentRAG$^{\diamondsuit}$.** Latent thought and subquery tokens in the first step align with tokens related to the first subquery, *Eugene Habecker chairman of which magazine*, while those in the second step shift toward tokens related to the second subquery, *Christianity Today magazine type*. A latent token can encode the whole semantic concept, such as *magazine type* or *Christianity Today*.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We tried our best to make the contributions and scope precise in the abstract and introduction (Section 1). The motivation can be found in the third and fourth paragraphs of Section 1, while our main contributions are summarized in the last paragraph of Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations and future work in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the details of our experiments in Section 5 and Appendix B. The source code for reproducing the experiments is provided via the Github link: https://github.com/zhengyijia/LatentRAG.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: We provide the processed data used in our experiments and the source code for reproducing our experiments via the Github link: https://github.com/zhengyijia/LatentRAG. We also provide the details of our experiments in Section 5 and Appendix B.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide the experimental setup and details in Section 5 and Appendix B. The full details are provided with the source code via the Github link: https://github.com/zhengyijia/LatentRAG.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to the high computational cost of large-scale LLM-based experiments, we do not include error bars. Instead, we conduct extensive evaluations across multiple retriever architectures and model scales. The proposed method consistently achieves comparable performance while significantly reducing latency by approximately 90% across all settings, suggesting that the efficiency improvements are stable and not sensitive to specific configurations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information on the computer resources and execution time is provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and conformed with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of this work in Appendix A.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The main contribution of this work is reducing the latency of existing agentic RAG methods. We do not introduce any newly pretrained language models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide references to the original papers or official URLs for all datasets and models used in this work (Appendix 5). In addition, we include links to all tools and libraries employed in our implementation (Section B).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide the detailed documentation (`README.md`) for our code via the Github link: `https://github.com/zhengyijia/LatentRAG`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We clearly describe the usage of LLMs as search agents in our agentic RAG framework in the methodology section (Section 4). The role of LLMs in the core method is explicitly defined and constitutes a central component of our approach. Beyond this, LLMs are only used for writing, editing, or formatting purposes, which do not affect the scientific contributions of the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.