

Framework Overview

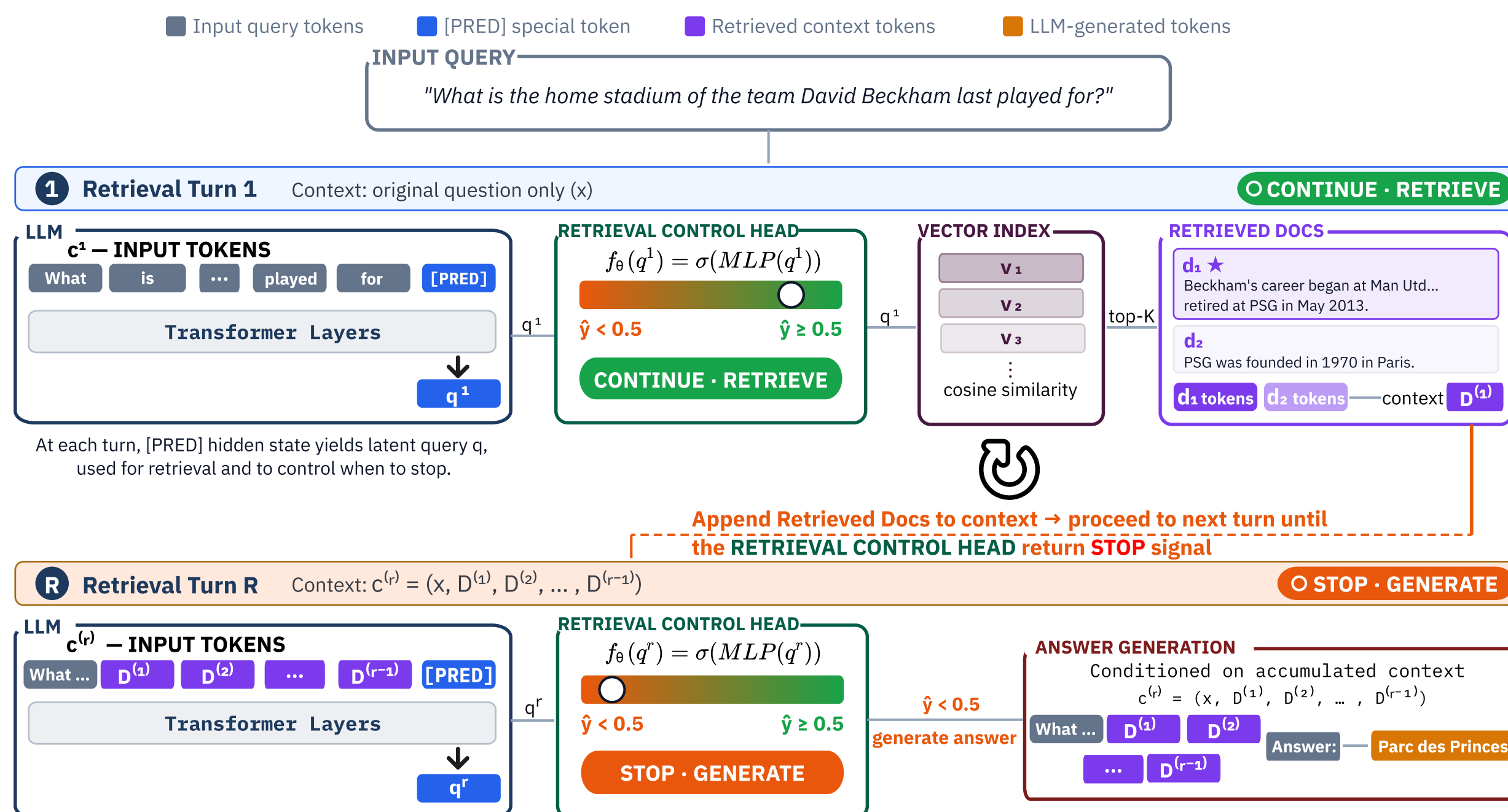


Figure 1. Queries are injected into the LLM and combined with a [PRE] token to form a latent query from hidden representations. This latent query is used for retrieval and to decide whether further retrieval is needed via a lightweight MLP Retrieval Control Head. The LLM then generates the answer from the retrieved context.

Introduction

Motivation. RAG mitigates hallucination by grounding LLMs in external evidence, but conventional pipelines (i) verbalize a natural-language query at every hop and (ii) enforce a hard architectural split between retriever and generator. This wastes the LLM's own representational capacity and adds latency from explicit token-level reasoning at each retrieval step.

Why not use LLMs' latent thoughts for retrieval and thinking?

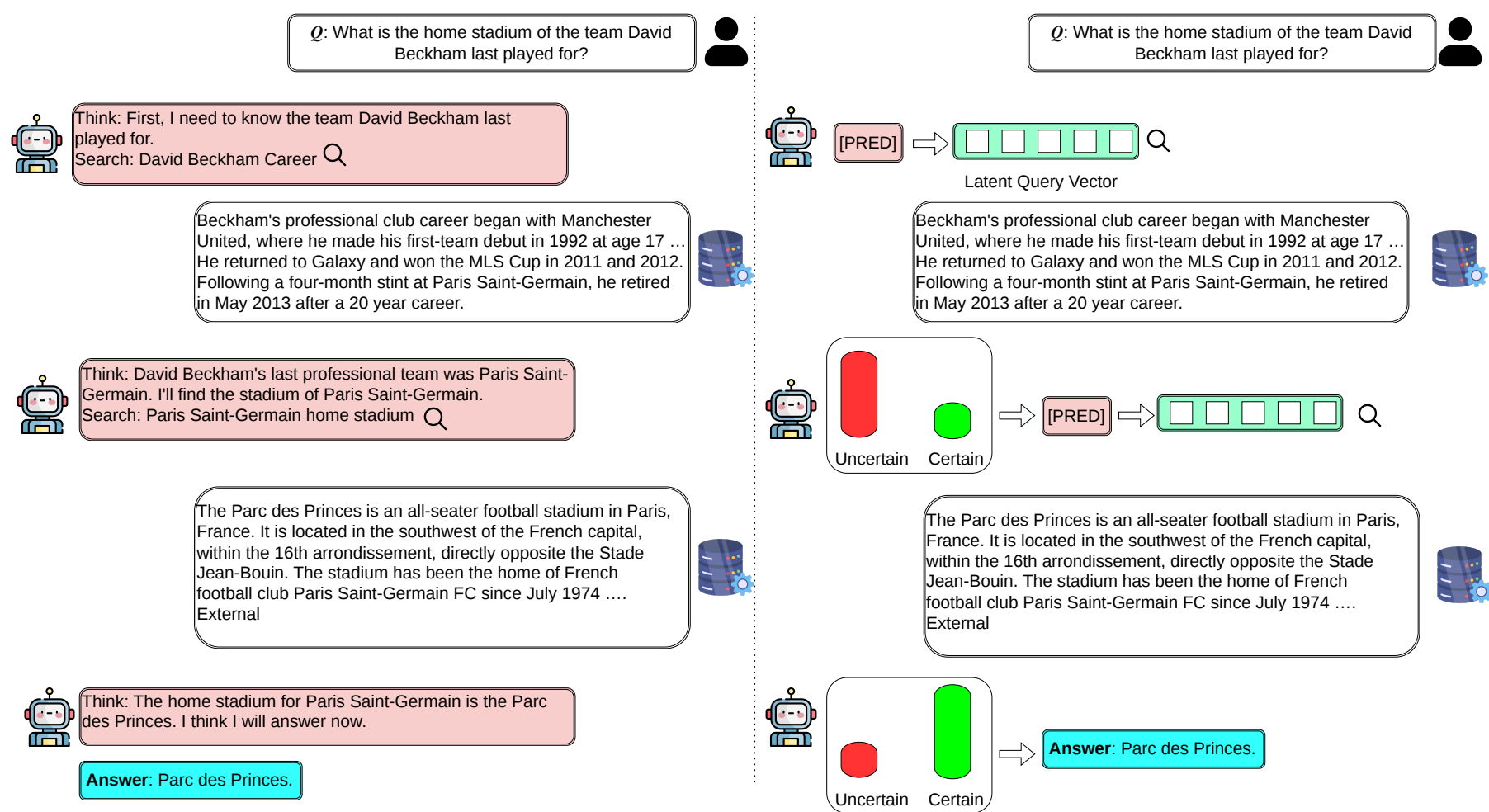


Figure 2. Conventional RAG (left) verbalizes every intermediate query and reasoning step. LAnR (right) reads a latent query from [PRE] and uses an MLP controller to decide when to stop, using no intermediate text.

Our objective. Enable a *single* LLM to jointly encode, retrieve, and generate entirely within its own latent space — using neither a separate retriever nor verbalized queries — while relying on its own hidden states to *adaptively* decide when sufficient evidence has been retrieved.

Contributions

- **Unified Latent RAG Framework** - jointly performs encoding, retrieval, and generation inside one LLM, removing the need for a separately trained dense retriever.
- **Implicit, Adaptive Retrieval Control** - a lightweight MLP head reads the LLM's hidden state to decide when retrieval is sufficient, with no explicit text-based reasoning.
- **Latent Query Construction** - retrieval query vectors are built directly from hidden representations, with no intermediate natural-language query.
- **Efficiency at No Accuracy Cost** - best average EM across 5 QA benchmarks with the **fewest retrieval calls** and $\sim 30\times$ fewer output tokens than prior methods.

Methodology

1. **Latent Query Construction.** Appending [PRE] to the input and reading its hidden state $q = h_{[PRE]} \in \mathbb{R}^d$ gives a query vector trained contrastively against document embeddings $d_i = h_{D_i}^{\text{last}}$ from the same model via last-token pooling.
2. **Multi-turn Retrieval.** At turn r , context $c^{(r)} = (x, D^{(1)}, \dots, D^{(r-1)})$ yields $q^{(r)}$; an adaptive target steers each turn toward *leftover* positives $P^{(r)} = P \setminus R^{(r-1)}$, avoiding redundant retrieval.
3. **MLP Retrieval Control Head.** $\hat{y}^{(r)} = \sigma(MLP(q^{(r)})) \in [0, 1]$ predicts CONTINUE (≥ 0.5) or STOP (< 0.5), self-supervised from whether all gold documents have been retrieved.
4. **Joint Objective.** $\mathcal{L} = \sum_{r=1}^R \mathcal{L}_{CL}^{(r)} + \lambda \mathcal{L}_{NTP} + \mu \mathcal{L}_{ctrl}$, balancing contrastive retrieval, next-token prediction, and control-head supervision.

Inference Efficiency

Because the retrieval signal is a latent vector rather than verbalized text, LAnR generates only ~ 5 output tokens per query - roughly $30\times$ fewer than Search-R1 (~ 163) or AutoRefine (~ 168) - for a $1.5\text{--}2.7\times$ wall-clock speedup, with no separate dense retriever to serve at inference time.

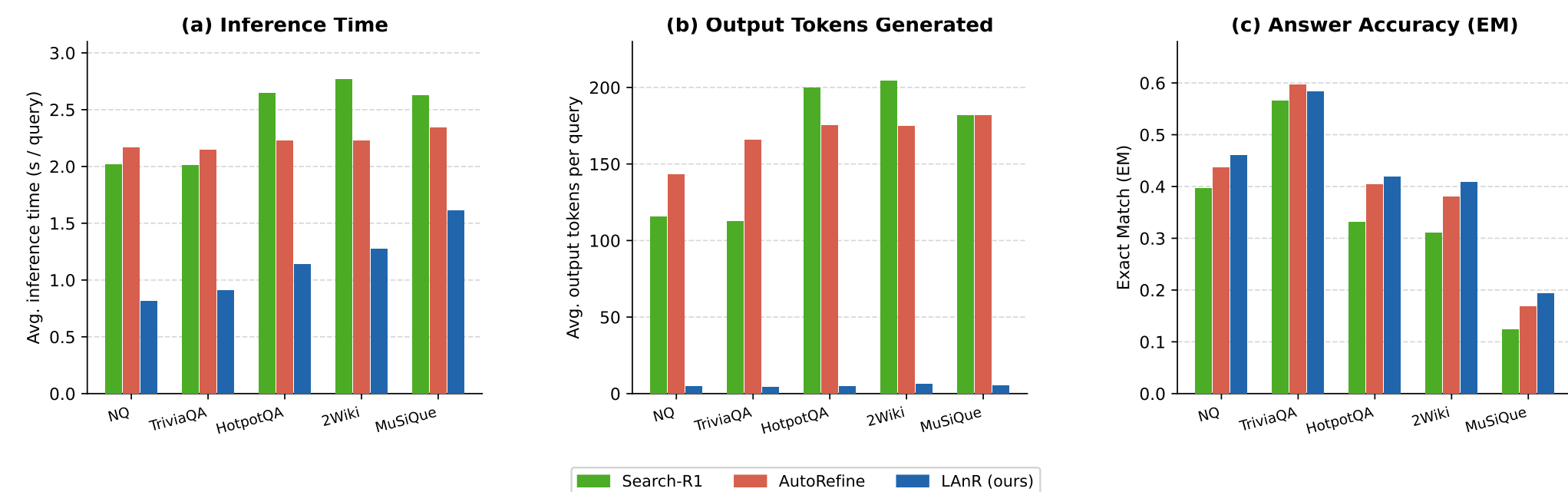


Figure 3. Comparison of inference time, generated tokens, and Exact Match accuracy between prior RAG methods and LAnR. Latent retrieval reduces latency and token generation while maintaining strong performance.

Experimental Results

We evaluate on five open-domain QA benchmarks: NQ, TriviaQA (single-hop) and HotpotQA, 2Wiki, MuSiQue (multi-hop), against retrieval-free, single-hop, and multi-hop agentic RAG baselines.

Methods	Single-Hop QA		Multi-Hop QA			
	NQ	TriviaQA	HotpotQA	2Wiki	Musique	Avg.
<i>w/o Retrieval</i>						
Direct Generation	0.106	0.288	0.149	0.244	0.020	0.134
SFT	0.249	0.292	0.186	0.248	0.044	0.176
<i>w/ Single-Hop Retrieval</i>						
Naive RAG [3]	0.348	0.544	0.255	0.226	0.047	0.270
<i>w/ Multi-Hop Retrieval</i>						
Search-o1 [4]	0.238	0.472	0.221	0.218	0.054	0.255
IRCoT [6]	0.111	0.312	0.164	0.171	0.067	0.181
ReSearch-Instruct [1]	0.365	0.571	0.351	0.272	0.095	0.331
ReSearch-Base [1]	0.427	0.597	0.305	0.272	0.074	0.319
Search-R1-Instruct [2]	0.397	0.565	0.331	0.310	0.124	0.336
Search-R1-Base [2]	0.421	0.583	0.297	0.274	0.066	0.312
AutoRefine-Instruct [5]	0.436	0.597	0.404	0.380	0.169	0.396
AutoRefine-Base [5]	0.467	0.620	0.405	0.393	0.157	0.405
LAnR-Instruct	<u>0.460</u>	<u>0.613</u>	0.419	0.408	0.193	0.418
LAnR-Base	0.455	0.610	<u>0.417</u>	<u>0.402</u>	<u>0.187</u>	<u>0.414</u>

Table 1. Accuracy comparison of LAnR and baselines on QA benchmarks. Bold and underline denote best and second-best results, respectively. "-Instruct" and "-Base" indicate the corresponding Qwen2.5-3B backbone variants.

Best average EM (0.418) — beating the strongest baseline (AutoRefine) by +14% relative EM on MuSiQue and +3.6% on 2Wiki.

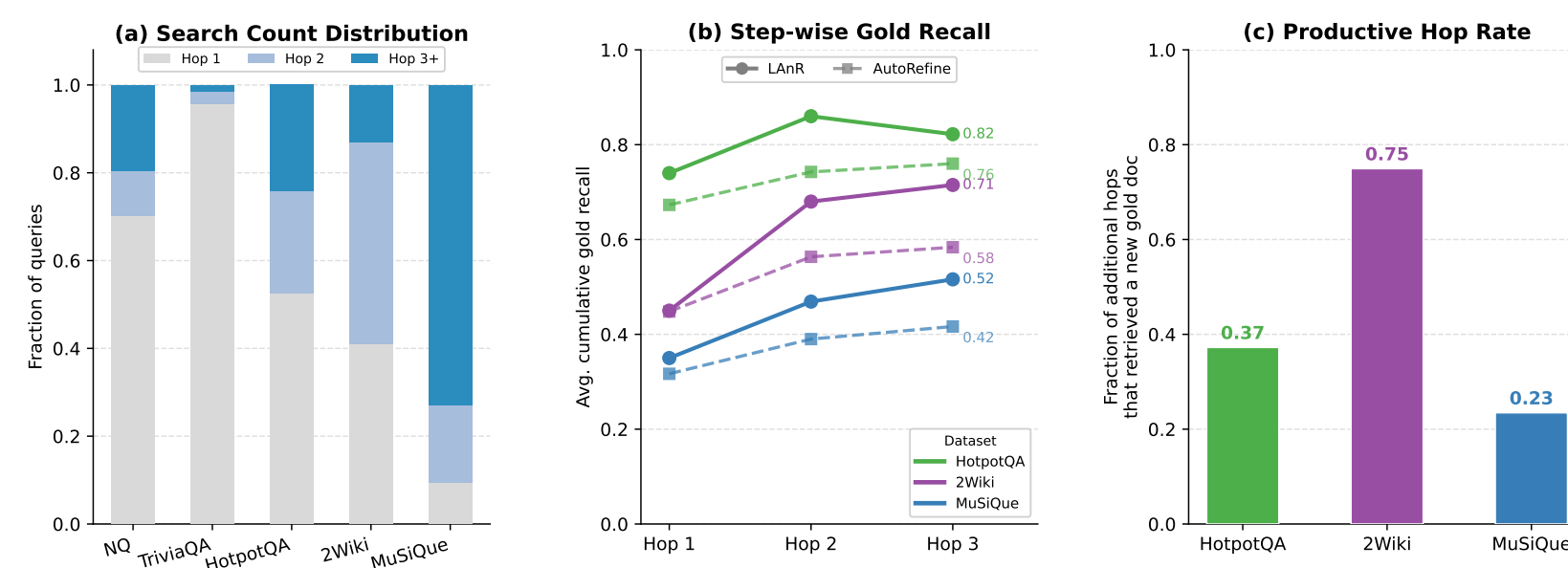


Figure 4. LAnR adaptive search behaviour. (a) Distribution of search counts per query across all five benchmarks. Single-hop datasets are dominated by a single search call, while compositional benchmarks require progressively more, showing the control head adapts to query complexity rather than applying a fixed budget. (b) Step-wise cumulative gold recall at each hop for LAnR (solid) and AutoRefine (dashed) on HotpotQA, 2Wiki, and MuSiQue. LAnR consistently outperforms AutoRefine at every hop on all three datasets. (c) Productive hop rate: fraction of additional search calls ($\geq$ Hop 2) that retrieved at least one new gold document per dataset.

Retrieval quality. On HotpotQA and 2Wiki, LAnR's retrieval recall matches specialized dense retrievers (BGE, E5) using only ~ 5 documents, remaining competitive on MuSiQue with ~ 7.9 documents, while issuing the **fewest retrieval calls** overall (avg. 1.54–2.47) with EM matching or exceeding AutoRefine and Search-R1.

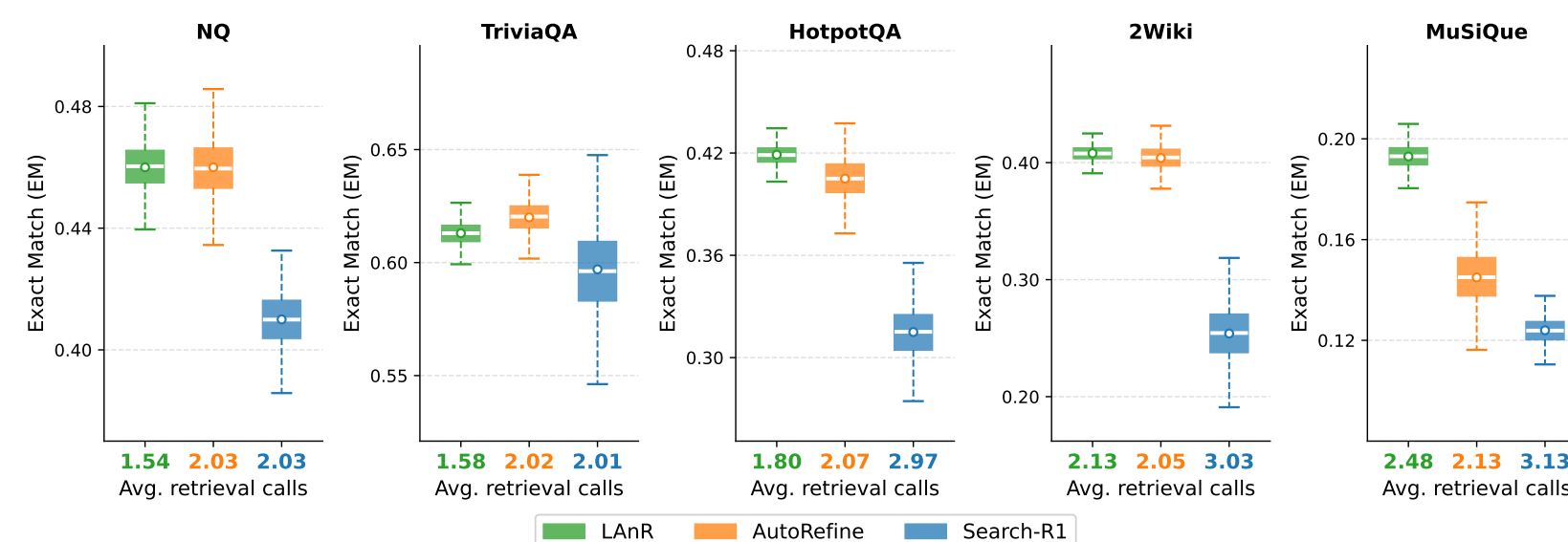
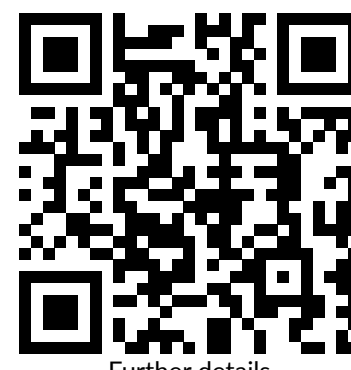


Figure 5. Per-dataset EM distributions for LAnR, AutoRefine, and Search-R1. LAnR achieves competitive or higher EM with the fewest retrieval calls and consistently narrow variance.

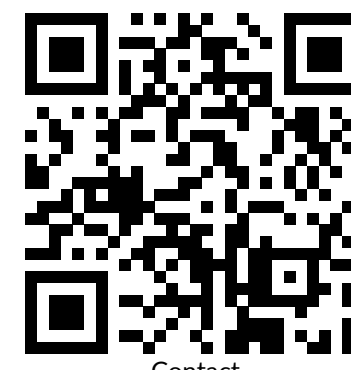
Takeaway

By replacing explicit query generation with compact latent vectors and an adaptive retrieval controller, LAnR performs **fewer but more effective** search calls, achieves **higher cumulative gold recall**, and maintains stable EM performance with **lower computational cost**.

Further details and Contact



Further details



Contact

Contact: Dung D. Le
Email: dung.ld@vinuni.edu.vn
Center for AI Research, VinUniversity
College of Engineering and Computer Science,
VinUniversity