Test-Time Strategies for More Efficient and Accurate Agentic RAG

Lightweight inference-time fixes to Search-R1 — no retraining required.

Abhinav Sharma, Brian Zhang, Deepti Guntur, Zhiyang Zuo, Shreyas Chaudhari, Wenlong Zhao

University of Massachusetts Amherst

Franck Dernoncourt, Puneet Mathur, Ryan A. Rossi, Nedim Lipka

Adobe Research

+5.6% Exact Match

+6.7% LLM Match

−10.5% Retrieval Turns

Overview

1. **Background**
   Agentic RAG and the Search-R1 framework
2. **Problem**
   Two recurring failure modes during inference
3. **Approach**
   Three test-time modules — no retraining
4. **Experiments**
   HotpotQA & Natural Questions, Qwen2.5-7B
5. **Results**
   Contextualization is the most effective
6. **Takeaways**
   Better evidence use beats retrieval diversity

Background: from RAG to Agentic RAG

Standard RAG

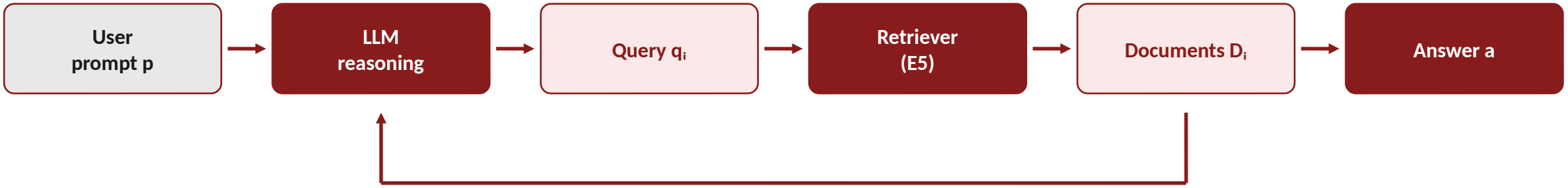
- Single-step retrieve → generate.
- Works for shallow questions where one hop of evidence suffices.
- Struggles with complex, multi-hop, or contextually layered questions.
- No reasoning loop — model can't ask follow-up queries based on what it found.

Agentic RAG

- LLM agent orchestrates retrieval — refines queries, decides when to stop.
- Reasoning and retrieval are interleaved across multiple turns.
- Search-R1 (Jin et al., 2025) trains LLMs with RL to issue search queries inside their reasoning trace.
- Up to 41% improvement over standard RAG on multi-hop QA.

This work targets agentic RAG, specifically the Search-R1 framework.

Search-R1 inference loop



At turn *i*: the LLM emits query q_i, the dense retriever (E5 over a 2018 Wikipedia dump) returns top-k passages D_i, and D_i is incorporated back into the trace. The loop runs until the model produces a final answer *a*.

Trained with reinforcement learning (PPO/GRPO).
Reward = exact match (EM) between predicted and ground-truth answer.

Two failures observed in Qwen2.5-7B Search-R1

① Information forgetting

The model repeatedly retrieves passages it has already seen in earlier turns.

Symptom: extra retrieval turns, more tokens consumed, higher latency — all without surfacing new evidence.

Diagnostic: *what if we block duplicates and force novel retrievals?*

② Ineffective evidence integration

Useful evidence is retrieved, but not effectively carried into the next reasoning step.

Symptom: the model issues follow-up queries instead of using what it already has — even when the answer is in the prior turn's documents.

Diagnostic: *what if we compress retrieved docs to task-relevant content and persist it across turns?*

Question: can lightweight, training-free changes at inference time fix both?

Three test-time modifications

All applied to a frozen Qwen2.5-7B Search-R1; no training, no architecture changes.

Contextualization

Compress evidence before reuse

External LLM extracts task-relevant info from D_i after each retrieval. Produces a concise summary D_i^* appended to a persistent memory cache, available at every later step.

Targets failure ②

De-duplication

Force novel retrievals

Maintains a set of doc IDs seen in earlier turns. If a top-k document was already retrieved, it is replaced by the next highest-ranked unseen passage — yielding D'_i.

Targets failure ①

Hybrid

Combine both, sequentially

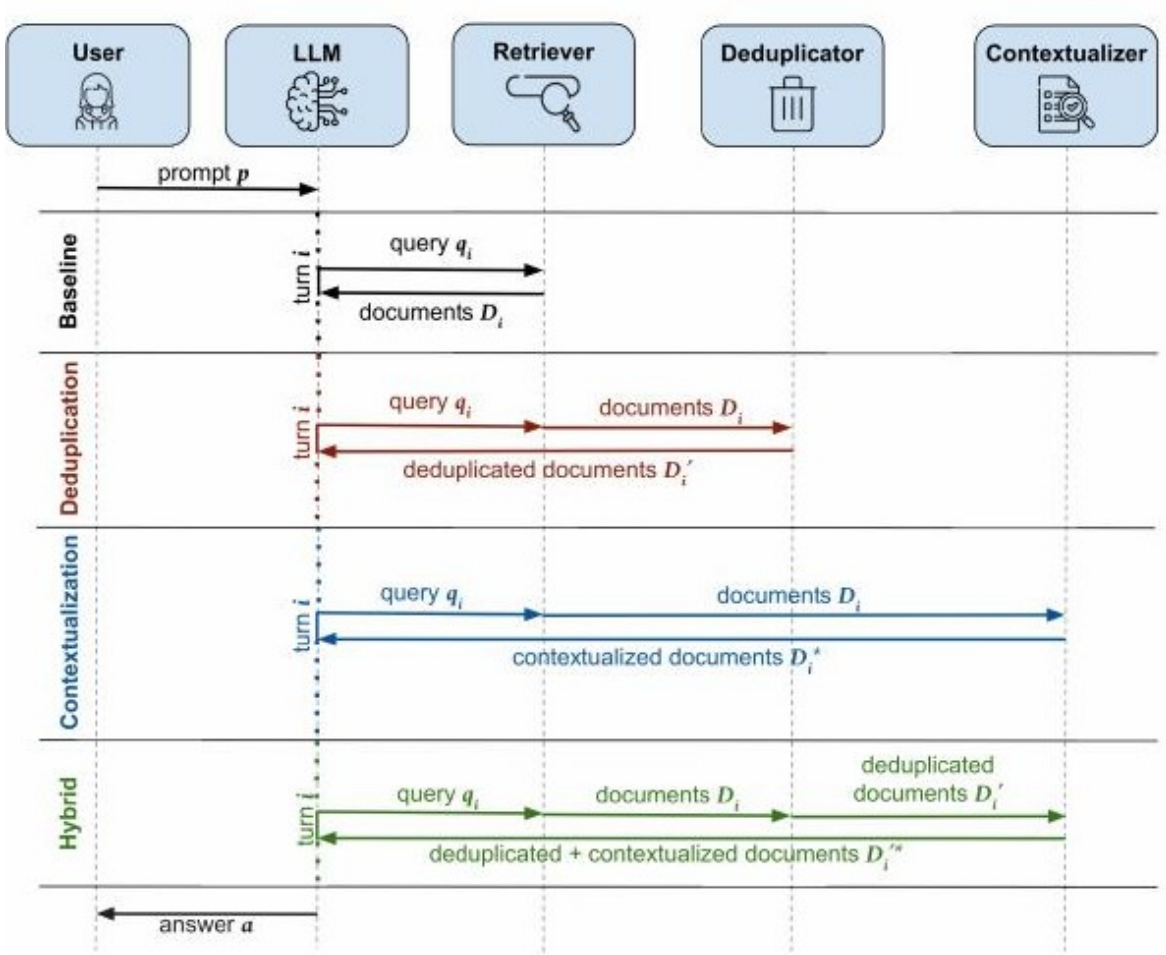
First de-duplicate the retrieved set, then contextualize the unseen documents into $D_i'^*$. Keeps novel evidence and compresses it for downstream reasoning.

Targets ① and ②

Each module wraps the retrieval call — the Search-R1 model itself is untouched.

Information flow per turn

Baseline vs. our three modules during a single inference turn i.



Baseline (Search-R1)
LLM → query q_i → retriever → D_i → appended to trace.

De-duplication
After retrieval, drop docs seen in earlier turns; pass D'_i instead.

Contextualization
Compress D_i to a task-relevant summary D_i^*; persist across turns.

Hybrid
De-duplicate first, then contextualize the unseen docs.

Module 1 — Contextualization

Mechanism

After each retrieval turn i:

- Pass the user prompt p and retrieved docs D_i to an external LLM (GPT-4.1-mini).
- Prompt the LLM to extract only the information useful for answering p, producing a concise summary D_i^*.
- Append D_i^* to a persistent memory cache shared across all subsequent reasoning steps.
- Search-R1's reasoning loop is otherwise unchanged.

Why this should help

- Reduces information forgetting — relevant context is retained even after later turns push earlier retrievals out of the active window.
- Reduces cognitive load — the model reasons over distilled evidence rather than raw passages.
- Decouples retrieval breadth from reasoning depth.

Cost / trade-off

- Adds an external LLM call per retrieval turn (latency, $).
- But fewer total retrieval turns offset some of that cost.

Module 2 — De-duplication

Mechanism

- Maintain a set of document IDs retrieved in earlier turns of the same prompt p.
- On each retrieval, scan the top-k results.
- If a result is already in the seen set, replace it with the next highest-ranked unseen passage.
- Pass the resulting set D'_i to the LLM as if it were the original retrieval output.

Both an intervention and a diagnostic

- If repeated retrieval reflects a real need to revisit the same evidence — blocking duplicates should hurt accuracy.
- If repeated retrieval is a failure to use prior evidence — blocking duplicates should leave accuracy roughly stable.

Spoiler: accuracy stays roughly stable, retrieval count goes up — the model just issues new queries to fill the gap.

Experimental setup

Datasets

- HotpotQA (Yang et al., 2018) — multi-hop QA
- Natural Questions (Kwiatkowski et al., 2019)
- 500 randomly sampled validation pairs from each
- No hyperparameter tuning on the eval subset

Backbone & retriever

- Qwen2.5-7B Search-R1-base (PPO) as primary model
- Smaller Qwen2.5-3B variants reported for comparison
- Dense retriever: E5 (Wang et al., 2022)
- Corpus: 2018 Wikipedia dump

Modules

- Contextualization & LLM-as-Judge: GPT-4.1-mini
- De-duplication: in-memory ID set per prompt
- Hybrid: de-duplication then contextualization
- Test-time only — no model updates

Metrics

- Exact Match (EM) — Search-R1's reward signal
- LLM Match — GPT-4.1-mini judges semantic equivalence
- Avg. # retrieval turns per question — efficiency proxy
- EM and LLM Match read together; turns is a proxy

Results — main table

All numbers on the 500-question evaluation set; bold = best variant.

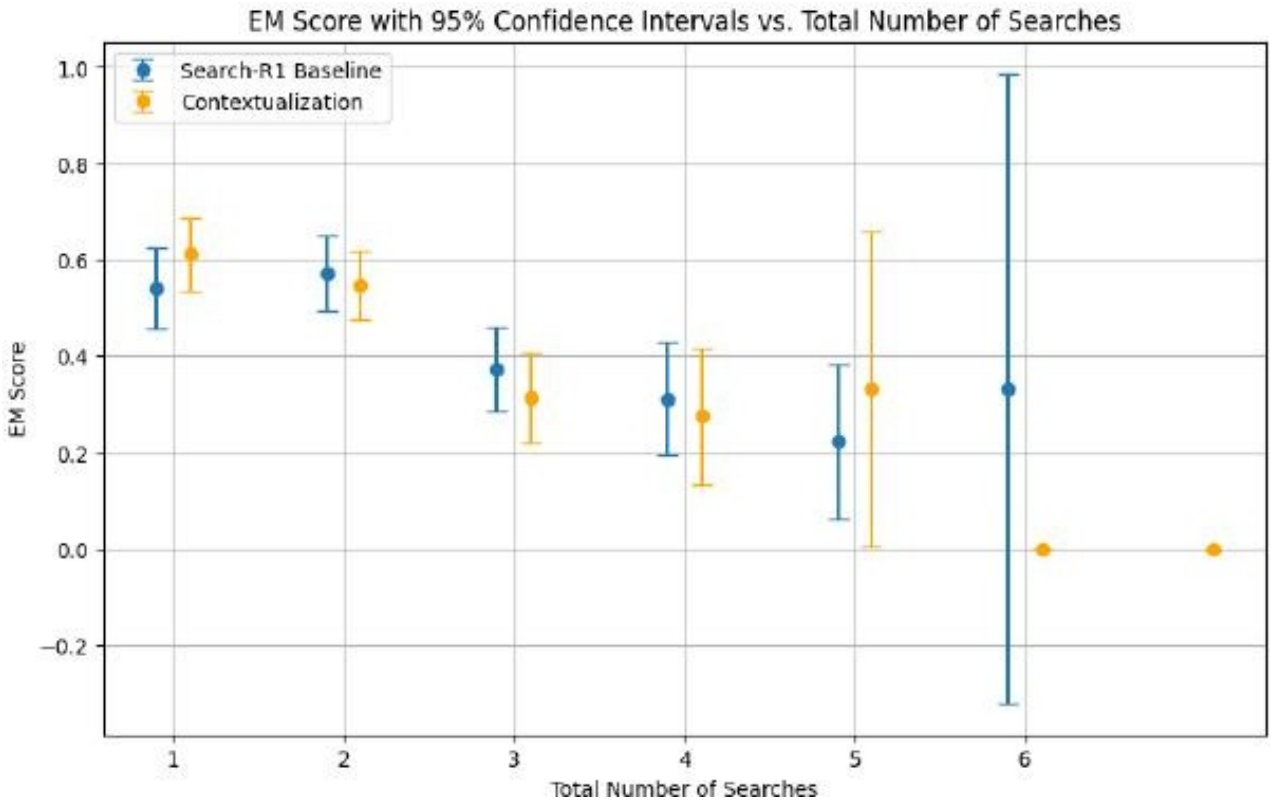
Variant	Exact Match	LLM Match	Avg. # Searches
Qwen2.5-3B base (PPO)	0.292	0.356	1.410
Qwen2.5-3B instruct (GRPO)	0.310	0.396	2.054
Qwen2.5-7B base (PPO)	0.464	0.538	2.392
+ Contextualization (Ours)	**0.490**	**0.574**	**2.142**
+ De-duplication (Ours)	0.478	0.560	2.498
+ Hybrid (Ours)	0.480	0.568	2.154

What to look at

- Contextualization wins on every metric — and it's the only method that simultaneously raises accuracy and lowers the search count.
- De-duplication trades accuracy gain for more searches (2.498 vs. baseline 2.392).
- Hybrid mostly tracks Contextualization — adding de-dup on top doesn't help much.
- LLM Match is consistently 16-18% above EM across all variants.

EM score vs. number of searches

Both methods trend downward — questions that need more turns are inherently harder.



Reading the chart

- x-axis: total retrieval calls used per question.
- y-axis: Exact Match score on those questions, with 95% CI.
- Blue = Search-R1 baseline. Yellow = + Contextualization.

Two takeaways

- Confidence intervals overlap at every search count — Contextualization's gain is not concentrated in any one regime.
- Both methods drop sharply after 2 searches — past that, the question itself is the bottleneck, not the strategy.

Conclusions

+5.6%
Exact Match
Contextualization vs. baseline

+6.7%
LLM Match
Contextualization vs. baseline

−10.5%
Retrieval Turns
2.142 vs. 2.392 average

1. **Better evidence use beats retrieval diversity.**
   De-duplication alone increases retrieval count without proportional accuracy gain — the dominant failure is ineffective use of evidence already retrieved.

2. **Contextualization is the only Pareto-improving intervention.**
   It's the single method that simultaneously raises both accuracy metrics and lowers the average retrieval count.

3. **Test-time wrappers are a cheap lever for agentic RAG.**
   Meaningful gains are achievable without retraining the agent or the retriever — useful where retraining is infeasible.

Limitations & future work

Limitations

- Two open-domain QA benchmarks; results may not generalize to other domains, long-document retrieval, or noisier corpora.
- Single backbone (Qwen2.5-7B) — smaller variants didn't reliably follow the Search-R1 output format.
- Contextualization and the LLM judge both rely on a proprietary model (GPT-4.1-mini), introducing variance and bias.
- Retrieval-turn count is a partial efficiency proxy — it does not capture the latency or token cost of the contextualization call.

Future directions

- Compare against fine-tuning the agent to produce its own contextualized memory (instead of an external LLM).
- Stronger efficiency metrics — wall-clock latency, token cost, $-per-question.
- Test on domain-specific QA, long-document settings, and stronger retrievers.
- Probe how the contextualizer's quality propagates to downstream answer accuracy.

Thank you — questions?