Plan*RAG: Efficient Test-Time Planning for Retrieval Augmented Generation

Prakhar Verma[1] Sukruta Prakash Midigeshi [2] Gaurav Sinha[1]
Arno Solin[1] Nagarajan Natarajan[2] Amit Sharma[2]

[1]Aalto University [2]Microsoft Research

TL;DR

- Plan*RAG externalizes reasoning as a DAG during inference to enable modular and efficient multi-hop retrieval.
- Enables parallelizable, grounded subqueries, better context window usage, and fine-grained control.
- Integrates seamlessly with existing RAG systems like Self-RAG and improves accuracy on multi-hop benchmarks.

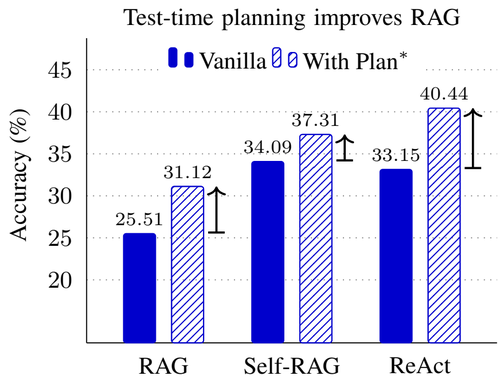


Figure 1: **Plan*RAG** improves performance on the HotpotQA benchmark substantially compared to various existing RAG methods, demonstrating the value of externalizing planning.

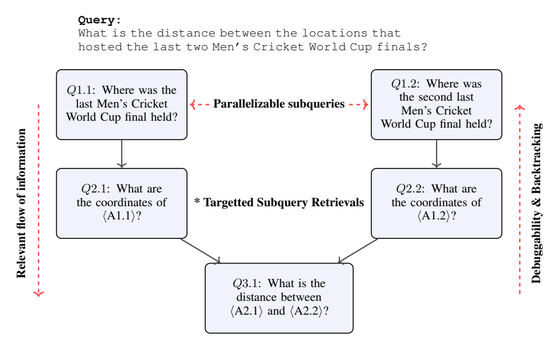


Figure 2: Reasoning DAG highlighting key advantages: only relevant information flows to each subquery, parallel subquery inference, and DAG allows for debugging & backtracking.

Motivation

Challenge: Multi-hop queries often fail in existing RAG systems due to:

- Fragmented reasoning within the LM's working memory
- Irrelevant or shallow document retrieval
- Context overflow

HotPotQA Query: *"Rumble Fish was a novel by the author of the coming-of-age novel published in what year by Viking Press?"*
Correct answer: *1967*

- **Vanilla RAG** generates an incorrect answer, *1975*

$$\mathbf{f}_{\text{LLM}} : Q \times \mathbf{D} \rightarrow G$$

- **Chain-of-Thought** infers wrong chain of thought and produces *1975*

$$\mathbf{f}_{\text{LLM}} : Q \times \mathbf{D} \times \bigoplus_{i=1}^{n} t_i \rightarrow G$$

- **Reason & React (ReAct)** despite using more test-time compute, fails to keep track of the original query intent and generates *1975*.

$$\mathbf{f}_{\text{LLM}} : Q \times \bigcup_{i=1}^{n} (t_i \times a_i \times o_i \times \mathbf{D}_i) \rightarrow G$$

- **Query Decomposition** despite explicitly decomposing the query fails to maintain coherent reasoning and generates *1975*

$$\mathbf{f}_{\text{LLM}} : Q \rightarrow \{q_i\}_{i=1}^{n}$$
$$\mathbf{f}_{\text{LLM}} : Q \times \mathbf{D} \times \bigcup_{i=1}^{n} (q_i, G_i) \rightarrow G$$

- **Self-RAG** relies on the LM's implicit reasoning capabilities within a single context window and generates *1975*

$$\mathbf{f}_{\text{LLM}} : Q \times \mathbf{D} \rightarrow \{(G_i, S_i)\}_{i=0}^{K}$$
$$G = \arg\max_{G_i} S_i$$

Plan*RAG: Test-Time Planning

Plan*RAG operates in two phases:

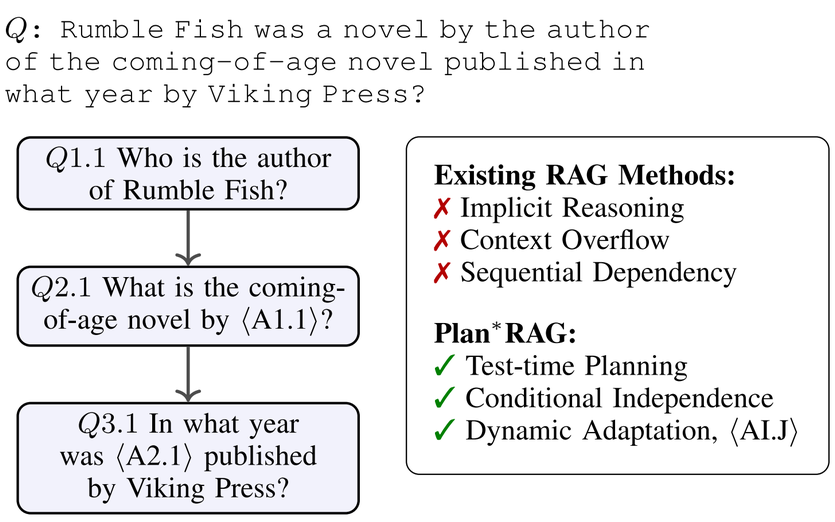
- decompose complex query into a external, reasoning plan as DAG

$$\mathbf{f}_{\text{LLM}} : Q \rightarrow \mathcal{G}(\mathbf{V}, \mathbf{E})$$

- generator LM traverse the DAG preserving dependencies

$$\mathbf{f}_{\text{LLM}} : G(\mathbf{Pa}(q)) \times q \times \mathbf{D} \rightarrow G(q) \quad \forall q \in \mathbf{V}$$

- Generates the correct answer: *1967*



Reasoning Plan: Properties & Benefits

- Atomic Subqueries Unlike sequential and local decomposition, Plan*RAG decomposes complex query into atomic subqueries resulting in precise retrievals and limiting hallucinations.
- Parallelization & Efficiency Plan*RAG captures conditional dependencies enabling parallel execution reducing latency

$$T^{\text{seq}} = \sum_{i=1}^{n} (t_{\text{gen}}(q_i) + t_{\text{ret}}(\mathbf{D}_i)) \,,$$

$$T^{\text{Plan*RAG}} = \sum_{d=0}^{D} \max_{q \in \mathbf{V}_d} (t_{\text{gen}}(q) + t_{\text{ret}}(\mathbf{D}_q)) \,,$$

where D is the DAG-depth and n is the number of sequential subqueries.

- Context Window Utilization Plan*RAG allows selective propagation of only parents information, rather than entire reasoning history providing a constant context window utilization,

$$C_t^{\text{seq}} = |Q| + \sum_{i=1}^{t} (|q_i| + |G(q_i)| + |\mathbf{D}|) \,,$$

$$C_q^{\text{ours}} = |q| + |\mathbf{D}_q| + \sum_{p \in \mathbf{Pa}(q)} (|p| + |G(p)|) \,.$$

- Debugging & Verification DAG enables systematic verification and granular control at the subquery level—errors can be isolated to specific nodes and automated interventions (like additional retrievals or alternative decompositions) can be targeted at specific reasoning steps

Discussion

- **Structured reasoning via DAGs:** Plan*RAG externalizes reasoning as a DAG, enabling systematic planning, parallel execution, and improved accuracy on multi-hop tasks — all while avoiding context overflow.
- **Modular and integratable:** Plan*RAG integrates cleanly with methods like Self-RAG and RQ-RAG, while maintaining comparable compute costs — making it a practical drop-in for existing RAG pipelines.
- **Unlocks new opportunities:** The external DAG enables dynamic plan refinement, backtracking, and extensions to tasks like fact-checking, math reasoning, and more.

More details...

See the paper:

arxiv.org/abs/2410.20753

References

[1] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *The Twelfth International Conference on Learning Representations*, 2023.

[2] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[3] C.-M. Chan, C. Xu, R. Yuan, H. Luo, W. Xue, Y. Guo, and J. Fu, "RQ-RAG: Learning to refine queries for retrieval augmented generation," in *First Conference on Language Modeling*, 2024.

prakhar.verma@aalto.fi amshar@microsoft.com