

Stronger Baselines for Retrieval-Augmented-Generation with Long-Context Language Models

Alex Laitenberger, Christopher D. Manning, Nelson F. Liu

Stanford University



At a Glance

Motivation

- Long-context LMs can process tens of thousands of tokens, how should RAG systems evolve?

What We Did

- Evaluated recent Multi-Stage RAG pipelines (RAPTOR, ReadAgent; [1, 2]) on long-context QA benchmarks
- Compared them with simpler Single-Stage baselines under equal token budgets

Key Finding

- Simple retrieve-then-read methods can **match or beat** complex pipelines with long-context LMs on QA

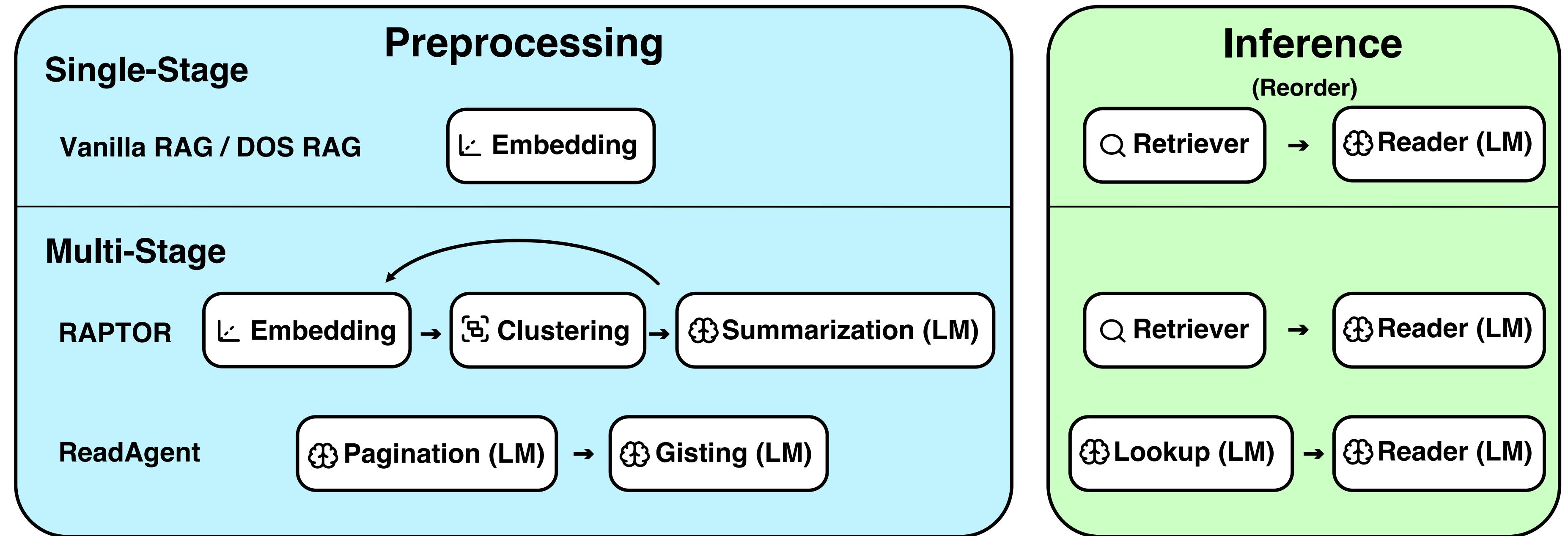


Figure 1. Comparison of Single-Stage baselines vs. Multi-Stage RAG methods.

Experiments

Question Answering Benchmarks

- ∞ Bench—229 multiple-choice questions on 58 documents (avg. 184K tokens)
- QuALITY—2K multiple-choice questions on 115 documents (2K-8K tokens)
- NarrativeQA—10K open questions on 355 documents (avg. 57K, up to 404K)

Setup

- Multi-Stage RAG pipeline methods: RAPTOR, ReadAgent
- Baselines: Vanilla RAG, DOS RAG, Full-Document; set up as ablations
- Retrieval via Snowflake Arctic embedding model [4]
- Token budgets systematically scaled, methods compared under equal budgets

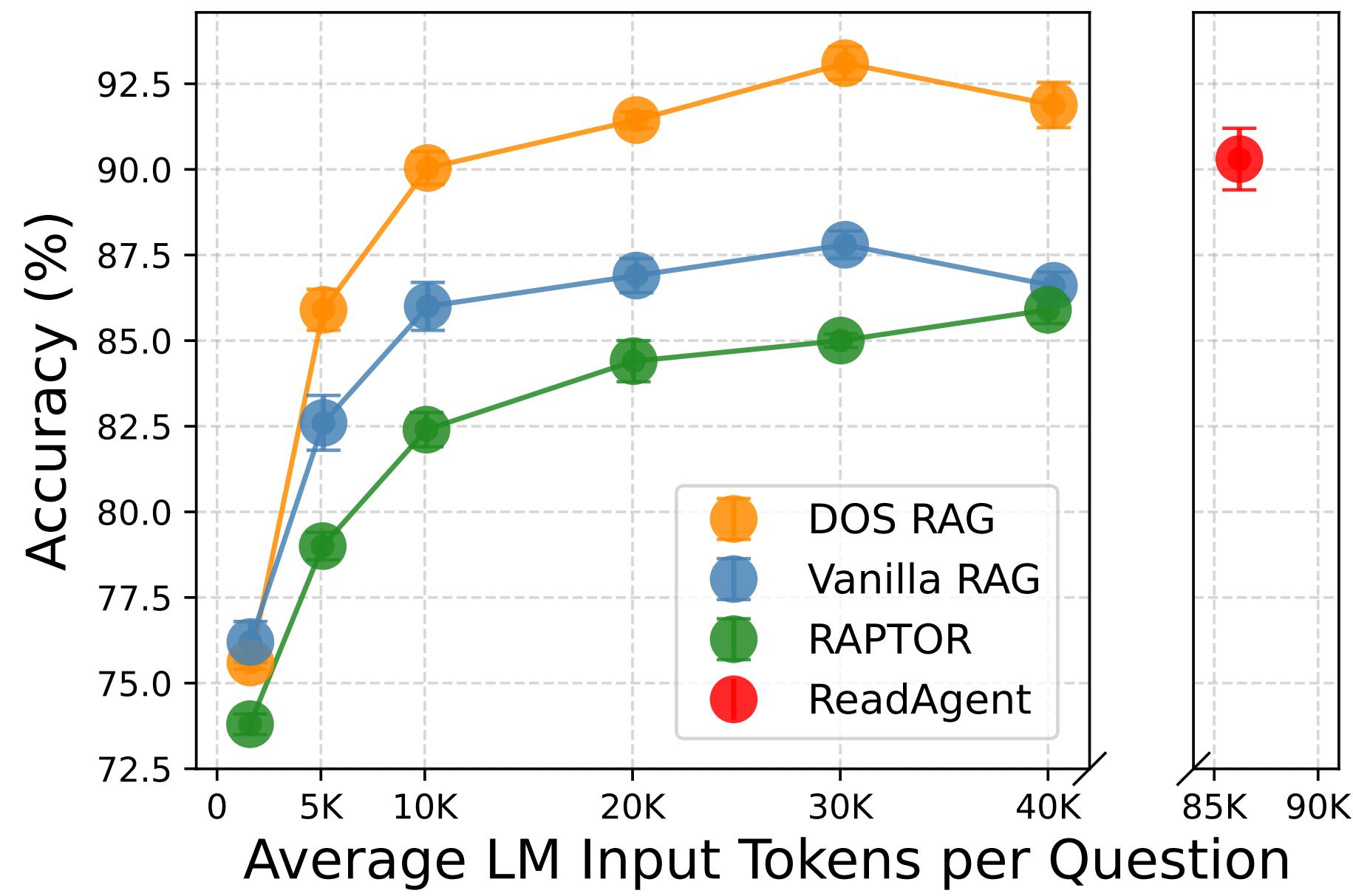


Figure 2. ∞ BenchEn.MC performance of various multi-stage RAG systems and long-context baselines with GPT-4o (mean \pm standard deviation over five runs).

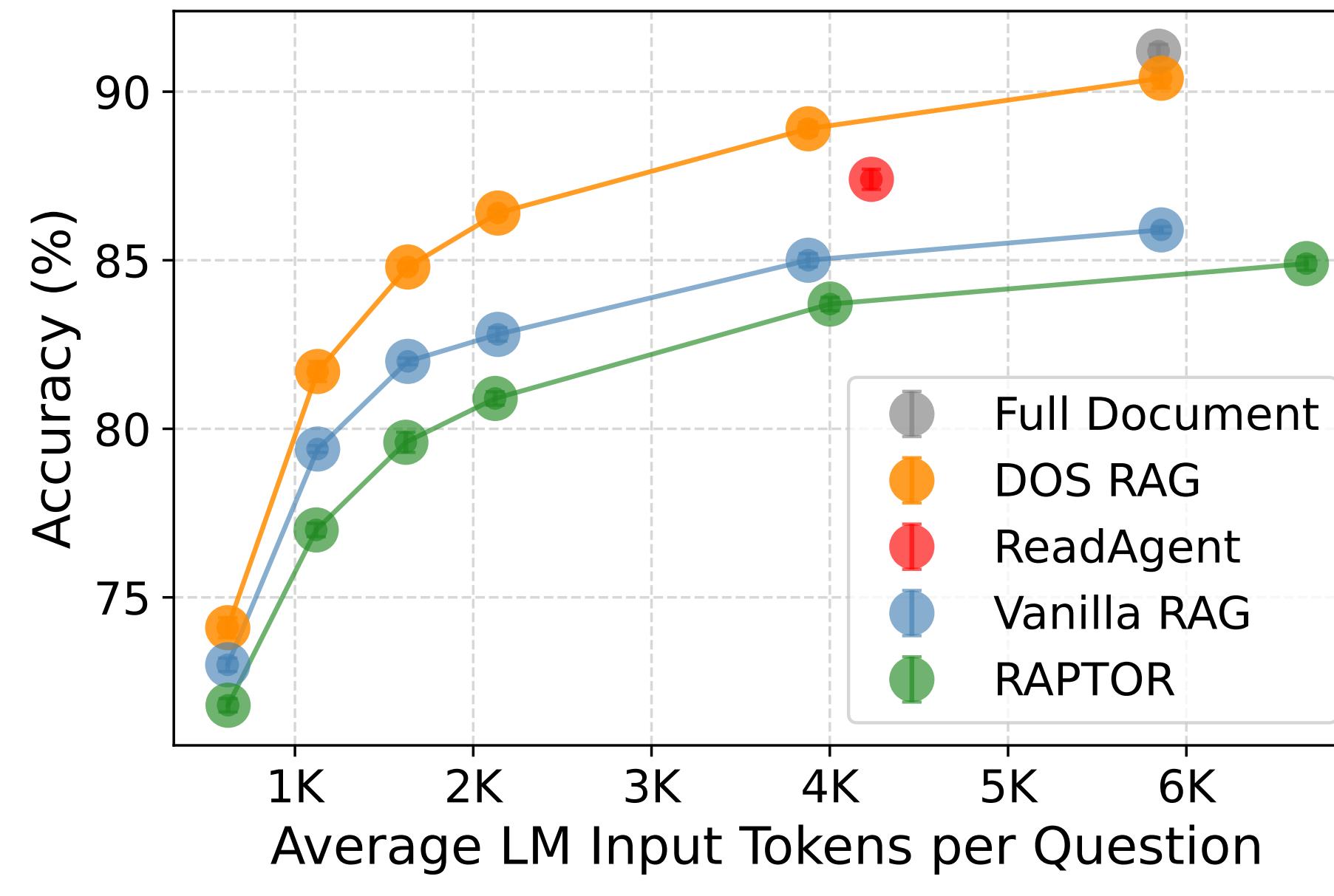


Figure 3. QuALITY performance of various multi-stage RAG systems and long-context baselines with GPT-4o (mean \pm standard deviation over five runs).

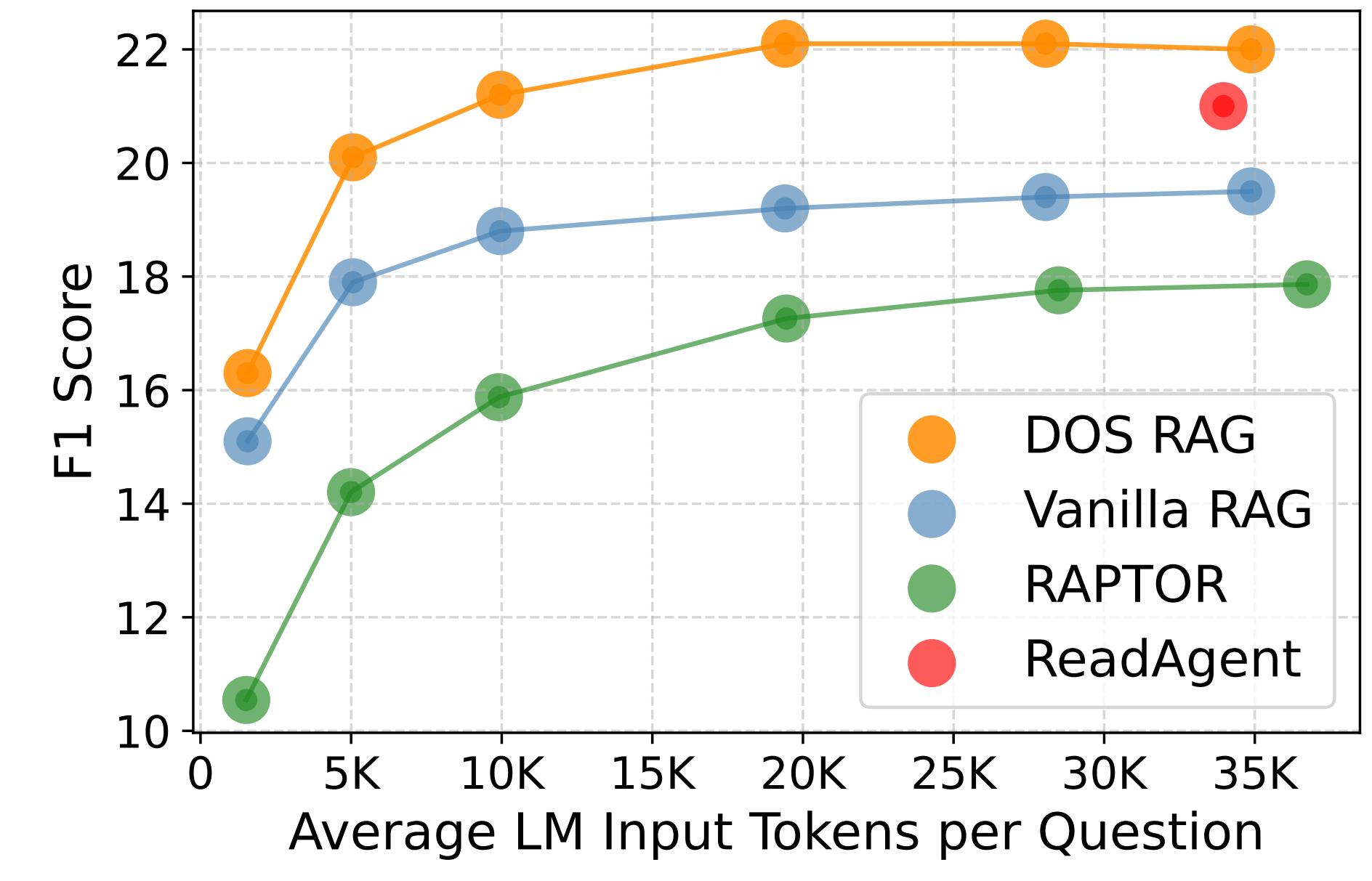


Figure 4. NarrativeQA performance of various multi-stage RAG systems and long-context baselines with GPT-4o-mini.

Analysis

Why is DOS RAG effective?

1. Retrieving from **original passages rather than generated summaries**, maintaining source fidelity and minimizing information loss.
2. Prioritizing **retrieval recall over precision**, ensuring critical information is included within the effective context window.
3. Reordering retrieved passages to **preserve original passage order**, which proved particularly beneficial when dealing with large retrieval budgets.
4. Favoring **simple over complex** pipelines, while leveraging strong embedding and language models for robustness.

Main References

- [1] Kuang-Huei Lee et al. A human-inspired reading agent with gist memory of very long contexts. In Proc. of ICML, 2024.
- [2] Parth Sarthi et al. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In Proc. of ICLR, 2024.
- [3] Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proc. of NeurIPS, 2020.
- [4] Luke Merrick. Embedding and clustering your data can improve contrastive pretraining, 2024. arXiv:2407.18887.

Conclusion

Result

- A simple retrieve-then-read baseline, such as DOS RAG, can match or beat recent complex Multi-Stage RAG pipelines, such as RAPTOR and ReadAgent

Recommendation

- Establish DOS RAG as simple yet strong baseline for future RAG evaluations
- Combine it with state-of-the-art embedding and language models with long-context capabilities
- Compare methods under matched LM token budgets

Limitations

Tasks

- multiple-choice and short-answer reading comprehension QA over single long documents
- open-domain benchmarks, no specialized knowledge (e.g., scientific, legal)

Models

- Reader: Proprietary OpenAI GPT-4o, GPT-4o-mini; no open-weight LMs
- Retriever: Snowflake Arctic embed (dense retrieval)