When One LLM Isn't Enough: How MA-RAG Uses a Team of Agents to Fix RAG's Biggest Problems

5 min read · Just now

R Rishikeshavlal Patel

Based on "MA-RAG: Multi-Agent Retrieval-Augmented Generation via Collaborative Chain-of-Thought Reasoning"

Link — https://arxiv.org/pdf/2505.20096

If you've played with modern chatbots, you've probably seen both sides of them:

- Ask a simple fact question → instant, correct answer.
- Ask something slightly messy like *"Who coached the team that beat Liverpool in the European Cup final where Jupp Heynckes played, and where was that match?"* → the model often guesses, confuses years, or just hallucinates.

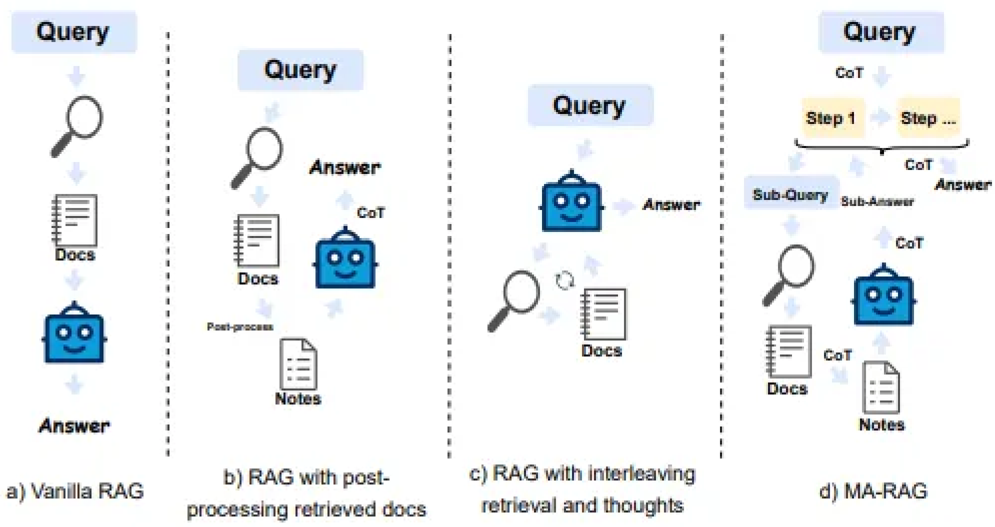
Traditional **Retrieval-Augmented Generation (RAG)** was invented to fix this: instead of relying only on whatever the model memorized during training, the system **retrieves documents** from an external knowledge base (like Wikipedia) and feeds them to the LLM to ground its answer in real evidence.

In practice though, real-world questions are:

- **Ambiguous** ("the final" — which year exactly?)
- **Multi-hop** (first find the year, then find the city, then maybe the stadium)
- Wrapped in **noisy retrieval** (lots of irrelevant or semi-relevant text in the retrieved chunks).

So a single LLM plus a flat list of documents still struggles.
The MA-RAG paper proposes a different angle: Don't treat RAG as a single black box. Treat it as a **team of specialized agents** that plan, retrieve, filter, and answer collaboratively.

In this article, I'll walk through what MA-RAG is, how this multi-agent setup works, why it beats many existing RAG systems, and what it means for the future of agentic AI.



A 30-Second Recap: What Is "Vanilla" RAG?

Classic RAG looks like this:

1. **You ask a question**
2. A **retriever** finds the top-k relevant documents from a corpus (e.g., Wikipedia)
3. The **LLM** gets a prompt containing your question + those documents
4. It generates an answer

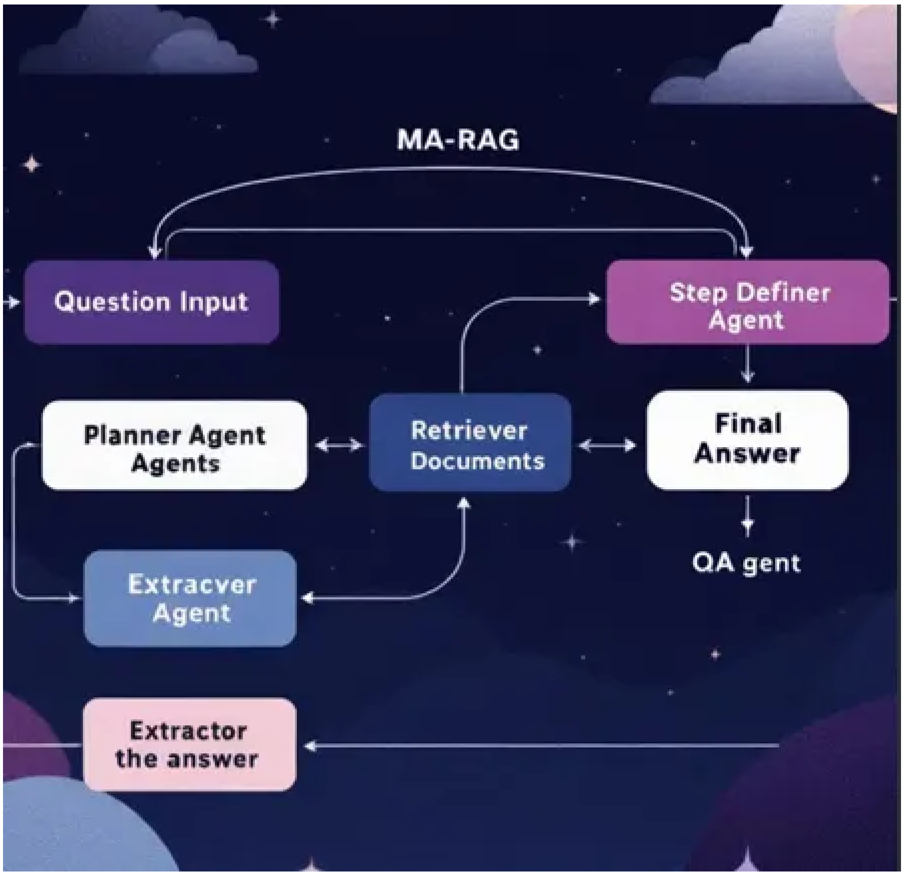
This setup already helps reduce hallucinations, but it has three big pain points for complex questions:

1. **One-shot retrieval —** The system gets one chance to retrieve documents. If the original query is vague or incomplete, retrieval may miss what we truly need.

2. **No real planning —** Multi-hop questions ("first find A, then use it to look up B") are handled in one big step. The LLM is expected to both *plan* and *reason* and *answer* in a single pass.

3. **Noisy, oversized context —** Retrieved documents are often long and partially irrelevant. Dumping everything into the context window can confuse the model and waste tokens.

MA-RAG doesn't just tweak the retriever. It **restructures the entire pipeline** into a modular, agent-based workflow that explicitly tackles ambiguity, planning, and noise.



Enter MA-RAG: Turning RAG Into a Multi-Agent Team

1. **Planner Agent** — *the strategist*
→ Reads the original question
→Detects ambiguity and complexity
→Breaks the question into a sequence of simpler **sub-tasks** (a "plan")

2. **Step Definer Agent** — *the query engineer*
→Takes one step from the plan at a time
→Turns that abstract step into a concrete, detailed **sub-query**, using the original question + previous answers as context.

3. Retrieval Tool + Extractor Agent — *the librarian & filter*
→ Retrieval tool fetches top-k documents (e.g., via FAISS dense retrieval).
→ **Extractor Agent** reads those and pulls out only **the sentences or spans that are actually relevant** to this step.
→ It outputs a small set of **clean, step-specific notes** instead of raw, noisy passages.

4. QA Agent — *the final storyteller*
→ Takes the step's sub-query + extracted notes.
→ Produces an answer for that step.
→ Step answers get passed forward and eventually combined into the final answer.

A Concrete Example: Jupp Heynckes and the European Cup Final

```
| Step | What the agent focuses on                                      | Example s
|------|----------------------------------------------------------------|----------
| 1    | Figure out **which final** we're talking about                 | "In which
|      | Planner + Step Definer + Retriever + Extractor + QA            | Answer: "
| 2    | Use that year to find the **location of the final**           | "Where wa
|      | Retriever + Extractor + QA                                     | Answer: "
| 3    | Combine everything into a final answer                         | "It was t
```

Let's go back to a slightly confusing football question:
"Where was the only European Cup Final in which Jupp Heynckes played held?"
MA-RAG instead turns this into a **mini-conversation between agents**. In the paper's example (also featured in my slides), the Planner creates this two-step plan:
Step 1: "Which year did Jupp Heynckes play in the European Cup Final?"
Step 2: "Where was the European Cup Final held in that year?"
Then the system runs:

- **Step Definer** sharpens Step 1 into a detailed search query
- **Retrieval Tool** gets documents about Jupp Heynckes and specific finals
- **Extractor** filters out everything except the part saying *he played in the 1977 European Cup Final*
- **QA Agent** answers: **1977**

Now Step 2 uses that answer:

- New sub-query: *"Where was the 1977 European Cup Final held?"*
- Retrieval + extraction find that the 1977 final was held in **Rome**
- QA Agent answers: **Rome**
  Putting it all together, MA-RAG returns the final answer:
  1977, in Rome.

Why MA-RAG Matters for the Future of Agentic AI

Stepping back, MA-RAG is interesting not only as "yet another RAG tweak," but as a **template for agentic systems:**

- It shows that you can **treat RAG as a reasoning pipeline**, not just an add-on to an LLM.
- It demonstrates that **small models + good orchestration** can beat larger models that work alone.
- It provides **interpretable intermediate artifacts** (plans, sub-queries, extracted notes) that humans can inspect for debugging and trust.

Final Thoughts

For me, the most compelling part of MA-RAG is not any single benchmark number, but the **shift in mindset**:

- From "let's cram more tokens into a bigger context window"
- To "let's *coordinate* smaller, focused reasoning steps across specialized agents"

As LLMs move from chatbots to **autonomous agents** solving real tasks — coding, research, analysis, planning — this style of design feels increasingly inevitable.
In my accompanying presentation and video walkthrough, I dive deeper into the actual agent prompts, example reasoning traces, and what it would look like to plug MA-RAG into a real application. But at a high level, the takeaway is simple:
One LLM is smart. A team of LLM agents, working together with a plan, can be much smarter.

AI · AI Agent · Agentic Ai · Multi Agent Ai · Retrieval Augmented Gen

Written by Rishikeshavlal Patel

0 followers · 1 following