



CS 394E: Contemporary Topics in LLMs

RAG at Scale and Multi-Agent Frameworks

Marco Canini

Professor of Computer Science, KAUST

Announcements

- Assignment 1 (fine-tuning) due tomorrow, Tue Oct 13, 23:59
- Assignment 3 (RAG) released tomorrow, due Oct 27, 23:59
 - Today's first half is its background
- Assignment 2 (agent loop) due Oct 20, 23:59
- No class next Monday, Oct 19: mid-semester break
- No midterm presentations: the midterm report (due Nov 5) carries the full 15%

Agenda

- RAG at scale and multi-agent frameworks: a short map (25 min)
- Paper: Retrieval-Augmented Generation (Lewis et al., NeurIPS 2020)
- Paper: AutoGen (Wu et al., COLM 2024)

RAG at Scale

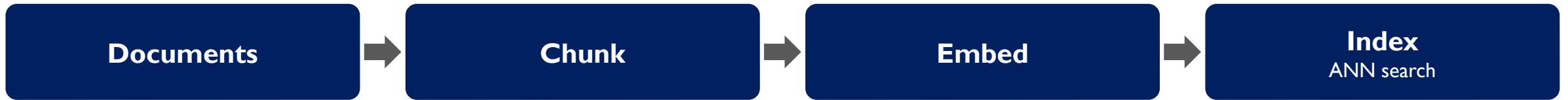
From the 2020 Paper to a 2026 RAG System

	Lewis et al., 2020	A typical system today
Retriever	DPR; query encoder trained end to end	Off-the-shelf embedding model, frozen
Generator	BART-large (400M), fine-tuned per task	General LLM, prompted, not fine-tuned
Fusion	Marginalize over the top-k passages	Paste the top-k chunks into the prompt
Corpus	21M Wikipedia passages, one fixed index	Private, changing documents; many indexes
Hard part	Training	The pipeline: chunking, indexing, ranking, freshness

The idea survived; the joint training mostly did not.

The RAG Pipeline

Offline: build the index



Online: answer a query



Errors compound: the LLM cannot use a chunk that was never retrieved.

Evaluate the stages separately: retrieval recall@k first, answer quality second.

From Document to Vectors

Fixed size

Museum Opens 9:00, Tuesday to Sunday.
Closes 18:00; last entry 17:30. Café
Ground floor, next to the shop. It closes at 22:00 on Fridays.

Cuts ignore meaning: “Café” lands in the museum chunk

By section

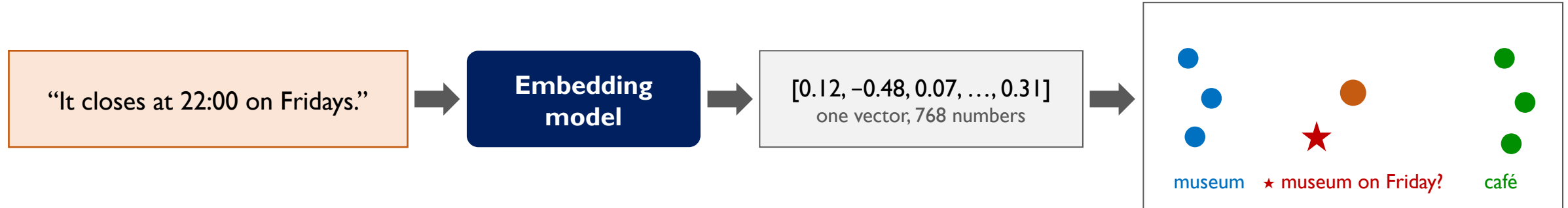
Museum Opens 9:00, Tuesday to Sunday. Closes 18:00; last entry 17:30.
Café Ground floor, next to the shop. It closes at 22:00 on Fridays.

Each chunk keeps its heading

Sliding window, with overlap

Museum Opens 9:00, Tuesday to Sunday. Closes 18:00; last entry 17:30.
Café Ground floor, next to the shop. It closes at 22:00 on Fridays.

Context survives a cut, at the cost of duplicated text



Cut from its heading, “It” could be the museum or the café: the vector lands in between, and a museum question can retrieve the café's hours.

Chunking and Embeddings Decide Recall

- **Chunk size:** too small loses context; too large dilutes the vector and spends prompt tokens
 - Typically a few hundred tokens, with some overlap
 - Split on structure (sections, paragraphs, tables), not character counts
- **Embedding model:** one vector per chunk (a bi-encoder)
 - Compare on public leaderboards (MTEB), then test on your own queries
- **Questions do not look like their answers**
 - Instruction-prefixed embeddings, query rewriting, HyDE (Gao et al., ACL 2023)
- **Changing the model means re-embedding the whole corpus**

Nearest-Neighbor Search at Scale

Exact search compares the query with every vector.

1B chunks $\times$ 768 dims $\times$ 4 bytes = **3 TB** of float32: not in RAM

Index	Idea	Trades away
IVF inverted file	k-means cells; scan only the few cells nearest the query	Recall at cell borders
HNSW hierarchical navigable small world	Layered neighbor graph; greedy walk toward the query	Memory for the links
PQ product quantization	Compress each vector to ~64 bytes (3 TB $\rightarrow$ 64 GB)	Distance precision

Every index is a point on a recall / latency / memory trade-off.

Faiss implements all three; vector databases add updates, metadata filters and replication on top.

Hybrid Search and Reranking



BM25 (1994): keyword scoring, TF-IDF refined: rare query words weigh more, repeats help with diminishing returns, long documents are normalized

Each retriever misses what the other finds: dense search blurs product codes and rare names; BM25 misses paraphrases

RRF: $\text{score}(d) = \sum 1 / (k + \text{rank}(d))$, with $k = 60$; no score calibration needed

A cross-encoder reads query and chunk together: more accurate, far too slow for the whole corpus, so it only reranks the top 50-100

Do We Still Need Retrieval with Long Context?

- Context windows went from 4K to 1M+ tokens. Why not paste everything in?
- Xu et al. (ICLR 2024), Llama 2 on long-context tasks:
 - 4K context + retrieval $\approx$ a model fine-tuned to 16K, at much lower cost
 - Retrieval still helps at 32K: retrieval-augmented Llama2-70B-32K beat GPT-3.5-turbo-16k on average over nine tasks
- **Cost:** every query pays for every token; prefill latency grows with the context
- **Quality:** models use long contexts unevenly ("lost in the middle", week 9)
- **Scale:** an organization's documents outgrow any window
- **Complements, not rivals:** retrieve to choose, long context to read

When to Retrieve, and Whether to Trust It

- Lewis et al. retrieve once, for every input
- Self-RAG (Asai et al., ICLR 2024): the model decides
 - Reflection tokens: Retrieve? · is the passage relevant? · is the claim supported? · is the answer useful?
- AutoGen's interactive retrieval: the assistant replies UPDATE CONTEXT to ask for more
 - The same decision, made inside a conversation (Yiting's talk)
- Agentic RAG: retrieval becomes one tool in an agent loop
 - This is where today's two halves meet
- Assignment 3: when should the system say "I don't know"?

Multi-Agent Frameworks

Composition Patterns: From One Call to Many

Pattern	Structure	What it buys
Chain of thought	One model reasons before it answers	Multi-step reasoning
Self-consistency	Sample many chains, majority vote	Robustness, at $N\times$ the cost
Debate	Agents answer, read each other, revise over rounds	Factuality and reasoning
Chain-of-verification	Draft $\rightarrow$ verification questions $\rightarrow$ answer them independently $\rightarrow$ revise	Fewer hallucinated facts
Role-based teams	Specialist agents + executor + critic	Division of labor, tools, code execution

A multi-agent system is a program: prompts are its instructions, conversation its control flow.

Who Decides What Happens Next?

- **Conversation-driven:** AutoGen; agents reply to each other, a manager LLM picks the next speaker
- **Graph / state machine:** LangGraph; the developer writes the control flow, LLMs fill the nodes
- **Roles and procedures:** MetaGPT, CrewAI; roles and handoffs fixed in advance

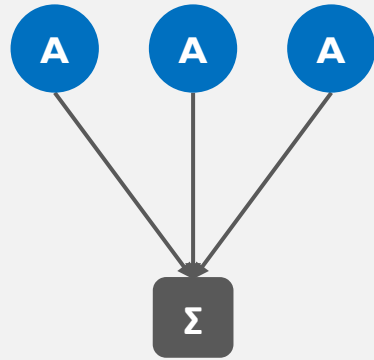
- The more control the LLM has, the more flexible the system and the less predictable
 - Production systems mostly pin the control flow down
- AutoGen itself kept moving: forked (AG2), rewritten (v0.4), now succeeded by Microsoft Agent Framework
 - Read the paper for the ideas, not the API

Five Ways to Coordinate Agents



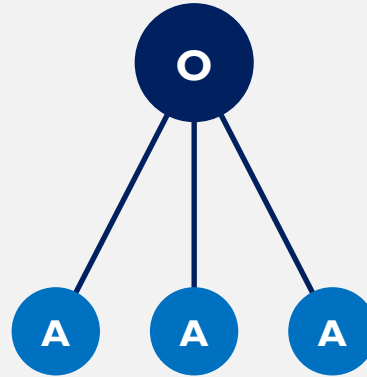
Single agent

one agent does every step, with all the tools



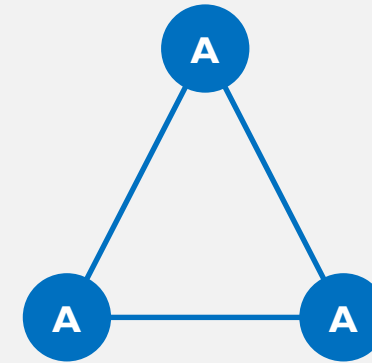
Independent

agents work in parallel, never talk; answers merged at the end



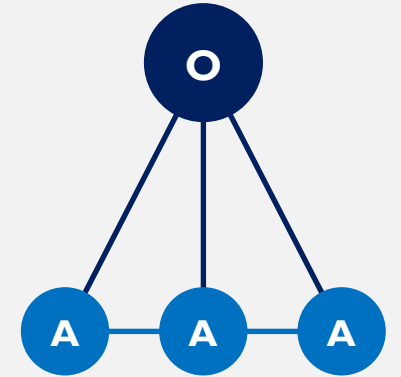
Centralized

an orchestrator delegates, checks and merges



Decentralized

peers talk to each other and reach consensus



Hybrid

an orchestrator plus peer-to-peer links

A agent O orchestrator Σ merge (e.g., vote or synthesis) lines: who talks to whom

Same tools, prompts and total token budget in every case: only the coordination changes.

Do More Agents Help?

Kim et al. (Google Research, Google DeepMind, MIT; arXiv, Dec 2025)

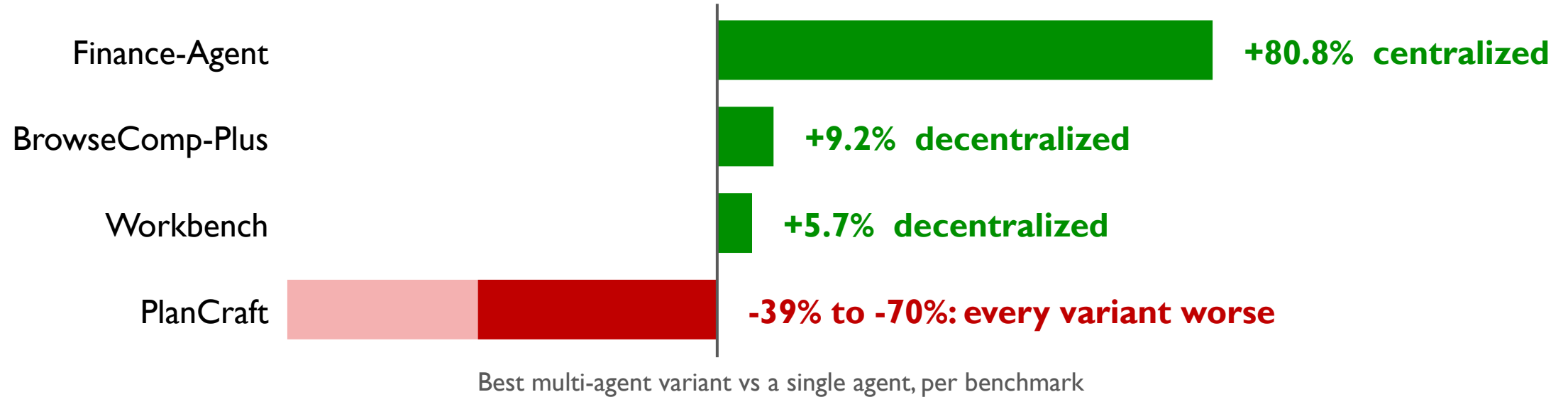
180 configurations: 5 architectures $\times$ 9 models $\times$ 4 agentic benchmarks

	Single agent	Independent	Decentralized	Centralized	Hybrid
Success rate	0.466	0.370	0.477	0.463	0.452
Token overhead	0%	58%	263%	285%	515%

On average, the best architecture gains one point over a single agent, at 3.6 $\times$ the tokens.

So the average is the wrong question. Which tasks gain, and which lose?

It Depends on the Task



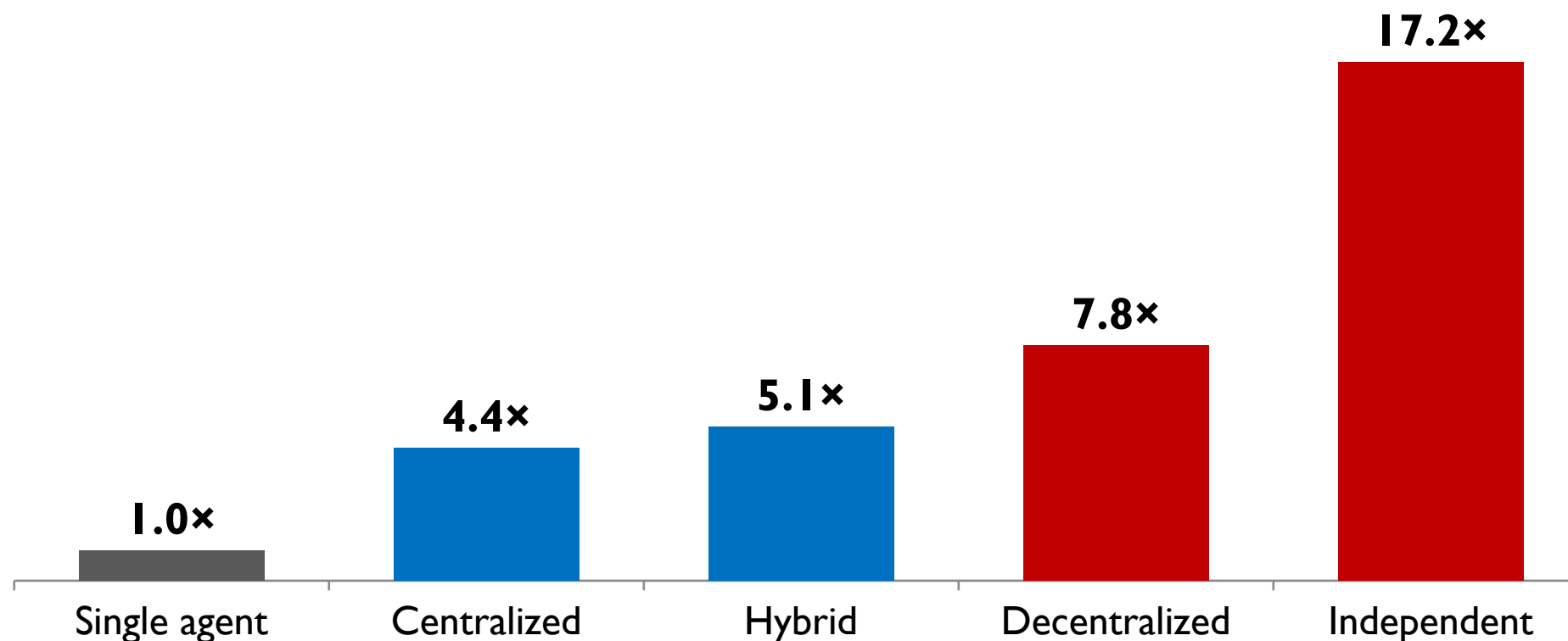
Finance-Agent: parallel subtasks

filings, news and operations analyzed independently, then merged by an orchestrator

PlanCraft: strictly sequential

find recipe → gather materials → craft; splitting the steps manufactures coordination

Errors Compound Without a Checker



A central orchestrator is a validation bottleneck: it catches sub-agents' errors.

Independent agents never check each other, so their errors flow straight into the answer.

Choosing an Architecture

- | | | | |
|---|---|---|--|
| 1 | A single agent already scores above ~45%? | → | Yes → stay single-agent |
| 2 | Does the task split into parallel subtasks? | → | No → stay single-agent |
| 3 | Is the task tool-heavy? | → | Yes → multi-agent only with care |
| 4 | Structured, decomposable work? | → | Yes → centralized (+80.8%, Finance-Agent) |
| 5 | Open-ended exploration? | → | Yes → decentralized (+9.2%, BrowseComp-Plus) |

Match the topology to the task: a single strong agent is the baseline to beat.

Not settled science. One study: four benchmarks, five fixed designs, one token budget. How to make many agents work well together is still an open question; week 9 starts from how they fail (MAST).

Next Time

- Oct 19: no class (mid-semester break)
- Oct 26: Agent Infrastructure, Memory, and Observability
 - Shared: Why Do Multi-Agent LLM Systems Fail? (Cemri et al., NeurIPS 2025)
 - Track A: MemGPT (Packer et al., 2023)
 - Track B: Recursive Language Models (Zhang, Kraska & Khattab, 2025)