CS6216 Advanced Topics in Machine Learning (Systems)

Application systems: server design, AI agents and RAGs

Yao LU

23 Oct 2024

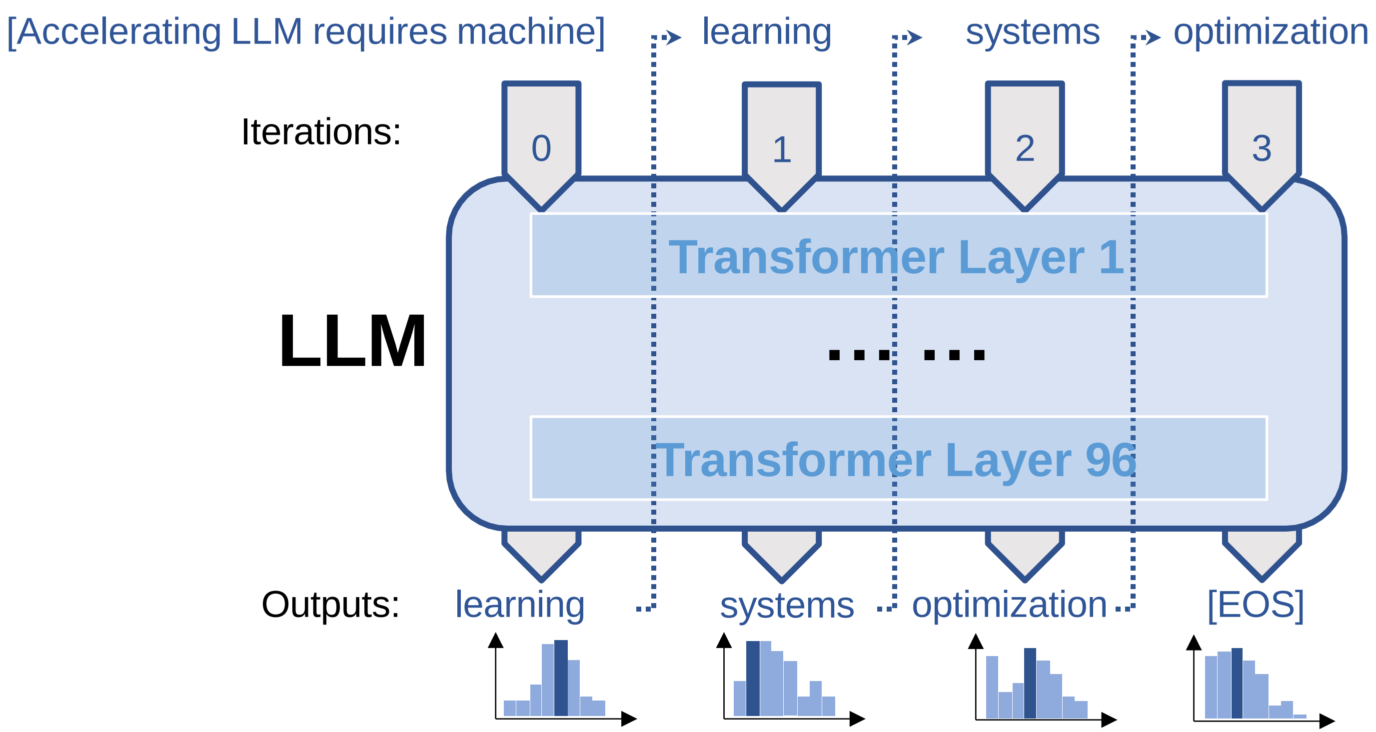
National University of Singapore
School of Computing

Application systems: outline

- Server design
- Retrieval Augmented Generation (RAG)
- AI agents

Recap: HW3 LLM incremental decoding

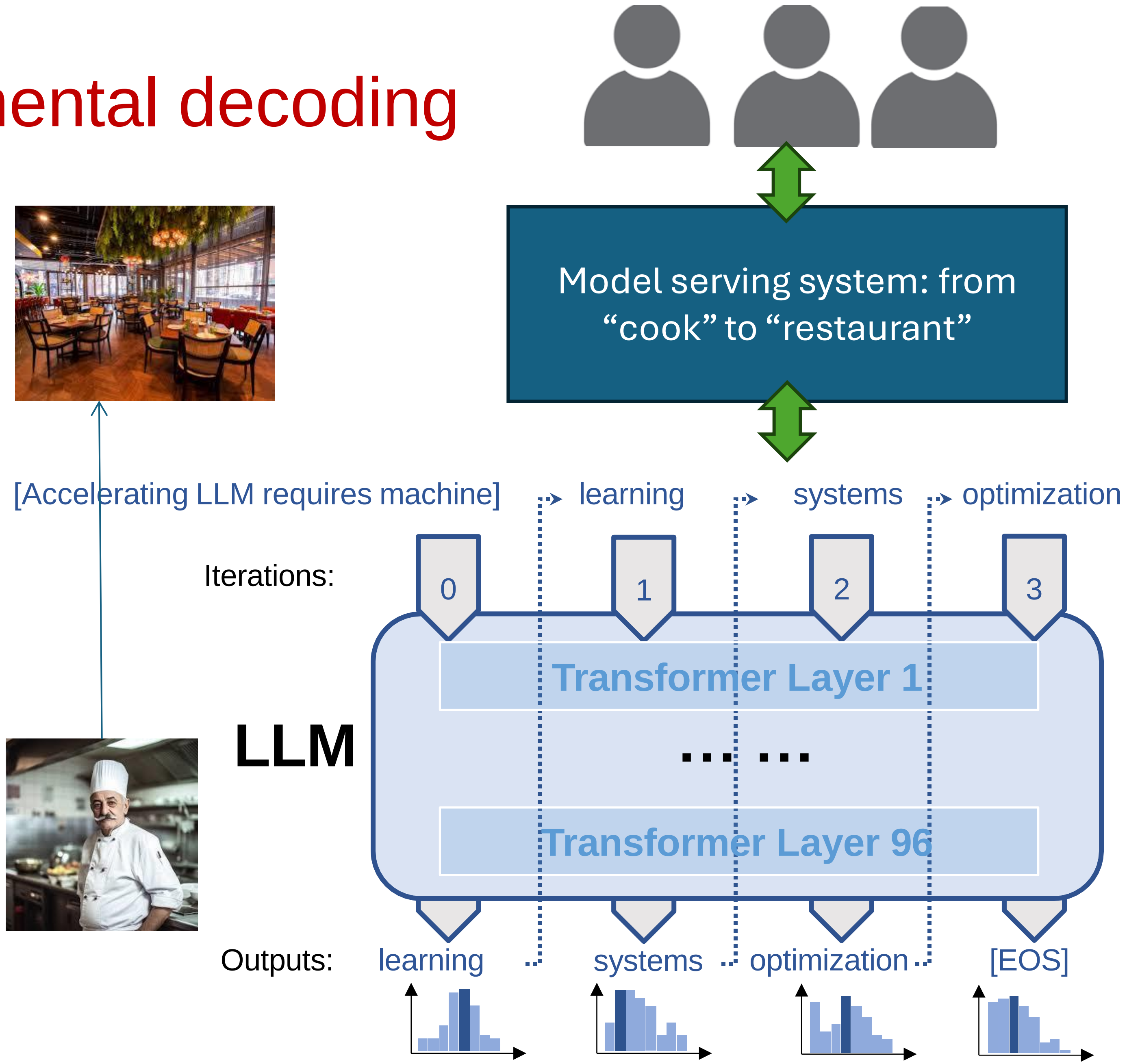
- **What's happening**
  - KV cache initialization & loading
  - Model forward propagation
  - Decoding algorithm
  - Stopping criterion
  - Tokenizer



Recap: HW3 LLM incremental decoding

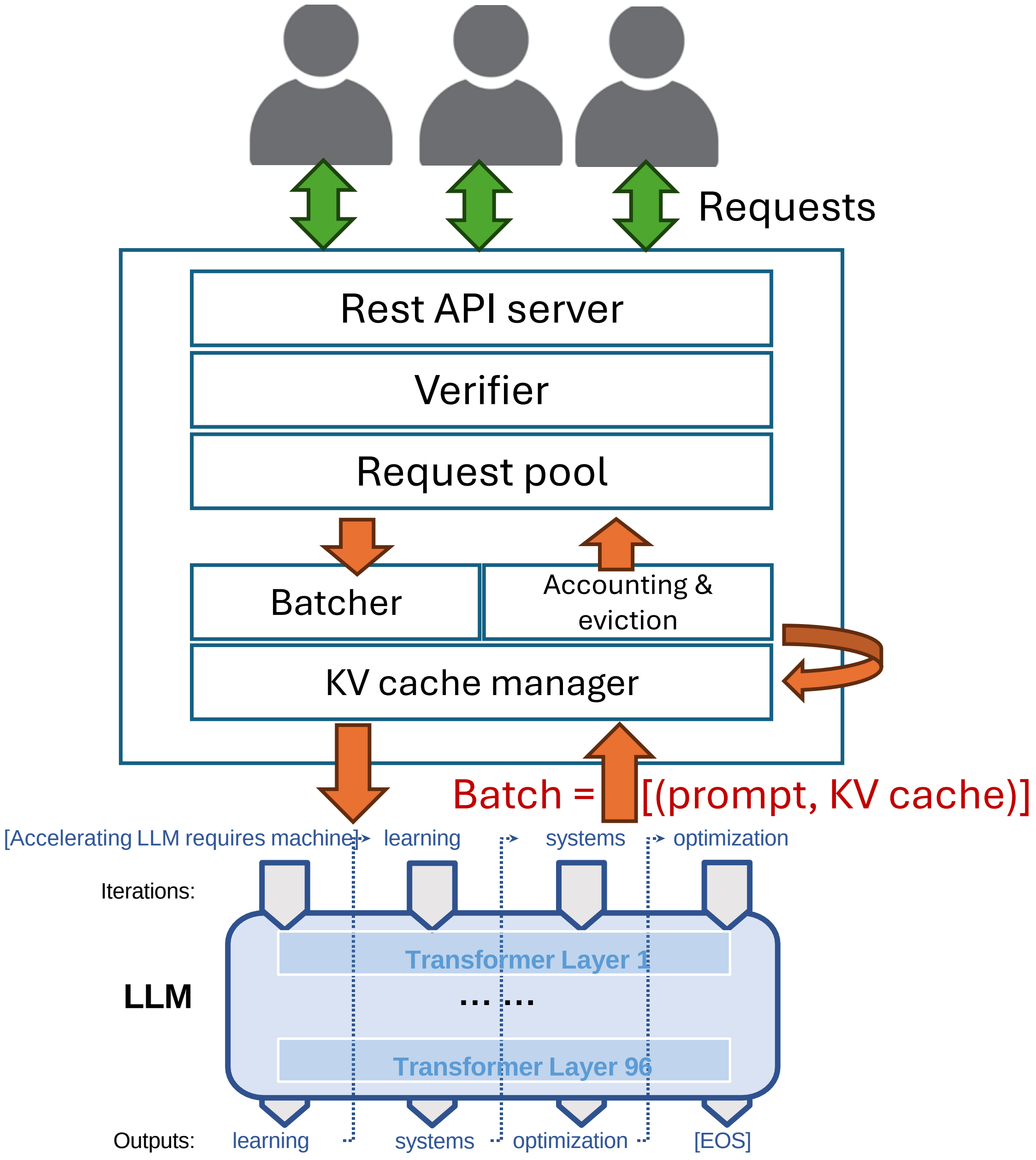
- **What's happening**
  - KV cache initialization & loading
  - Model forward propagation
  - Decoding algorithm
  - Stopping criterion
  - Tokenizer

- **What's missing**
  - API server
  - Queueing & batching
  - Accounting & perf stats



Server workflow

- **Rest API server** Handles LLM and control requests.
- **Verifier** checks if models, user access (authorization), and generation parameters are valid. Handle errors.
- **Request pool** stores generation requests, their states and outputs.
- **Batcher** assembles the next batch to compute by fetching new or on-going queries from the pool. Note: the amount that can be processed in a batch can be less than the request pool.
- **Accounting & eviction** counts tokens and removes finished, force stops (due to user interrupt, low account balance etc.) queries.
- **KV cache manager** initializes and maintains the kv cache to use for each query, offload or evict if necessary.



A typical LLM server

Batching and eviction logics

- A two-stage solution (solution from TGI, other solutions exist)

New queries

[Req 1, Prompt 1, KV cache 1, state_new]
[Req 2, Prompt 2, KV cache 2, state_new]
[Req 3, Prompt 3, KV cache 3, state_decode]
[Req 4, Prompt 4, KV cache 4, state_decode]
....
[Req k, Prompt k, KV cache k, state_decode]

Request pool
Taking which? FIFO or by $$

[Req 1, Prompt 1, KV cache 1, state_new]
[Req 2, Prompt 2, KV cache 2, state_new]
[Req k, Prompt k, KV cache k, state_decode]

Input batch – Step 1
Pick new queries, initialize KV cache, send for prefill

[Req 1, Prompt 1 + 1 tk, KV cache 1, state_decode]
[Req 2, Prompt 2 + 1 tk, KV cache 2, state_decode]
[Req k, Prompt k, KV cache k, state_decode]

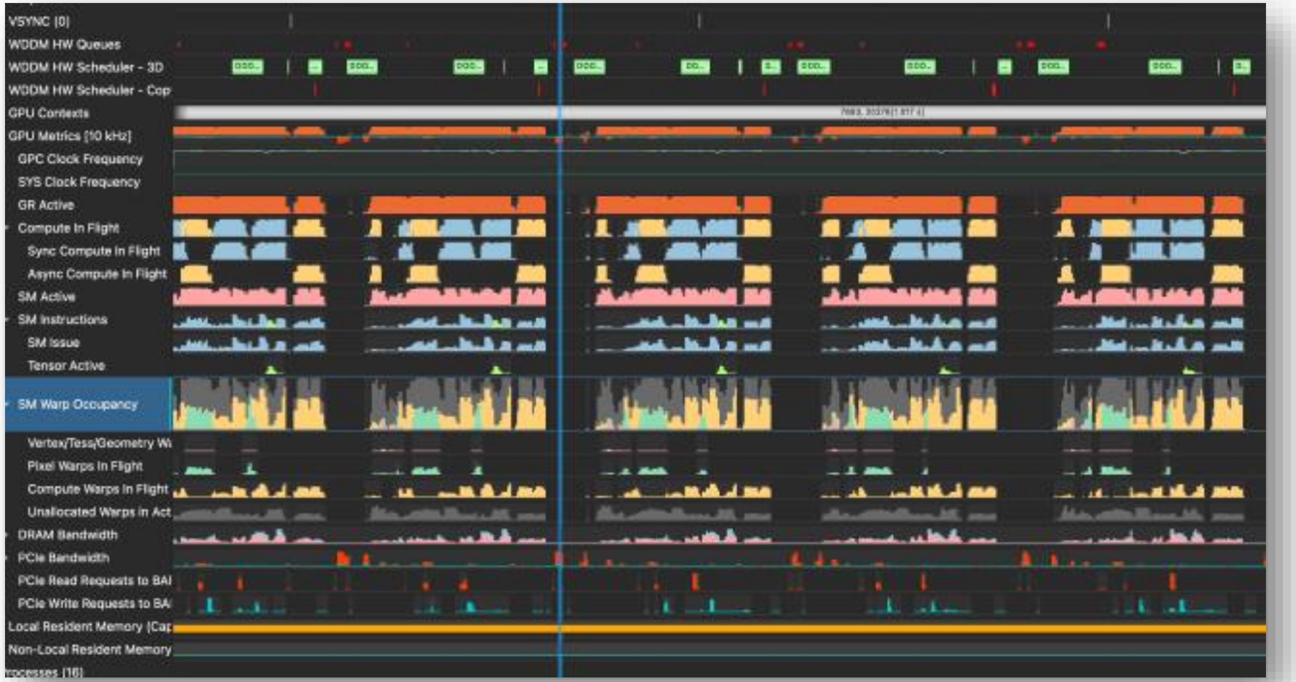
Input batch – Step 2 decode

[Req 1, Prompt 1 + 1 tk, KV cache 1, state_decode]
[Req 2, Prompt 2 + 1 tk, KV cache 2, state_decode]
[Req k, Prompt k + 1 tk, KV cache k, state_finish]

Input batch – Step 3 eviction, update pool

Other useful modules

- **Session management**
  - In chat applications, some conversations last very long. This results in KV cache getting "stuck" in GPU memory.
  - Offload KV cache to lower tiers of memory/storage. Simple solutions include FIFO, LRU-k etc. Some better solutions such as InfiniGen, OSDI 2024.
- **Perf stats and profiling**
  - Prometheus for systems monitoring
  - Nvidia nsight for GPU perf. profiling
- **Debugging and testing**
  - So far, hand crafted solutions only
  - Some simple sanity checks:
    - Unit tests, logits comparisons
    - "Secret" testing prompts



API server

- REST APIs with the OpenAI standard
  - Stream, complete, chat, assistant APIs
  - Platform & runtime independent

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
     "model": "gpt-4o-mini",
     "messages": [{"role": "user", "content": "Say this is a test!"}],
     "temperature": 0.7
   }'
```

- Easy-to-use, off-the-shelf Python libraries: Uvicorn, FastAPI
- High performance REST servers available: Actix-web (Rust)

- Good ones can be 10x faster, but API server is a small overhead, relatively
- **Geo-distributed? Highly available systems?**

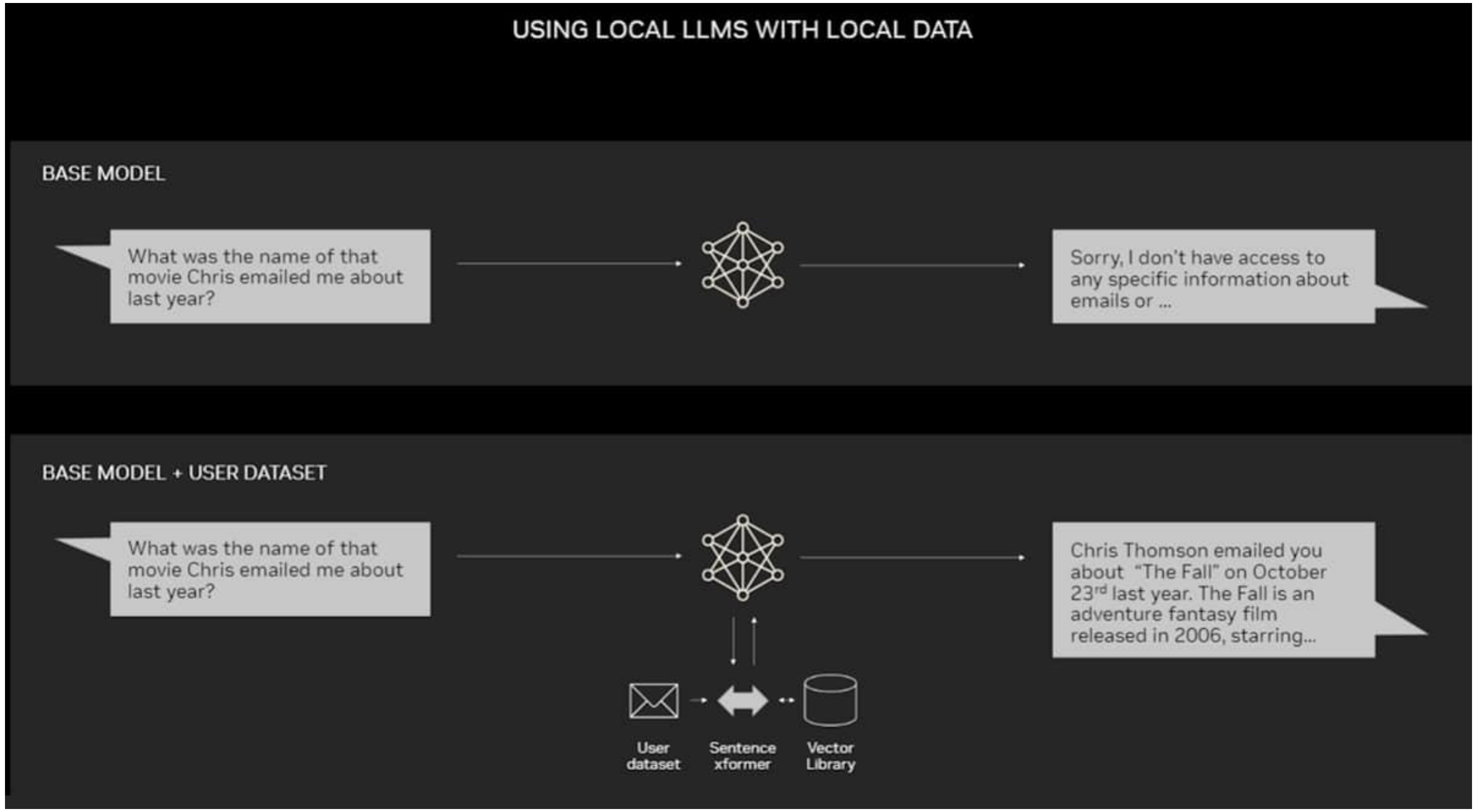
Application systems: outline

- Server design
- **Retrieval Augmented Generation (RAG)**
- AI agents

Retrieval Augmented Generation (RAG)

Directly using LLMs faces problems

- Information lag
- Model hallucination
- Hard to incorporate proprietary data



Retrieval Augmented Generation (RAG)

LangChain

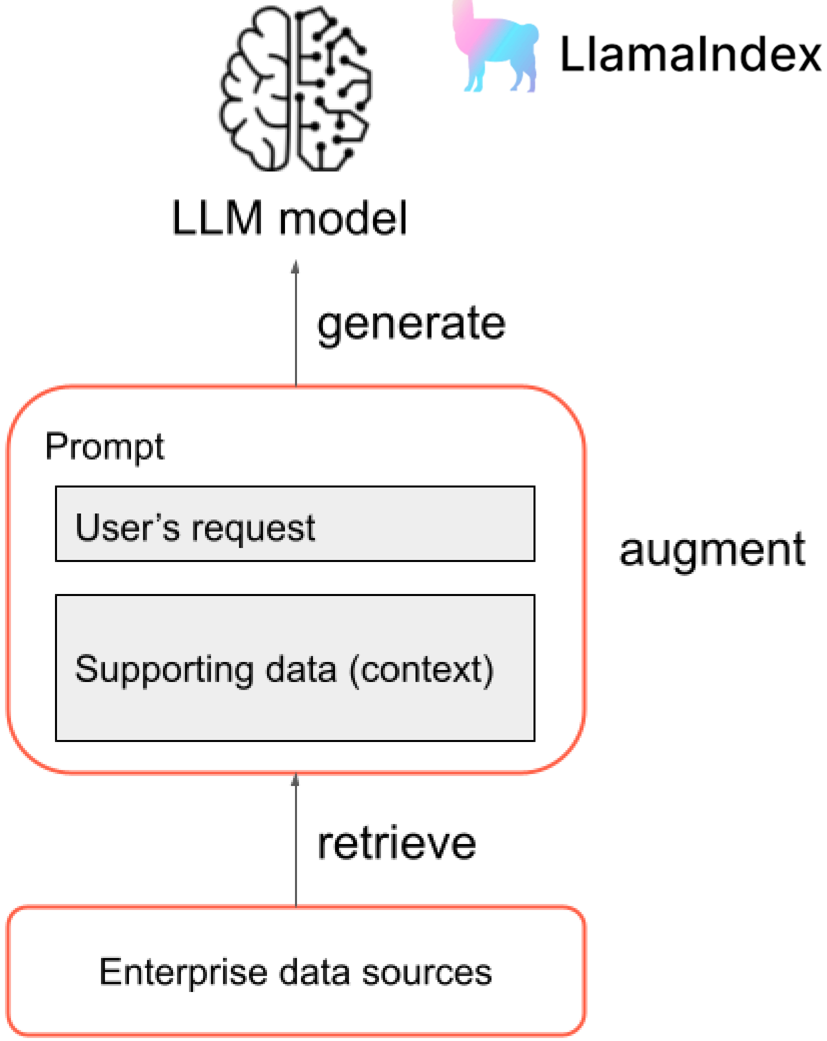
LlamaIndex

Directly using LLMs faces problems

- Information lag
- Model hallucination
- Hard to incorporate proprietary data

Instead, we need RAG =

- **Retrieval**: The user's request is used to query some external info - querying a vector store, a keyword search over text, or querying a database. This is to obtain supporting data / context that helps the LLM provide a useful response.
- **Augmentation**: The supporting data / context is combined with the user request, often using a template with instructions to the LLM, to create a prompt.
- **Generation**: The LLM generates a response to the prompt.



With an LLM alone	Using LLMs with RAG
No proprietary knowledge: LLMs are generally trained on publicly available data, so they cannot accurately answer questions about a company's internal or proprietary data.	**RAG applications can incorporate proprietary data:** A RAG application can supply proprietary documents such as memos, emails, and design documents to an LLM, enabling it to answer questions about those documents.
Knowledge isn't updated in real time: LLMs do not have access to information about events that occurred after they were trained. For example, a standalone LLM cannot tell you anything about stock movements today.	**RAG applications can access real-time data:** A RAG application can supply the LLM with timely information from an updated data source, allowing it to provide useful answers about events past its training cutoff date.
Lack of citations: LLMs cannot cite specific sources of information when responding, leaving the user unable to verify whether the response is factually correct or a hallucination.	**RAG can cite sources:** When used as part of a RAG application, an LLM can be asked to cite its sources.
Lack of data access controls (ACLs): LLMs alone can't reliably provide different answers to different users based on specific user permissions.	**RAG allows for data security/ACLs:** The retrieval step can be designed to find only the information that the user has credentials to access, enabling a RAG application to selectively retrieve personal or proprietary information.

RAG workflow

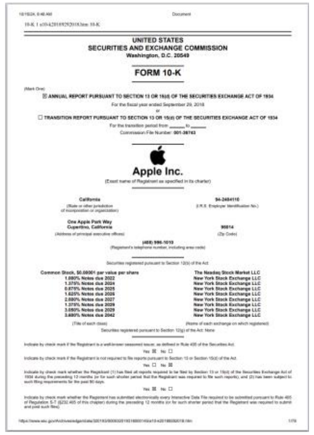
(Offline) Preprocess

- Chunking documents with simple heuristics (1)
- Compute embeddings w/ a pre-trained model (2)
- Indexing & store the embeddings in a database (2)

(Online) When a user query comes

- Compute embedding for the user query (3)
- Retrieve relevant embeddings from the database (4)
- Assemble a prompt, send it to LLM for result (5-7)

Example: Ask “How many employees?” to an SEC filing


UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
FORM 10-K
Apple Inc.

~100 pages, tables, text

“Retrieved” context from the document:

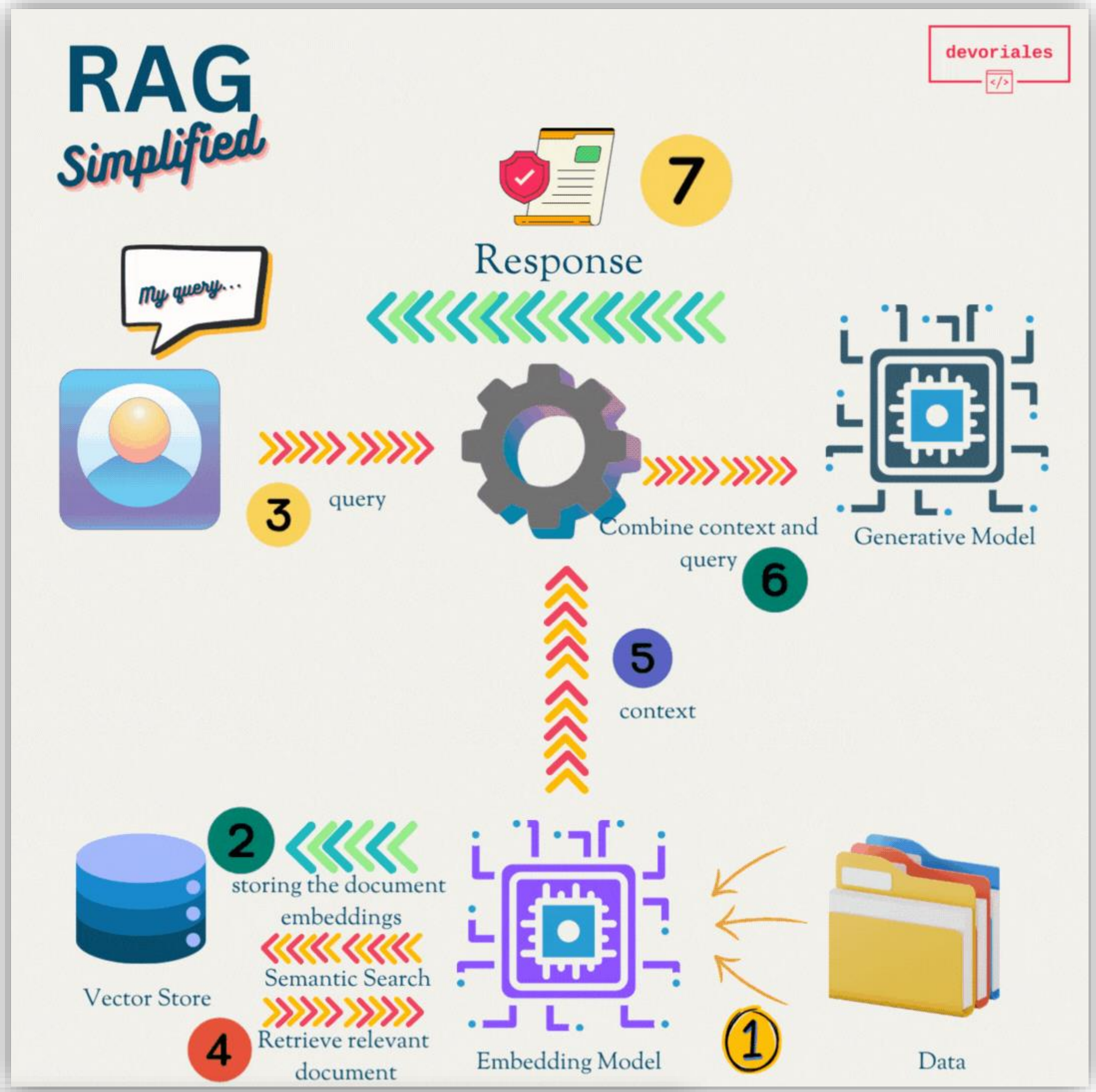
Backlog

In the Company’s experience, the actual amount of product backlog at any particular time is not a meaningful indication of its future business prospects. In particular, backlog often increases immediately following new product introductions as customers anticipate shortages. Backlog is often reduced once customers believe they can obtain sufficient supply. Because of the foregoing, backlog should not be considered a reliable indicator of the Company’s ability to achieve any particular level of revenue or financial performance.

Employees

As of September 29, 2018, the Company had approximately 132,000 full-time equivalent employees.

Apple Inc. | 2018 Form 10-K | 6



Credits: devoriales.com

Drawbacks of RAG

- **What if retrieval goes wrong?**
  - Raw documents are highly nonstructured
  - Documents are too long
  - Complex retrieval
  - Ranking is wrong
- **What if generation goes wrong?**
  - Prompt is too complex / long
  - Generation doesn't follow instruction / format requirement

Note 3 – Financial Instruments

Cash, Cash Equivalents and Marketable Securities

The following tables show the Company's cash, cash equivalents and marketable securities by significant investment category as of December 31, 2022 and September 24, 2022 (in millions):

	December 31, 2022						
	Adjusted Cost	Unrealized Gains	Unrealized Losses	Fair Value	Cash and Cash Equivalents	Current Marketable Securities	Non-Current Marketable Securities
Cash	$ 17,908	$ —	$ —	$ 17,908	$ 17,908	$ —	$ —
Level 1 [1]:							
Money market funds	818	—	—	818	818	—	—
Mutual funds	330	2	(40)	292	—	292	—
Subtotal	1,148	2	(40)	1,110	818	292	—
Level 2 [2]:							
U.S. Treasury securities	24,128	1	(1,576)	22,553	13	9,105	13,435
U.S. agency securities	5,743	—	(643)	5,100	—	310	4,790
Non-U.S. government securities	17,778	14	(1,029)	16,763	—	9,907	6,856
Certificates of deposit and time deposits	2,025	—	—	2,025	1,795	230	—
Commercial paper	237	—	—	237	—	237	—
Corporate debt securities	85,895	14	(7,039)	78,870	1	10,377	68,492
Municipal securities	864	—	(26)	838	—	278	560
Mortgage- and asset-backed securities	22,448	3	(2,405)	20,046	—	84	19,962
Subtotal	159,118	32	(12,718)	146,432	1,809	30,528	114,095
Total [3]	$ 178,174	$ 34	$ (12,758)	$ 165,450	$ 20,535	$ 30,820	$ 114,095

	September 24, 2022						
	Adjusted Cost	Unrealized Gains	Unrealized Losses	Fair Value	Cash and Cash Equivalents	Current Marketable Securities	Non-Current Marketable Securities
Cash	$ 18,546	$ —	$ —	$ 18,546	$ 18,546	$ —	$ —
Level 1 [1]:							
Money market funds	2,929	—	—	2,929	2,929	—	—
Mutual funds	274	—	(47)	227	—	227	—
Subtotal	3,203	—	(47)	3,156	2,929	227	—
Level 2 [2]:							
U.S. Treasury securities	25,134	—	(1,725)	23,409	338	5,091	17,980
U.S. agency securities	5,823	—	(655)	5,168	—	240	4,928
Non-U.S. government securities	16,948	2	(1,201)	15,749	—	8,806	6,943
Certificates of deposit and time deposits	2,067	—	—	2,067	1,805	262	—
Commercial paper	718	—	—	718	28	690	—
Corporate debt securities	87,148	9	(7,707)	79,450	—	9,023	70,427
Municipal securities	921	—	(35)	886	—	266	620
Mortgage- and asset-backed securities	22,553	—	(2,593)	19,960	—	53	19,907
Subtotal	161,312	11	(13,916)	147,407	2,171	24,431	120,805
Total [3]	$ 183,061	$ 11	$ (13,963)	$ 169,109	$ 23,646	$ 24,658	$ 120,805

(1) Level 1 fair value estimates are based on quoted prices in active markets for identical assets or liabilities.

(2) Level 2 fair value estimates are based on observable inputs other than quoted prices in active markets for identical assets and liabilities, quoted prices for identical or similar assets or liabilities in inactive markets, or other inputs that are observable or can be corroborated by observable market data for substantially the full term of the assets or liabilities.

(3) As of December 31, 2022 and September 24, 2022, total marketable securities included $13.6 billion and $12.7 billion, respectively, that were restricted from general use, related to the European Commission decision finding that Ireland granted state aid to the Company, and other agreements.

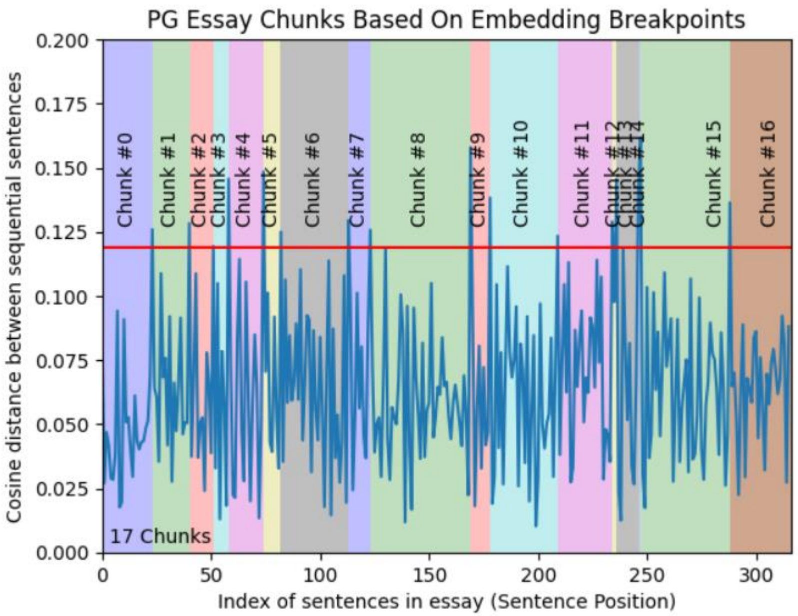
Looking back on the info retrieval literature

Many IR techniques can be applied to RAG

- Better chunking mechanisms
- Prompt compression
- Learning to rank / re-ranking
- Model selection, finetuning & distillation
- Multi-way retrieval
- Graph RAG

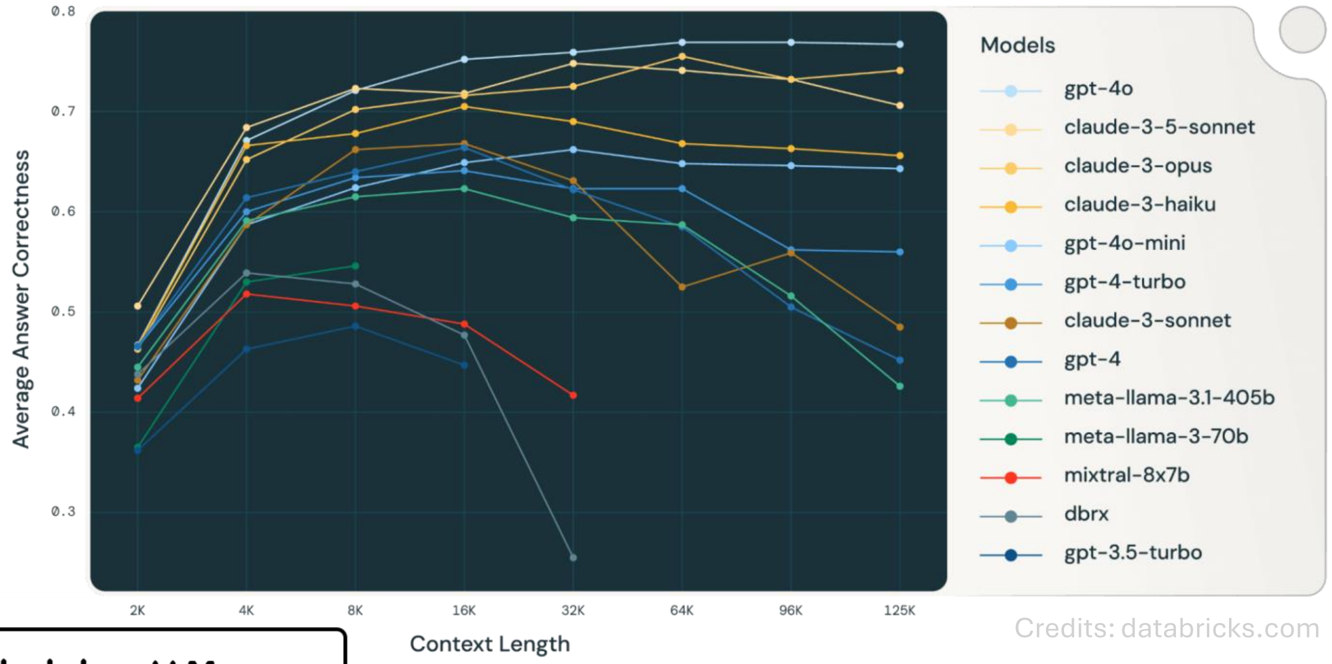
Better chunking mechanisms

- Besides the simple fix-length chunking, there are many other ways:
  - **Overlapping windows** to make sure information is captured in some windows
  - **Structure-aware chunking** to avoid breaking in the middle of paragraphs and sentences
  - **Document based chunking** that leverages the document property (Markdown, HTML, LaTeX etc.)
  - **NLP/Semantic chunking** to detect topic changes
  - **Agentic chucking** uses AI agents to decide if a sentence should be added to the previous chunk.

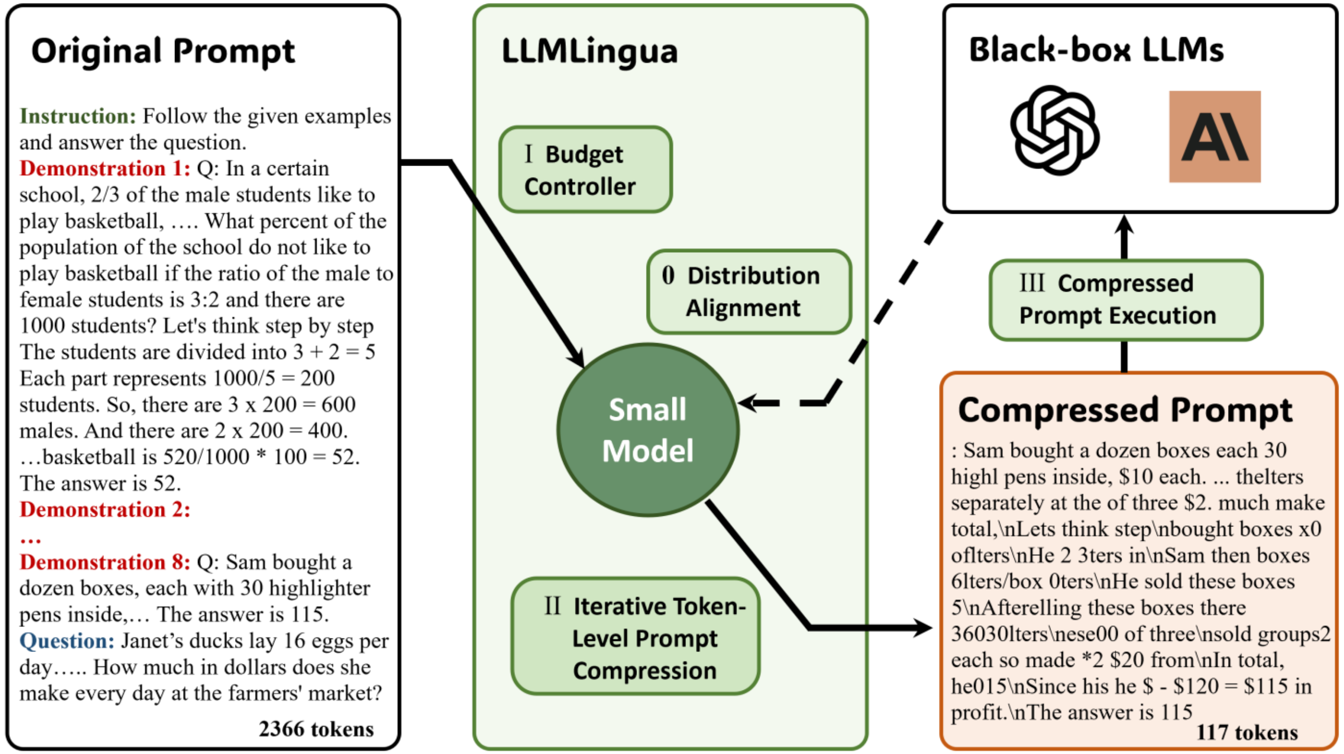


Prompt compression

- More context = more accurate (at cost)
- LLMLingua EMNLP 2023 (Instruction tuning!)



Credits: databricks.com



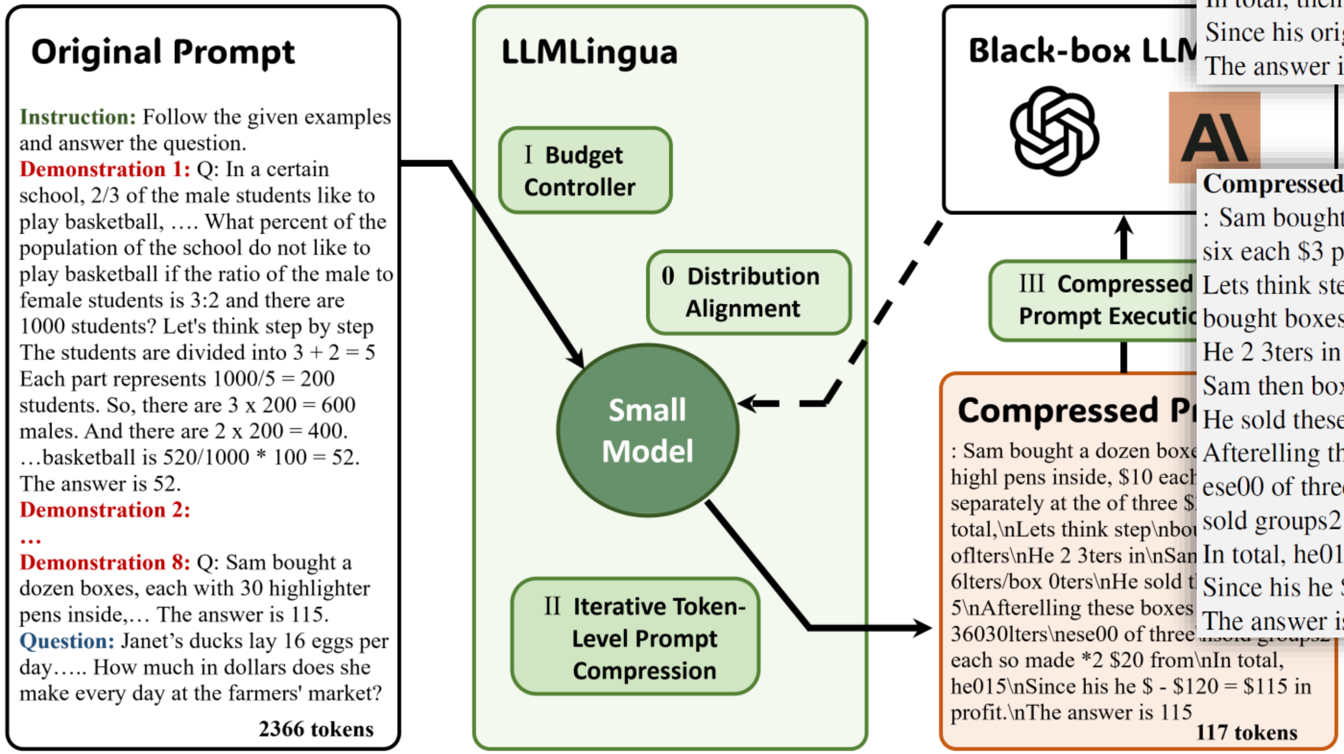
Prompt compression

- More context = more accurate (at cost)
- LLMLingua EMNLP 2023 (Instruction tuning!)

Original Prompt(9-steps Chain-of-Thought):
Question: Sam bought a dozen boxes, each with 30 highlighter pens inside, for $10 each box. He rearranged five of these boxes into packages of six highlighters each and sold them for $3 per package. He sold the rest of the highlighters separately at the rate of three pens for $2. How much profit did he make in total, in dollars?

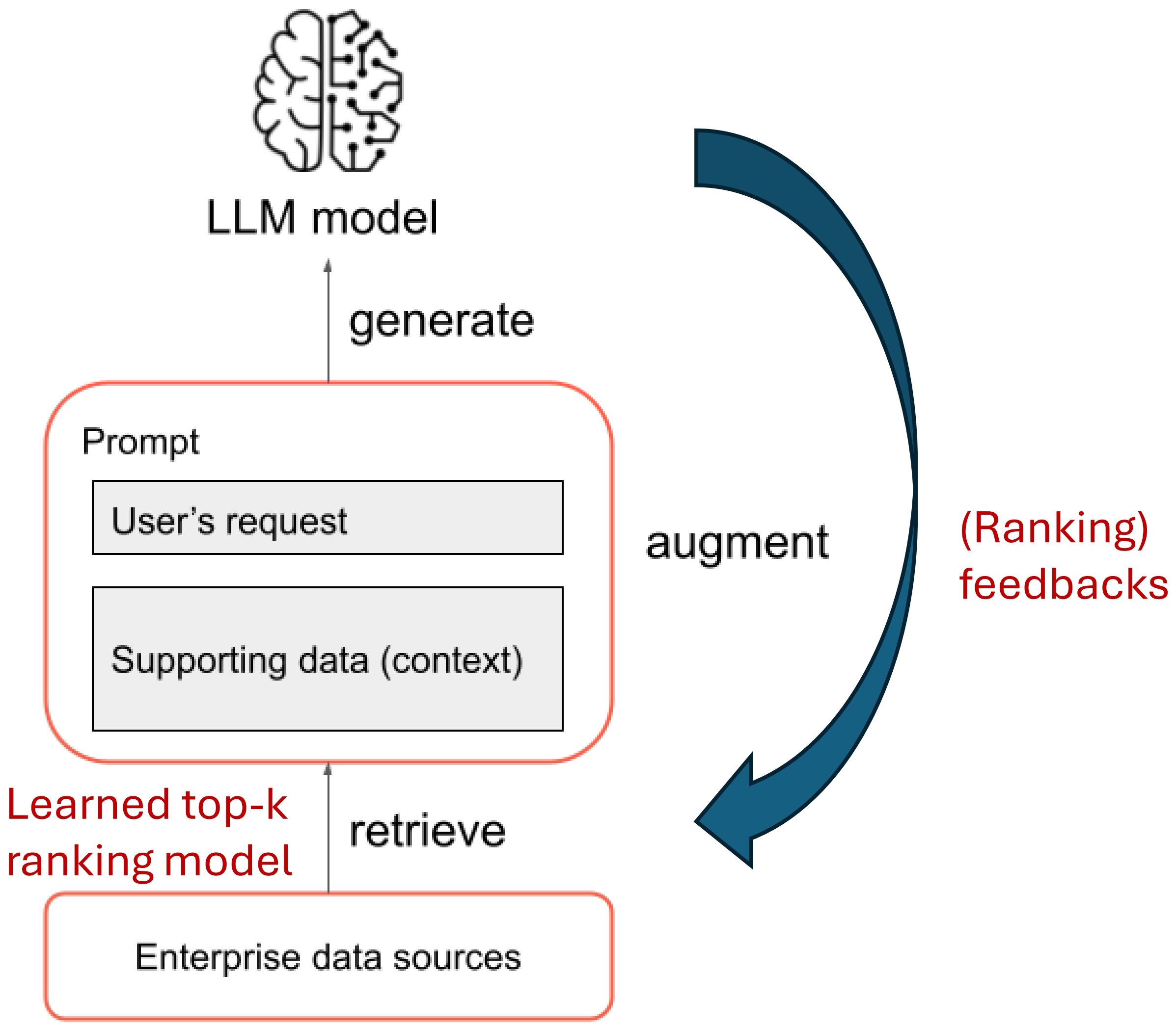
Let's think step by step
Sam bought 12 boxes x $10 = $120 worth of highlighters.
He bought 12 * 30 = 360 highlighters in total.
Sam then took 5 boxes × 6 highlighters/box = 30 highlighters.
He sold these boxes for 5 * $3 = $15
After selling these 5 boxes there were 360 - 30 = 330 highlighters remaining.
These form 330 / 3 = 110 groups of three pens.
He sold each of these groups for $2 each, so made 110 * 2 = $220 from them.
In total, then, he earned $220 + $15 = $235.
Since his original cost was $120, he earned $235 - $120 = $115 in profit.
The answer is 115

Compressed Prompt:
: Sam bought a dozen boxes each 30 highl pens inside, $10 each. He reanged five of boxes into of six each $3 per. He sold the thelters separately at the of three $2. much make total,
Lets think step
bought boxes x0 oflters
He 2 3ters in
Sam then boxes 6lters/box 0ters
He sold these boxes 5
Afterelling these boxes there 36030lters
ese00 of three
sold groups2 each so made *2 $20 from
In total, he015
Since his he $ - $120 = $115 in profit.
The answer is 115



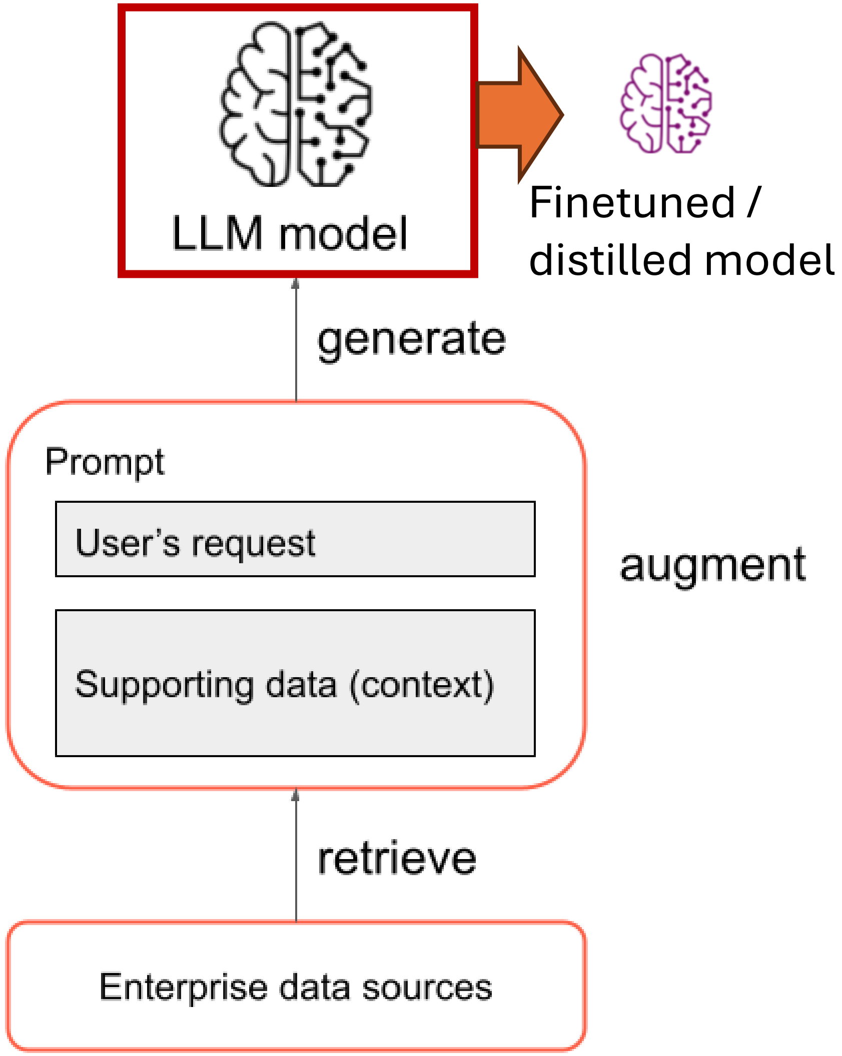
Learning to rank / re-ranking

- The “retrieval” part can be improved by using a learned top-k ranking model (should be cheaper than the later LLM)
- Automatic and free labels from previous runs
- Reduces context length requirements (improve P@K)

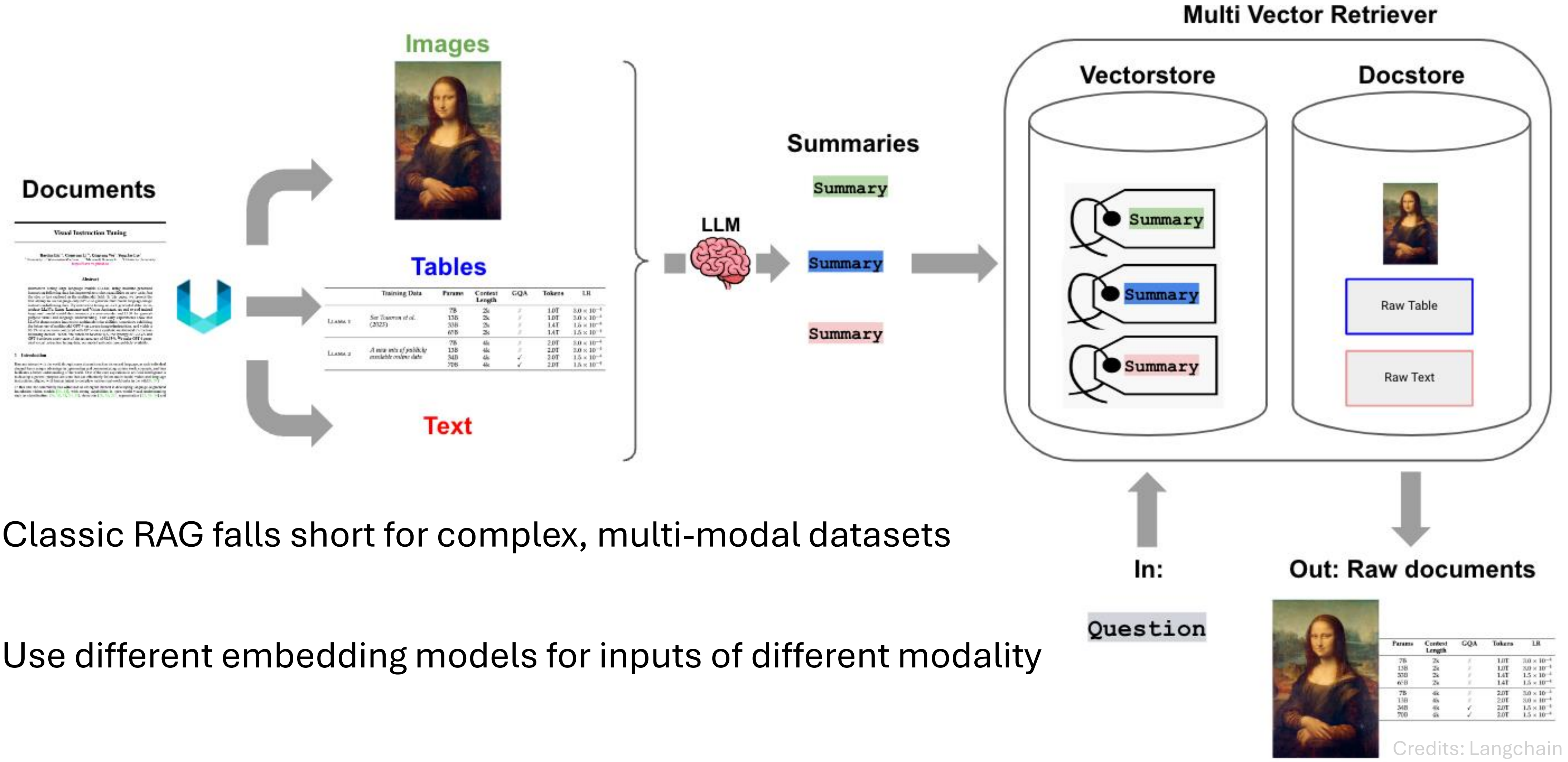


Model selection, finetuning & distillation

- Finetune or distill the generation model in order to reduce size, adapt to formatting requirements. e.g., collect RAG outputs from Llama 70b and send them to finetune Llama 13b
- Or for different queries, use different generation models
- Further, we can propagate the gradients to the embedding phrase, **and finetune embedding models**



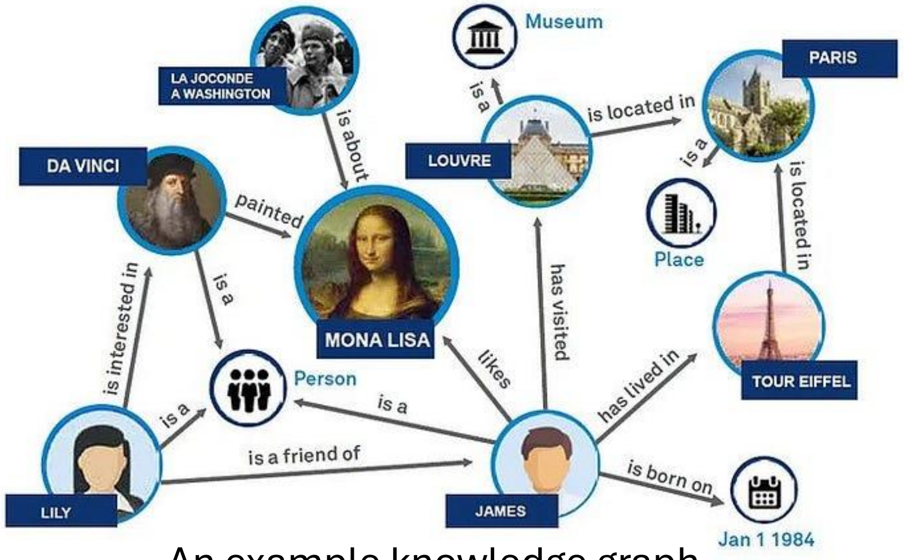
Multi-vector retrieval



- Classic RAG falls short for complex, multi-modal datasets
- Use different embedding models for inputs of different modality

Graph RAG

- Classic RAG approaches do not consider links between entities.
- They also have a wholistic view of the dataset (with simple similar search)
- Given a private dataset, GraphRAG from Microsoft generates the knowledge graph using LLMs, and retrieve for relevant content for new RAG queries.



An example knowledge graph

Query: *"What has Novorossiya done?"*

Baseline RAG	GraphRAG
The text does not provide specific information on what Novorossiya has done.	Novorossiya, a political movement in Ukraine, has been involved in a series of destructive activities, particularly targeting various entities in Ukraine [Entities (6494, 912)]. The movement has been linked to plans to destroy properties of several Ukrainian entities, including Rosen, the Odessa Canning Factory, the Odessa Regional Radio Television Transmission Center, and the National Television Company of Ukraine [Relationships (15207, 15208, 15209, 15210)]. In addition to these, Novorossiya has also been associated with plans to destroy properties of Roshen, the Odessa Conservatory Factory, the Odesa Regional Radio Television Transmission Center, and the National Telecommunications Company of Ukraine [Relationships (8329, 8330, 8331, 8332)]. Furthermore, Novorossiya has been implicated in plans to blow up ATMs of PrivatBank, the largest state bank in Ukraine [Relationships (8335, 8335)]. The Office of the General Prosecutor in Ukraine has reported on the creation of Novorossiya, indicating the government's awareness and potential concern over the activities of this movement [Relationships (8264, 8264)]. The destructive activities of Novorossiya have had significant impacts on the targeted entities and have likely contributed to the ongoing tensions and conflicts in the region. In summary, Novorossiya has been involved in a series of destructive activities targeting various entities in Ukraine. These activities have had significant impacts on the targeted entities and have likely contributed to the ongoing tensions and conflicts in the region. The Ukrainian government, through the Office of the General Prosecutor, has acknowledged the existence and activities of Novorossiya, indicating a level of concern over the movement's actions.

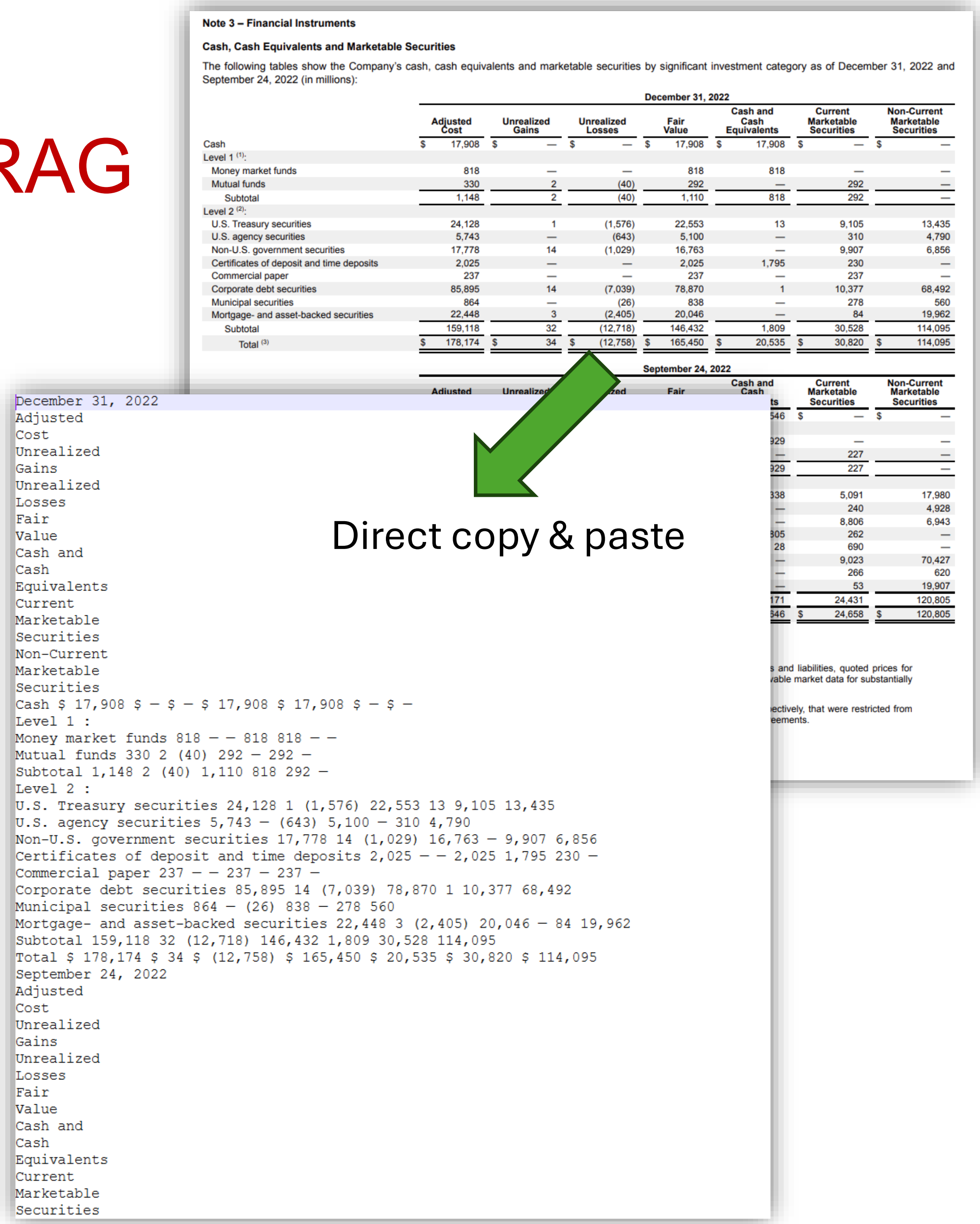
What’s more: raw documents in RAG

- Parsing & cleaning raw documents into structured data is often challenging: noisy, unstructured, long documents

- Long-context vs RAG
  - **Long-context LLMs**: simple (for developers) but often more expensive (for users), can lost in the middle
  - **RAG**: cheaper, deterministic security, easier to debug, up-to-date info

- Our recent work UDA: A Benchmark Suite for Retrieval Augmented Generation in Real-world Document Analysis. NeurIPS 2024.
  - Studied ~3K real-world documents with ~30K annotated QA pairs.
  - Many existing RAG solutions assume clean & structured inputs, which results in accuracy degrade.
  - Small models already work well in certain RAG applications.
  - Long-context LLMs often fall short in some tasks that require numerical reasoning.

  - Access: https://github.com/qinchuanhui/UDA-Benchmark



Note 3 – Financial Instruments

Cash, Cash Equivalents and Marketable Securities

The following tables show the Company's cash, cash equivalents and marketable securities by significant investment category as of December 31, 2022 and September 24, 2022 (in millions):

	December 31, 2022						
	Adjusted Cost	Unrealized Gains	Unrealized Losses	Fair Value	Cash and Cash Equivalents	Current Marketable Securities	Non-Current Marketable Securities
Cash	$ 17,908	$ —	$ —	$ 17,908	$ 17,908	$ —	$ —
Level 1 (1):							
Money market funds	818	—	—	818	818	—	—
Mutual funds	330	2	(40)	292	—	292	—
Subtotal	1,148	2	(40)	1,110	818	292	—
Level 2 (2):							
U.S. Treasury securities	24,128	1	(1,576)	22,553	13	9,105	13,435
U.S. agency securities	5,743	—	(643)	5,100	—	310	4,790
Non-U.S. government securities	17,778	14	(1,029)	16,763	—	9,907	6,856
Certificates of deposit and time deposits	2,025	—	—	2,025	1,795	230	—
Commercial paper	237	—	—	237	—	237	—
Corporate debt securities	85,895	14	(7,039)	78,870	1	10,377	68,492
Municipal securities	864	—	(26)	838	—	278	560
Mortgage- and asset-backed securities	22,448	3	(2,405)	20,046	—	84	19,962
Subtotal	159,118	32	(12,718)	146,432	1,809	30,528	114,095
Total (3)	$ 178,174	$ 34	$ (12,758)	$ 165,450	$ 20,535	$ 30,820	$ 114,095

September 24, 2022

```
December 31, 2022
Adjusted
Cost
Unrealized
Gains
Unrealized
Losses
Fair
Value
Cash and
Cash
Equivalents
Current
Marketable
Securities
Non-Current
Marketable
Securities
Cash $ 17,908 $ — $ — $ 17,908 $ 17,908 $ — $ —
Level 1 :
Money market funds 818 — — 818 818 — —
Mutual funds 330 2 (40) 292 — 292 —
Subtotal 1,148 2 (40) 1,110 818 292 —
Level 2 :
U.S. agency securities 24,128 1 (1,576) 22,553 13 9,105 13,435
U.S. agency securities 5,743 — (643) 5,100 — 310 4,790
Non-U.S. government securities 17,778 14 (1,029) 16,763 — 9,907 6,856
Certificates of deposit and time deposits 2,025 — — 2,025 1,795 230 —
Commercial paper 237 — — 237 — 237 —
Corporate debt securities 85,895 14 (7,039) 78,870 1 10,377 68,492
Municipal securities 864 — (26) 838 — 278 560
Mortgage- and asset-backed securities 22,448 3 (2,405) 20,046 — 84 19,962
Subtotal 159,118 32 (12,718) 146,432 1,809 30,528 114,095
Total $ 178,174 $ 34 $ (12,758) $ 165,450 $ 20,535 $ 30,820 $ 114,095
September 24, 2022
Adjusted
Cost
Unrealized
Gains
Unrealized
Losses
Fair
Value
Cash and
Cash
Equivalents
Current
Marketable
Securities
```

Application systems: outline

- Server design
- Retrieval Augmented Generation (RAG)
- **AI agents**

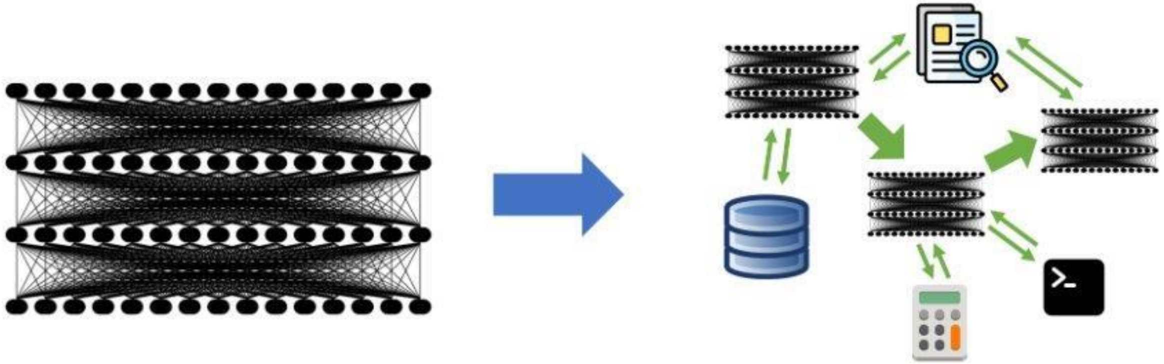
What are future AI applications like?

Generative

- Generate content like text & image

Agentic

- Execute complex tasks on behalf of human



Zaharia et al. 2024. The Shift from Models to Compound AI Systems

Examples of agentic AI

- Personal assistants
- Autonomous robots
- Gaming agents

- Science agents
- Web agents
- Software agents

Key benefits of agentic AI

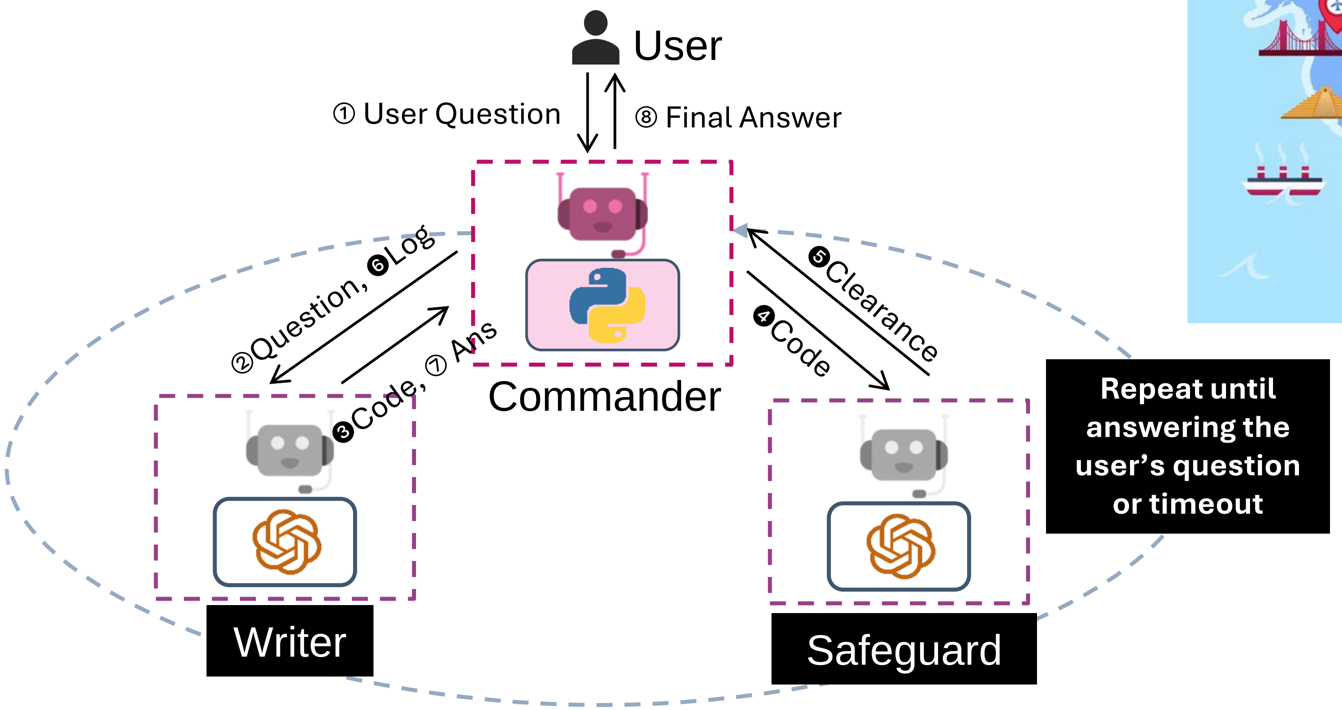
- Useful Interface
  - Natural interaction with human agency

- Strong Capability
  - Operate with minimal human intervention

- Useful Architecture
  - Intuitive programming paradigm

Example workflow of agentic AI

What if we prohibit shipping from supplier 1 to roastery 2?

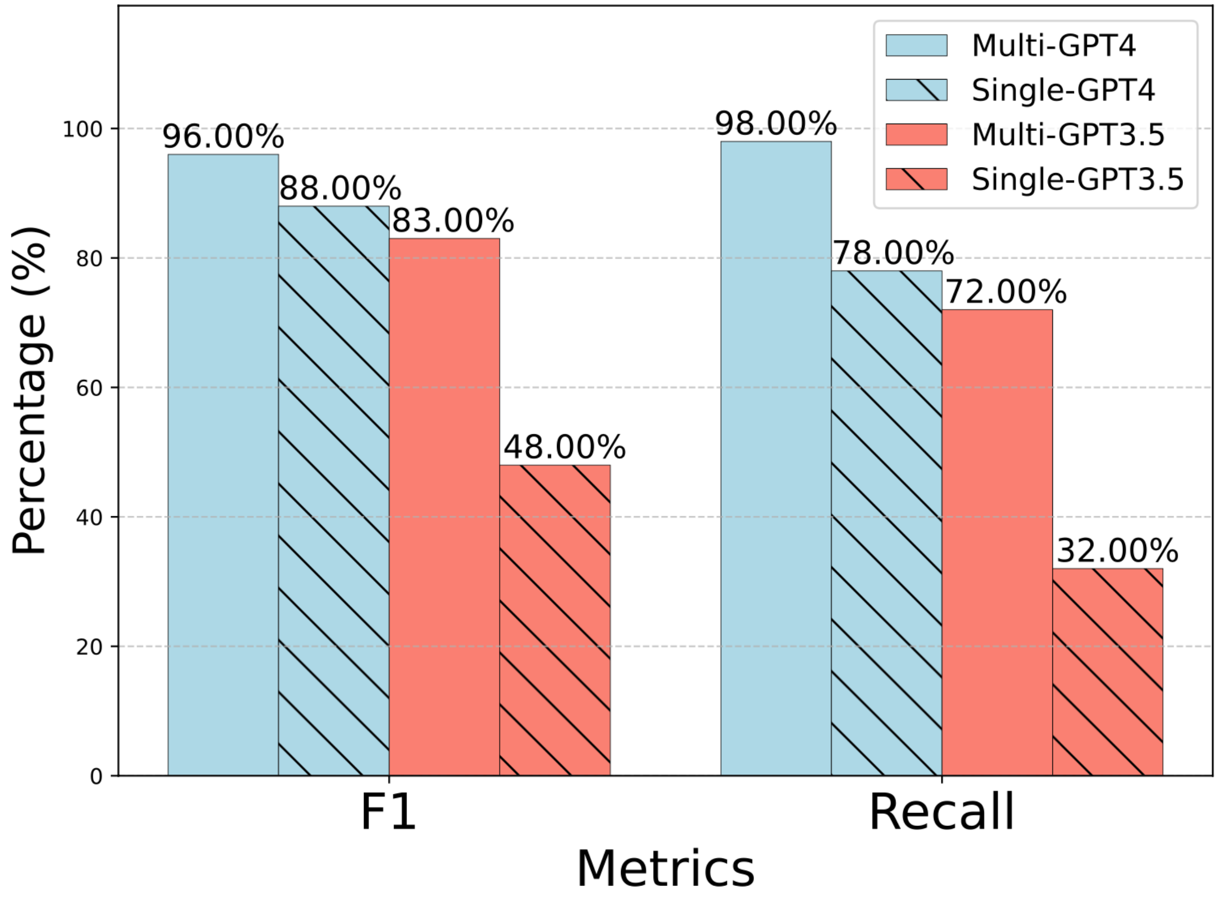


Example workflow of agentic AI


```
Optimize a model with 11 rows, 18 columns and 36 nonzeros
Model fingerprint: 0x8aa2c280
Variable types: 0 continuous, 18 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+00, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 2e+02]
Found heuristic solution: objective 2900.0000000
Presolve time: 0.00s
Presolved: 11 rows, 18 columns, 36 nonzeros
Variable types: 0 continuous, 18 integer (0 binary)
Found heuristic solution: objective 2896.0000000

Root relaxation: objective 2.470000e+03, 11 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    2470.0000000 2470.00000  0.00%     -    0s

Explored 1 nodes (11 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 14 (of 14 available processors)

Solution count 3: 2470 2896 2900
```

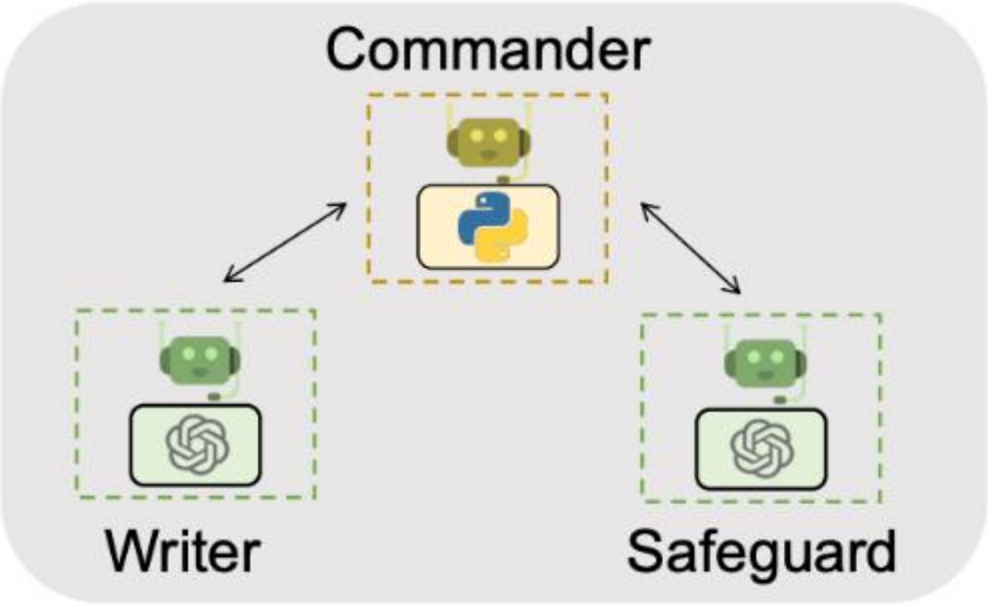
Agentic programming

- Handle more complex tasks / Improve response quality
  - Improve over natural iteration
  - Divide & conquer
  - Grounding & validation



Agentic programming

- Easy to understand, maintain, extend
  - Modular composition
  - Natural human participation
  - Fast & creative experimentation

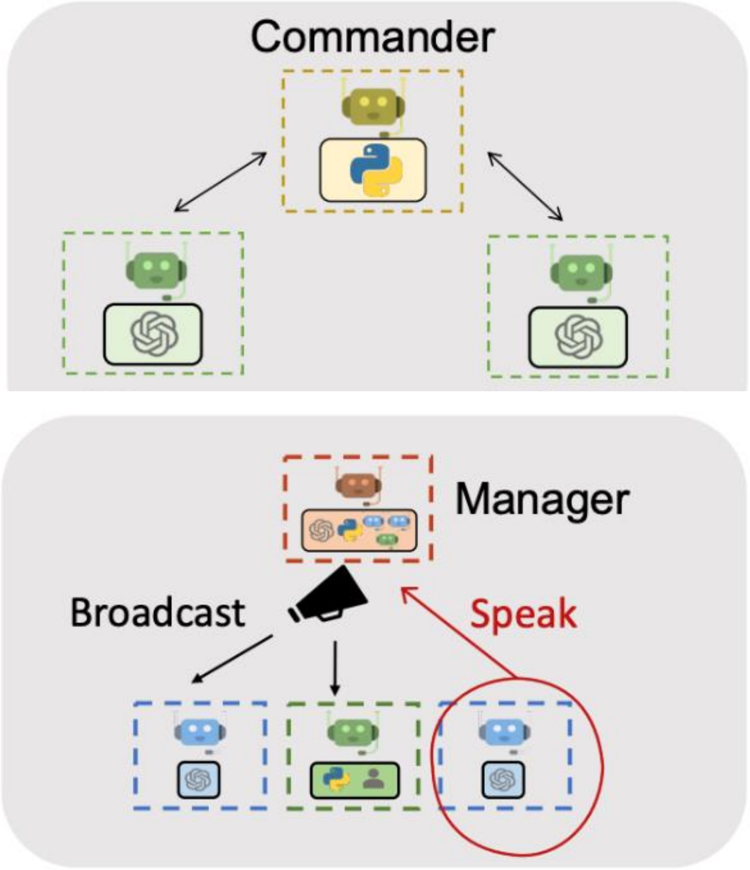


Agentic abstraction

Unify models, tools, human for compound AI systems

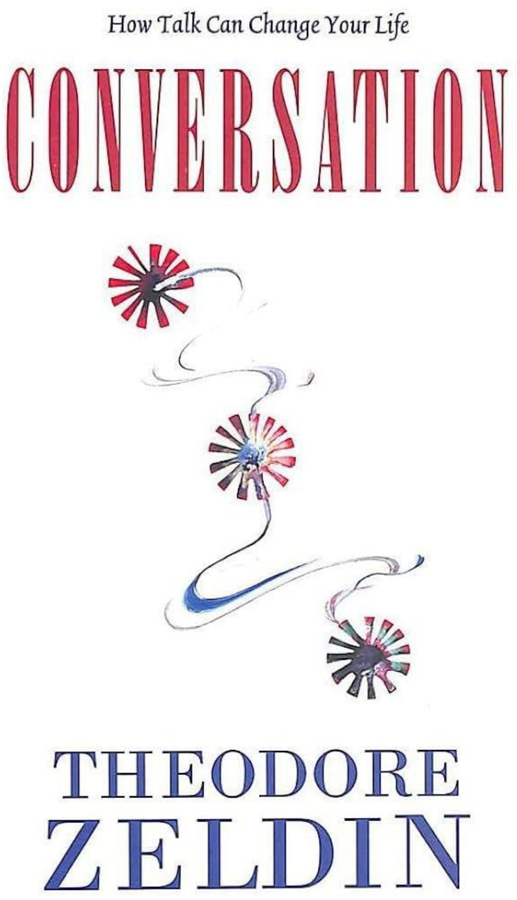
Multi-agent orchestration

- Static/dynamic
- NL/PL
- Context sharing/isolation
- Cooperation/competition
- Centralized/decentralized
- Intervention/automation

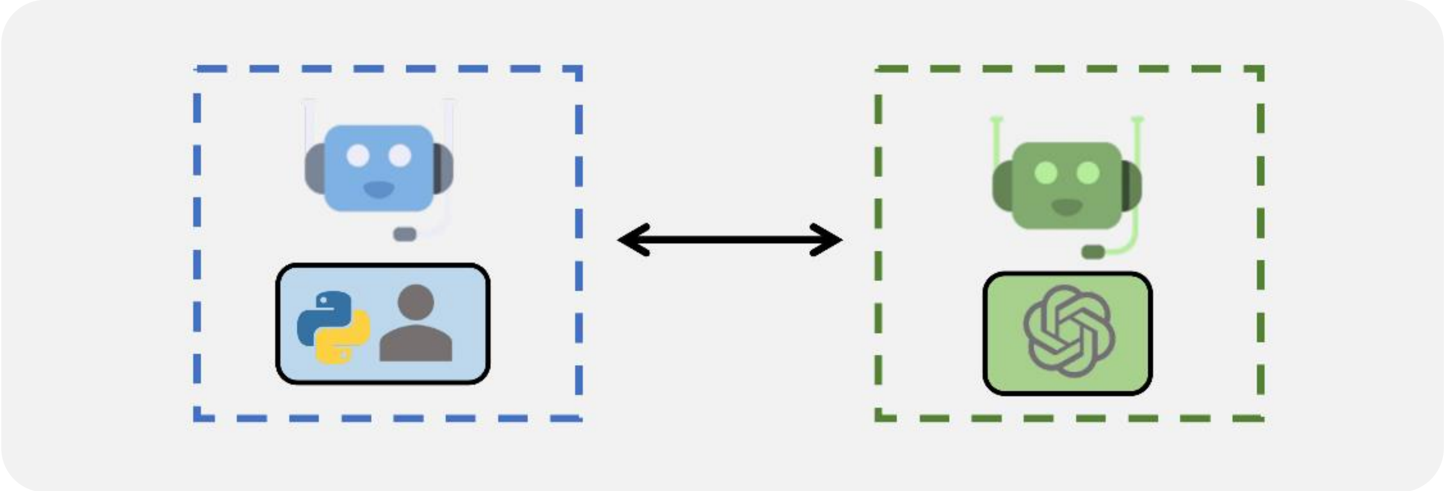


Agentic design patterns

- Conversation
- Prompting & reasoning
- Tool use
- Planning
- Integrating multiple models, modalities and memories



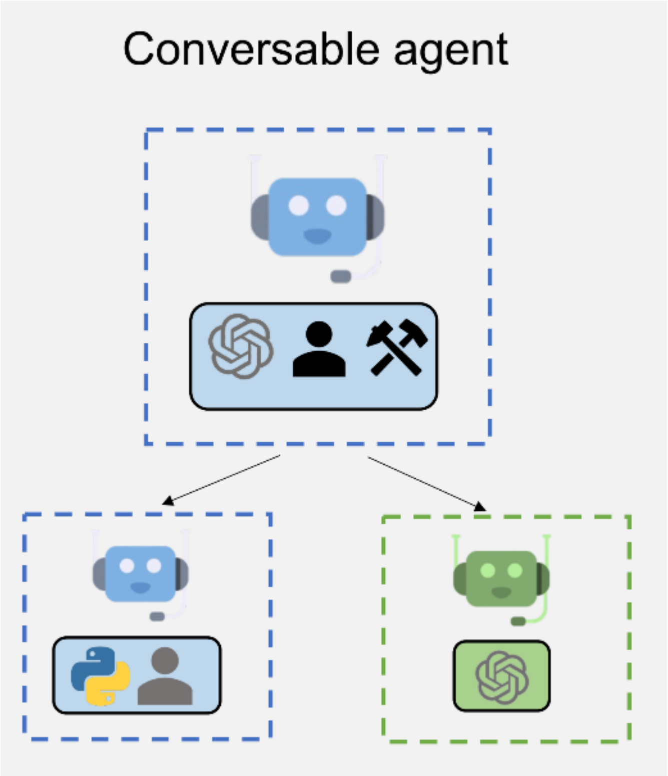
AutoGen: a programming framework for agentic AI



Initially developed in FLAML (Nov 2022)
Spined off to a standalone repo (October 2023)
Standalone GitHub organization AutoGen-AI (August 2024)

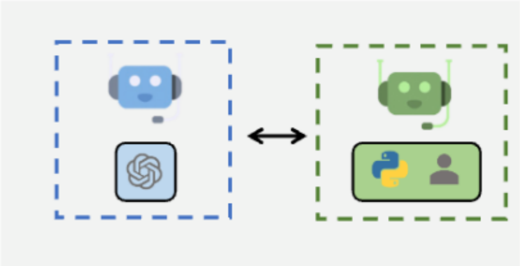
https://github.com/autogen-ai

Define agents:
Conversable & Customizable

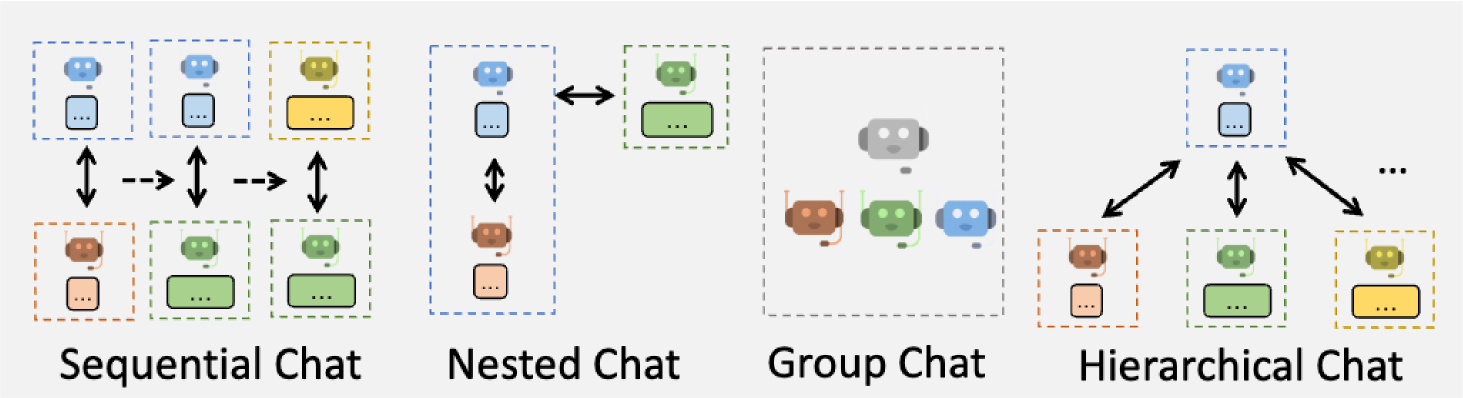


Agent Customization

Get them to talk:
Conversation Programming



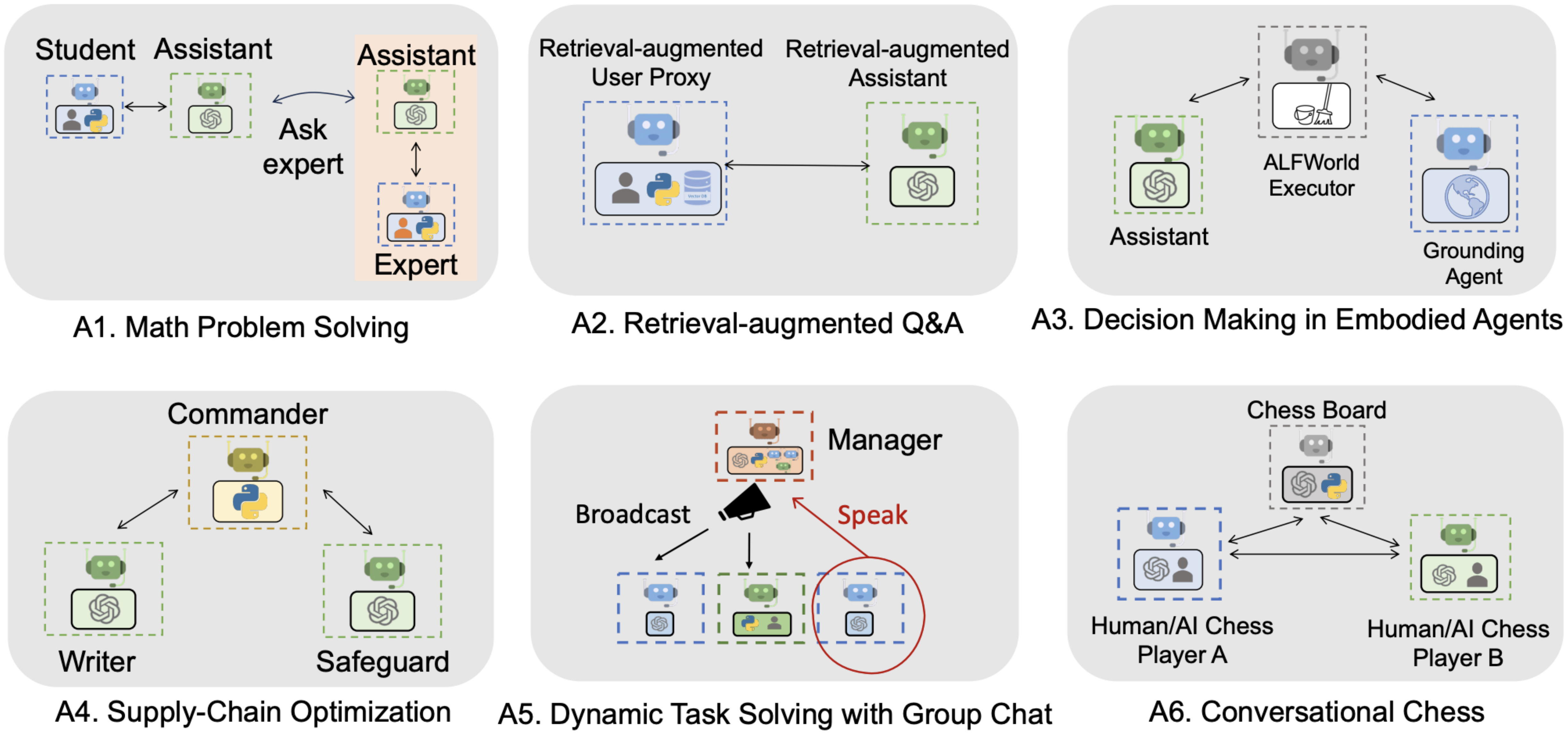
Multi-Agent Conversations



Flexible Conversation Patterns

Simple programming interface

```
writer = Writer("writer", llm_config=llm_config)
safeguard = autogen.AssistantAgent("safeguard", llm_config=llm_config)
commander = Commander("commander", llm_config=llm_config)
user = autogen.UserProxyAgent("user")


writer_chat_queue = [{"recipient": writer, "message": writer_init_messsage,
                      "summary_method": writer_success_summary}]
safeguard_chat_queue = [{"recipient": safeguard, "message": safeguard_init_message,
                         "max_turns": 1, "summary_method": safeguard_summary}]
commander.register_nested_chats(safeguard_chat_queue, trigger="writer")
commander.register_nested_chats(writer_chat_queue, trigger="user")


chat_res = user.initiate_chat(commander, message="What if we prohibit shipping from supplier 1 to roastery 2?")
```

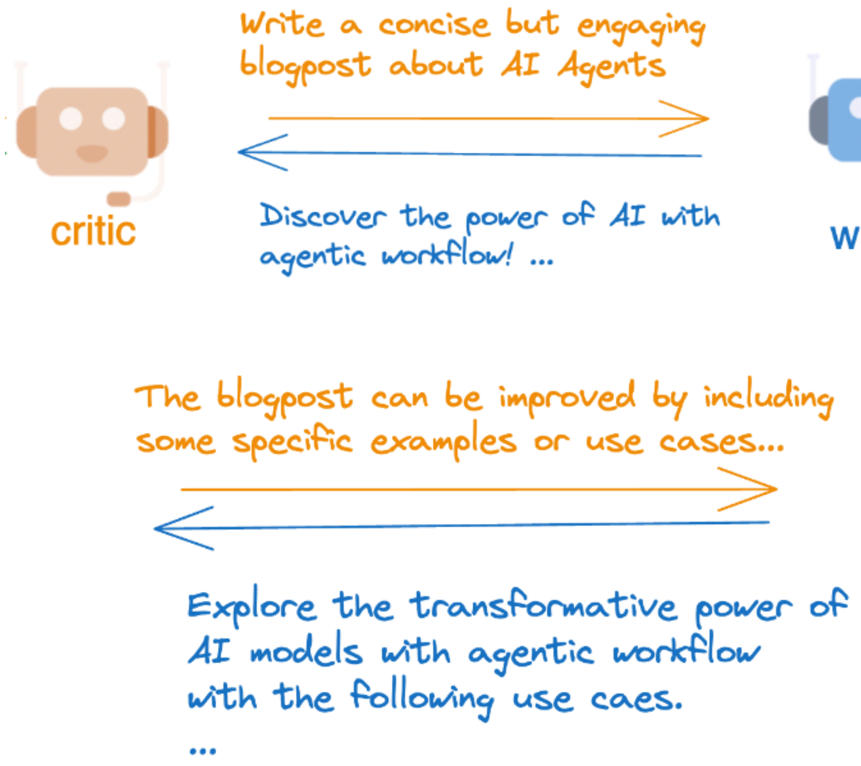


For more examples: https://autogen-ai.github.io/autogen/docs/notebooks

Blogpost writing with reflection

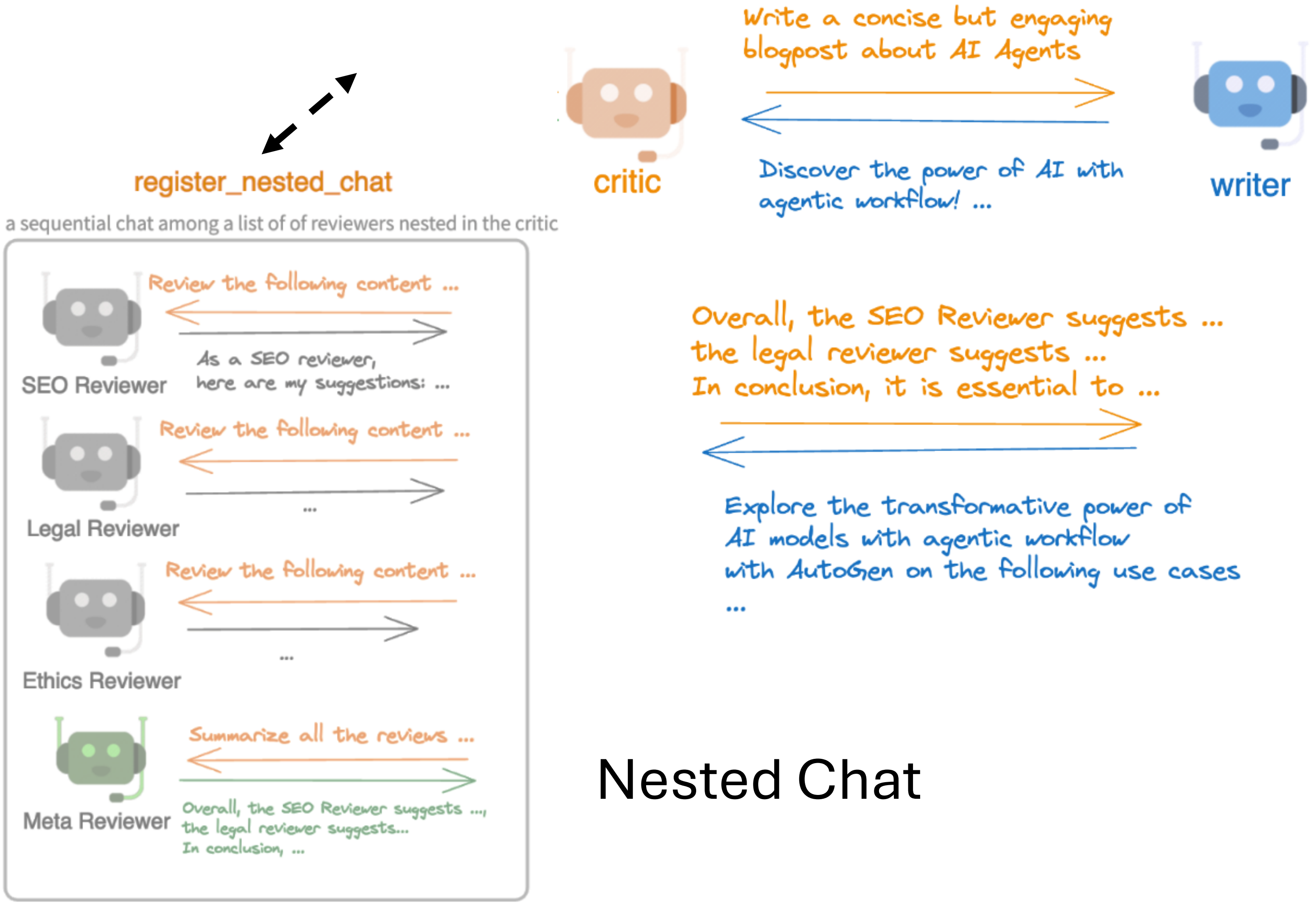
Two-Agent Reflection

```python
writer = autogen.AssistantAgent(
    name="Writer",
    system_message="You are a writer...",
    llm_config=llm_config,
)

critic = autogen.AssistantAgent(
    name="Critic",
    is_termination_msg=lambda x: x.get("content", "").find("TERMINATE") >= 0,
    llm_config=llm_config,
    system_message="You are a critic...",
)

critic.initiate_chat(
    recipient=writer,
    message=task,
    max_turns=2,
    summary_method="last_msg"
)
```



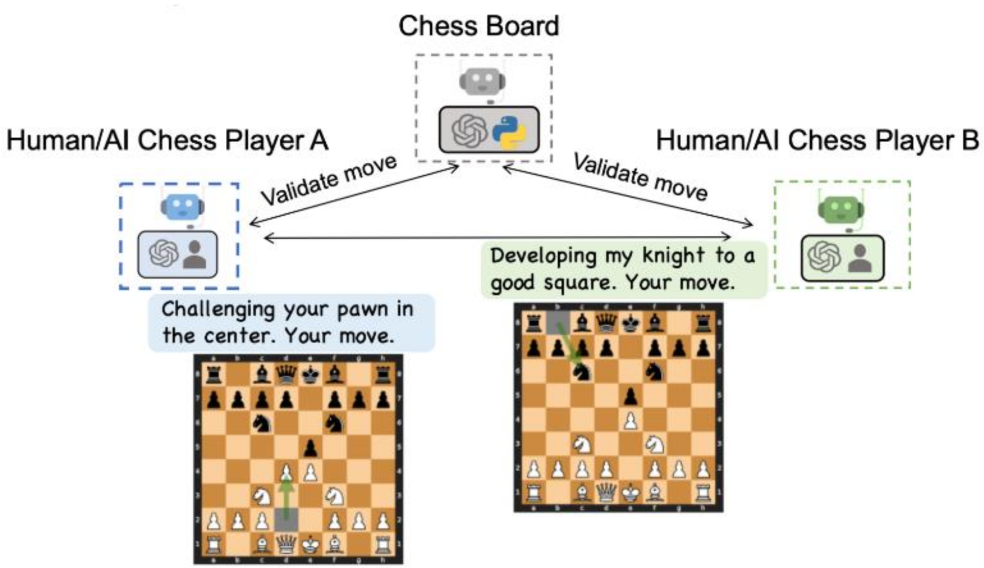
Blogpost writing with advanced reflection

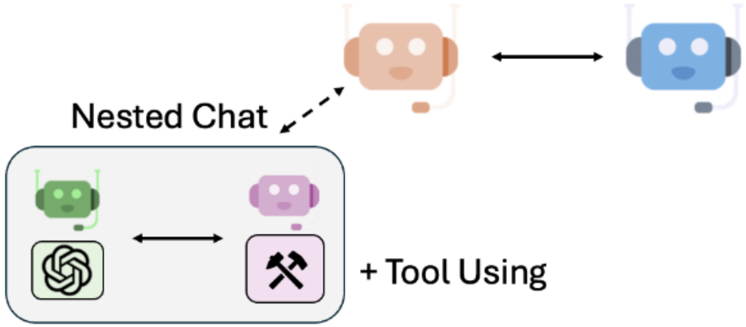
```python
critic.register_nested_chats(
    review_chats,
    trigger=writer,
)

critic.initiate_chat(
    recipient=writer,
    message=task,
    max_turns=2,
    summary_method="last_msg"
)
```

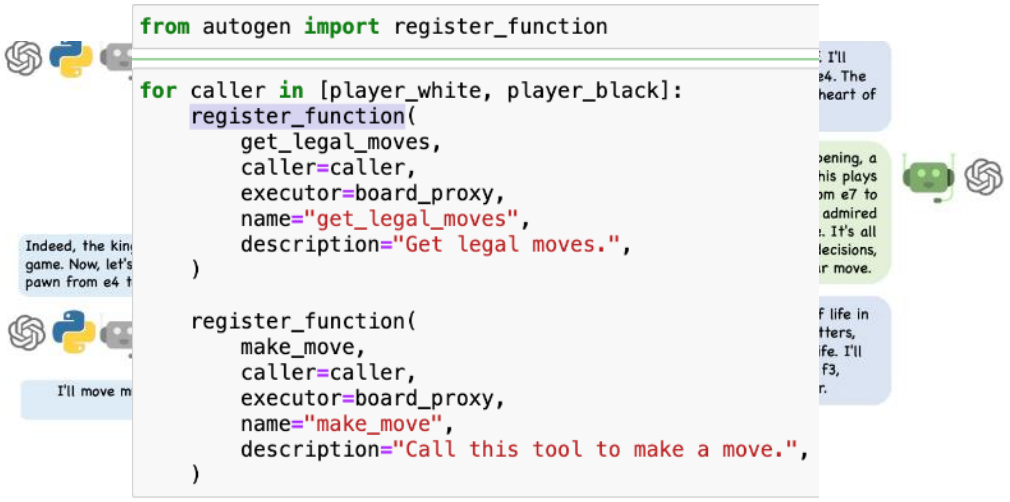


Nested Chat

Conversational chess

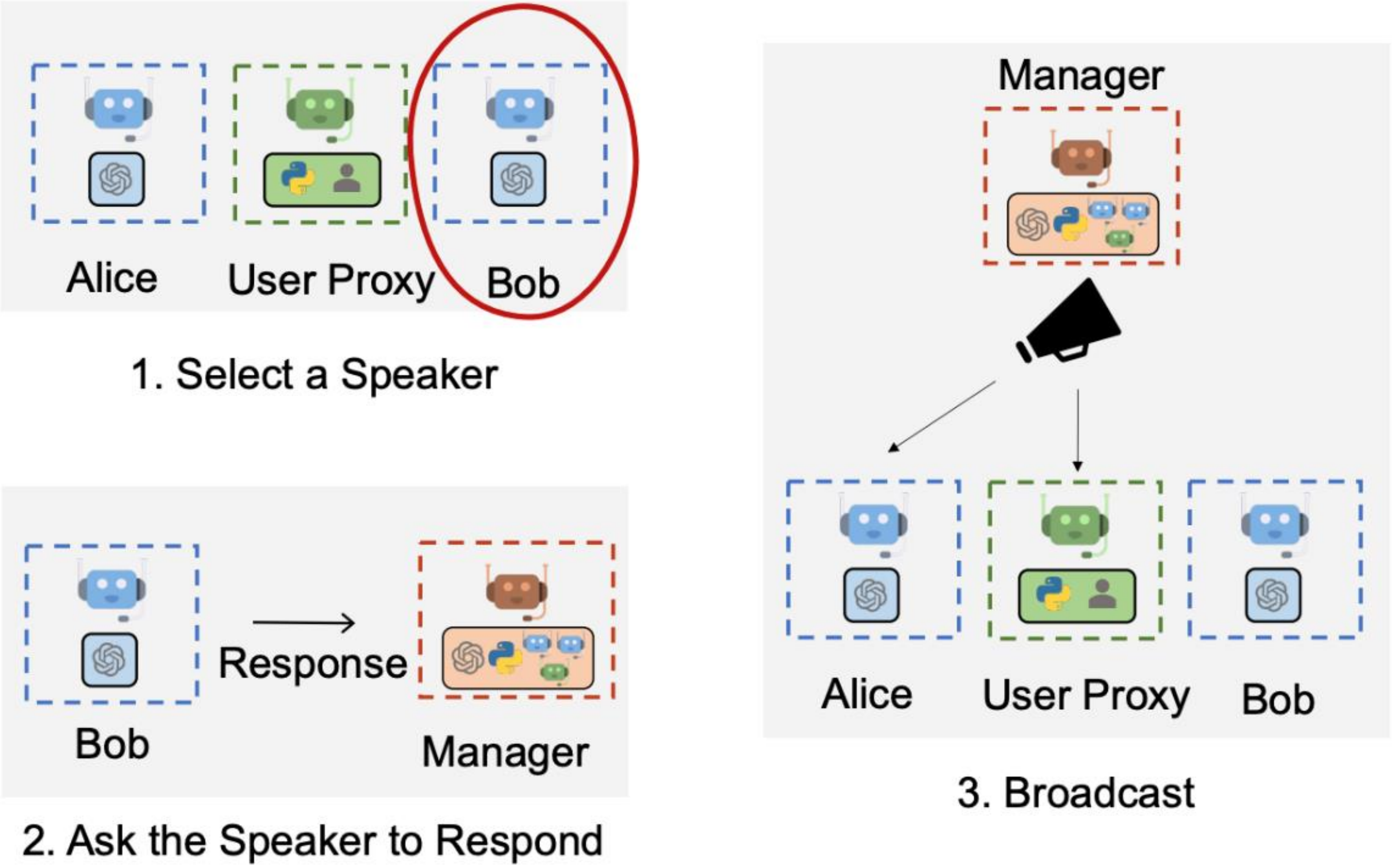






```python
from autogen import register_function

for caller in [player_white, player_black]:
    register_function(
        get_legal_moves,
        caller=caller,
        executor=board_proxy,
        name="get_legal_moves",
        description="Get legal moves.",
    )

    register_function(
        make_move,
        caller=caller,
        executor=board_proxy,
        name="make_move",
        description="Call this tool to make a move.",
    )
```

Complex task planning and solving with group chat



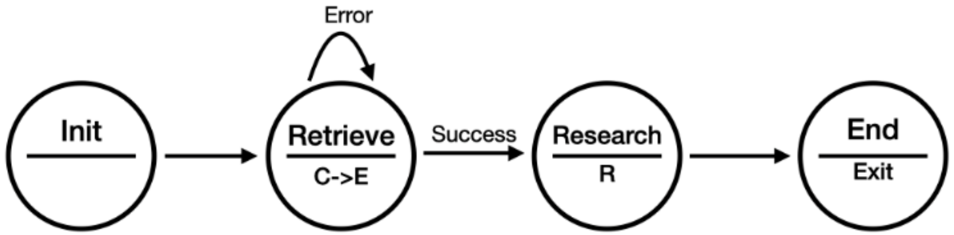
Complex task planning and solving with group chat

StateFlow - Build State-Driven Workflows with Customized Speaker Selection in GroupChat

February 29, 2024 · 7 min read

Yiran Wu
PhD student at Pennsylvania State University

TL;DR: Introduce Stateflow, a task-solving paradigm that conceptualizes complex task-solving processes backed by LLMs as state machines. Introduce how to use GroupChat to realize such an idea with a customized speaker selection function.



C: Coder
E: Code Executor
R: Research

```python
def state_transition(last_speaker, groupchat):
    messages = groupchat.messages

    if last_speaker is initializer:
        # init -> retrieve
        return coder
    elif last_speaker is coder:
        # retrieve: action 1 -> action 2
        return executor
    elif last_speaker is executor:
        if messages[-1]["content"] == "exitcode: 1":
            # retrieve --(execution failed)--> retrieve
            return coder
        else:
            # retrieve --(execution success)--> research
            return scientist
    elif last_speaker == "Scientist":
        # research -> end
        return None


groupchat = autogen.GroupChat(
    agents=[initializer, coder, executor, scientist],
    messages=[],
    max_round=20,
    speaker_selection_method=state_transition,
)
```

SciAgents: Automating scientific discovery through multi-agent intelligent graph reasoning*

Alireza Ghafarollahi
Laboratory for Atomistic and Molecular Mechanics (LAMM)
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139, USA

Markus J. Buehler
Laboratory for Atomistic and Molecular Mechanics (LAMM)
Center for Computational Science and Engineering
Schwarzman College of Computing
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139, USA

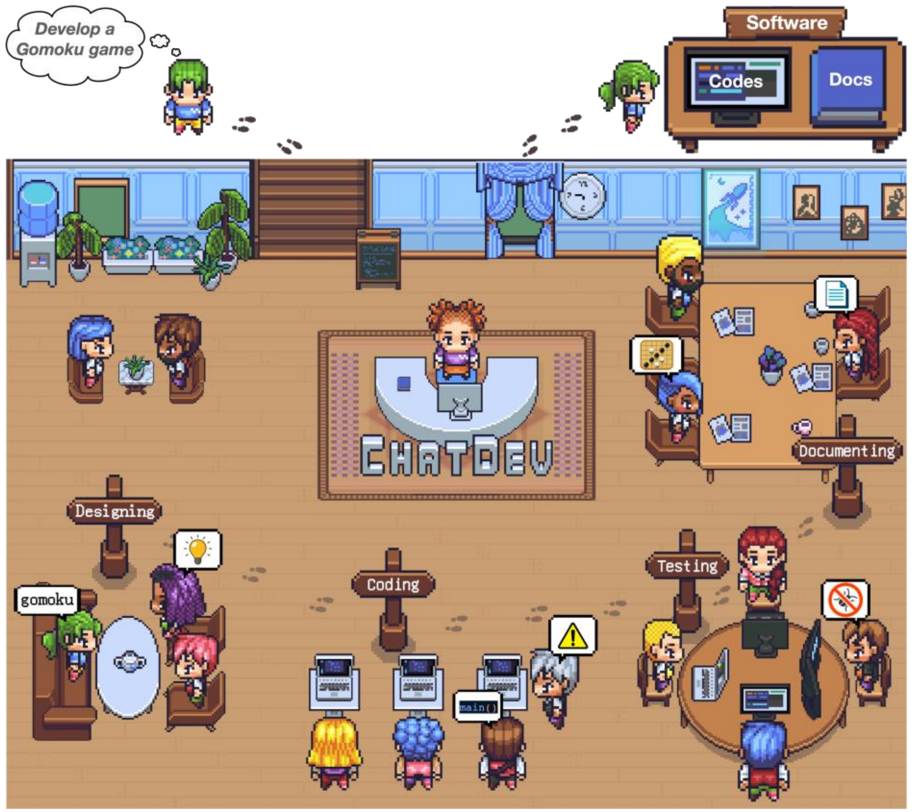
Correspondence: `mbuehler@MIT.EDU`

Abstract

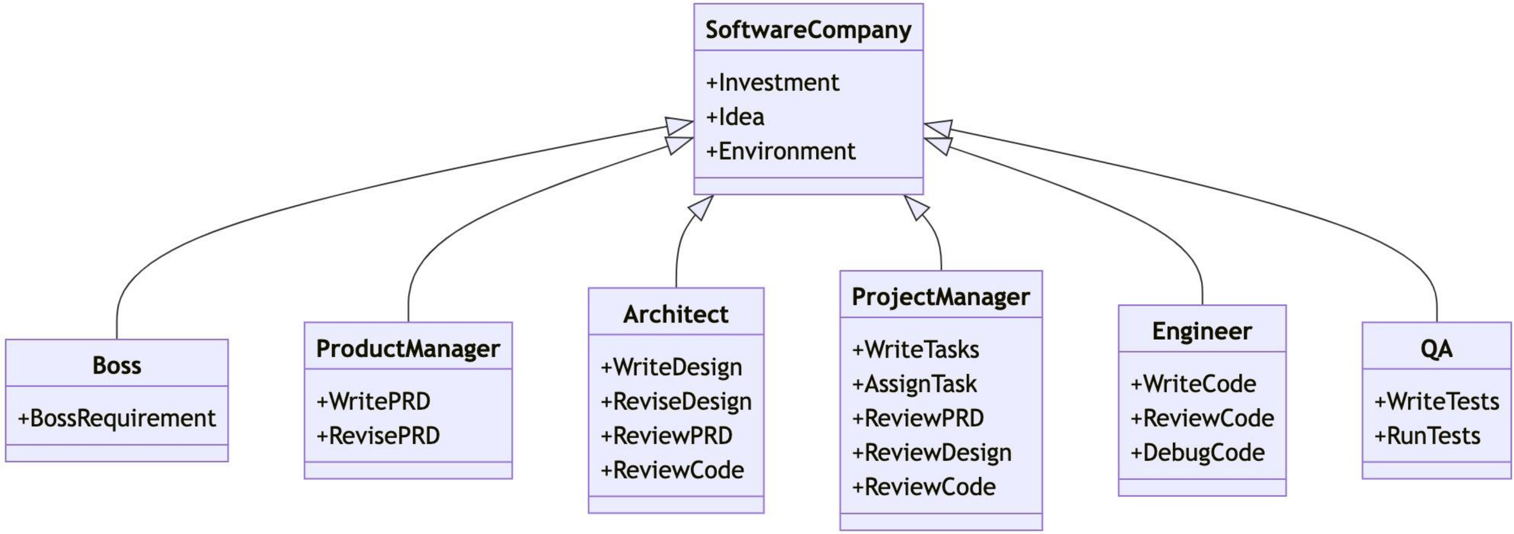
A key challenge in artificial intelligence is the creation of systems capable of autonomously advancing scientific understanding by exploring novel domains, identifying complex patterns, and uncovering previously unseen connections in vast scientific data. In this work, we present SciAgents, an approach that leverages three core concepts: (1) the use of large-scale ontological knowledge graphs to organize and interconnect diverse scientific concepts, (2) a suite of large language models (LLMs) and data retrieval tools, and (3) multi-agent systems with *in-situ* learning capabilities. Applied to biologically inspired materials, SciAgents reveals hidden interdisciplinary relationships that were previously considered unrelated, achieving a scale, precision, and exploratory power that surpasses traditional human-driven research methods. The framework autonomously generates and refines research hypotheses, elucidating underlying mechanisms, design principles, and unexpected material properties. By integrating these capabilities in a modular fashion, the intelligent system yields material discoveries, critique and improve existing hypotheses, retrieve up-to-date data about existing research, and highlights their strengths and limitations. Our case studies demonstrate scalable capabilities to combine generative AI, ontological representations, and multi-agent modeling, harnessing a 'swarm of intelligence' similar to biological systems. This provides new avenues for materials discovery and accelerates the development of advanced materials by unlocking Nature's design principles.

Keywords Scientific AI · Multi-agent system · Large language model · Natural language processing · Materials design · Bio-inspired materials · Knowledge graph · Biological design

Other multi-agent systems



ChatDev



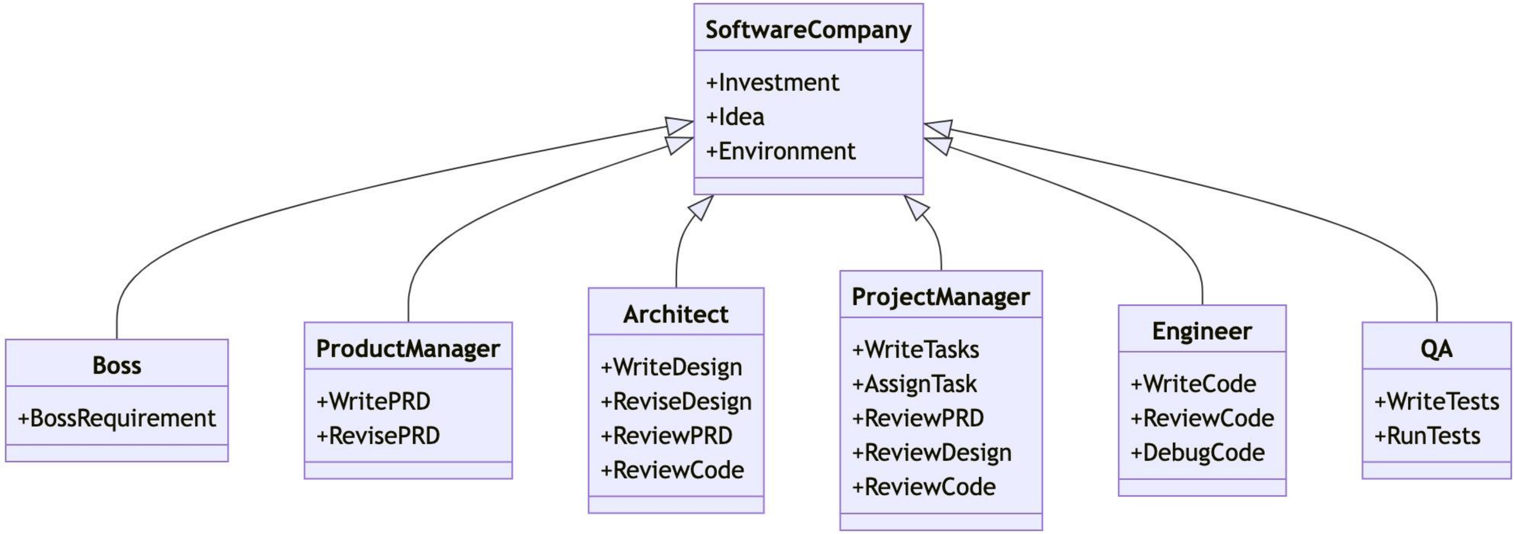
MetaGPT

Other multi-agent systems

Many solutions are more application/software engineering oriented. Lots research opportunities like

- Result interpretability and controllability
- Scalability
- Some guarantee & trustworthy AI
- Collaboration among RL- and LLM- agents

ChatDev



MetaGPT

Overview

- Server design
- Retrieval Augmented Generation (RAG)
- AI agents