CS 839 · FALL 2026 · LECTURE 1

AI Agents for Science

Course Overview

Fred Sala

University of Wisconsin–Madison

Logistics: Lecture Location

- **Tues. & Thurs.** · 9:30–10:45 AM
- **Mechanical Engineering 1164**
- Slides, blackboard, discussion, and in-class activities

> We meet in person twice each week.

Logistics: Enrollment

- Approximately **40 students**
- Teams of four for activities and projects
- Ten teams, each with a modest shared API budget

The class is designed to include **shared experimentation**.

Logistics: Instructor

- **Fred Sala**
- Office: 5514 Morgridge Hall
- Office hours: TBD

Logistics: Course Resources

1. **Course website**
   schedule, slides, policies, links
2. **Slack**
   all class announcements and questions · link TBD
3. **Canvas**
   assignments, submissions, grades

Course Content / Schedule

Part I · Sep–Oct 15 instructor lectures

Part II · Nov–Dec student-led presentations and project work

We will move from foundation models → agents → scientific workflows.

Thursday Sept. 4	Introduction and Course Overview			
Tuesday Sept. 9	Machine Learning Mini-Review	• Patterns, Predictions, and Actions		
Thursday Sept. 11	Architectures I: Transformers & Attention	• Attention Is All You Need • The Illustrated Transformer • Tokenization • RoFormer: Rotary Position Embedding (RoPE) • ALiBi: Train Short, Test Long		
Tuesday Sept. 16	Architectures II: Subquadratic Architectures	• Mamba: Linear-Time Sequence Modeling with Selective State Spaces • Mamba: The Hard Way • Retentive Network (RetNet) • Transformers are SSMs		
Thursday Sept. 18	Models I: Encoder-Decoder, Encoder-Only	• BERT Paper • RoBERTa Paper • T5 Paper	HW 1 Release	
Tuesday Sept. 23	Models II: Decoder-Only. Start Prompting I: Basics	• GPT-3 Paper • Llama 3.1 Paper		
Thursday Sept. 25	Prompting I: Basics, In-Context Learning, Chain-of-Thought	• Finetuned Language Models Are Zero-Shot Learners • Learning without training: The implicit dynamics of in-context learning • CoT • Large Language Models are Zero-Shot Reasoners		
Tuesday Sept. 30	Specialization: Fine-Tuning, Adaptation, Editing	• Low-Rank Adaptation of Large Language Models • Fast Model Editing at Scale • QLoRA: Efficient Finetuning of Quantized LLMs		HW 1 Due
Thursday Oct. 2	Alignment: RHLF, DPO, and friends	• Hugging Face RLHF Blog • Training language models to follow instructions with human feedback • Direct Preference Optimization: Your Language Model is Secretly a Reward Model • Constitutional AI: Harmlessness from AI Feedback • RLAIF: Scaling RLHF with AI Feedback	Presentation Info Release	
Tuesday Oct. 7	Reinforcement Learning with Verifiable Rewards (RLVR)	• RLVR (Chapter 3.2)	HW 2 Release	
Thursday Oct. 9	Efficient Training	• FlashAttention • Megatron-LM		

Logistics: Lecture Formats

Two types of class sessions:

Type 1: Lectures

- Mostly slides, some whiteboard
- Brief activities during class
- Questions during lecture and breaks

Type 2: Paper Presentations

- Student-led research discussions
- One team introduces the paper; everyone participates
- Presentations dominate the final third

> Start with Type 1; conclude the semester with Type 2.

Logistics: Grading (Tentative)

10%

Assignments

agent-building exercises

40%

Paper presentation

presentation and participation

50%

Final research project

graded through a group interview / discussion

Class Setup: Reading

- No textbook
- Research papers and system documentation
- Expect one paper before selected classes
- Presentations: choose from a list or propose an option

> **Read skeptically.** Separate a compelling demonstration from evidence of a reliable scientific capability.

Class Setup: Background

More on this at the end of class, but:

Basic ML

- At the level of CS 760 or so
- Short review next lecture

Technical components

- Linear algebra
- Calculus
- Probability
- Comfort reading research papers

> This class is partially conceptual and partially technical.

Class Setup: Goals

Two goals:

- Become acquainted with how to use large pretrained / language / foundation models as components of agents
- Understand the technical underpinnings of these models—and why agent systems succeed or fail

The object of study is the **whole system** : model, tools, memory, environment, and feedback.

Class Setup: Goals II

Mini-goals:

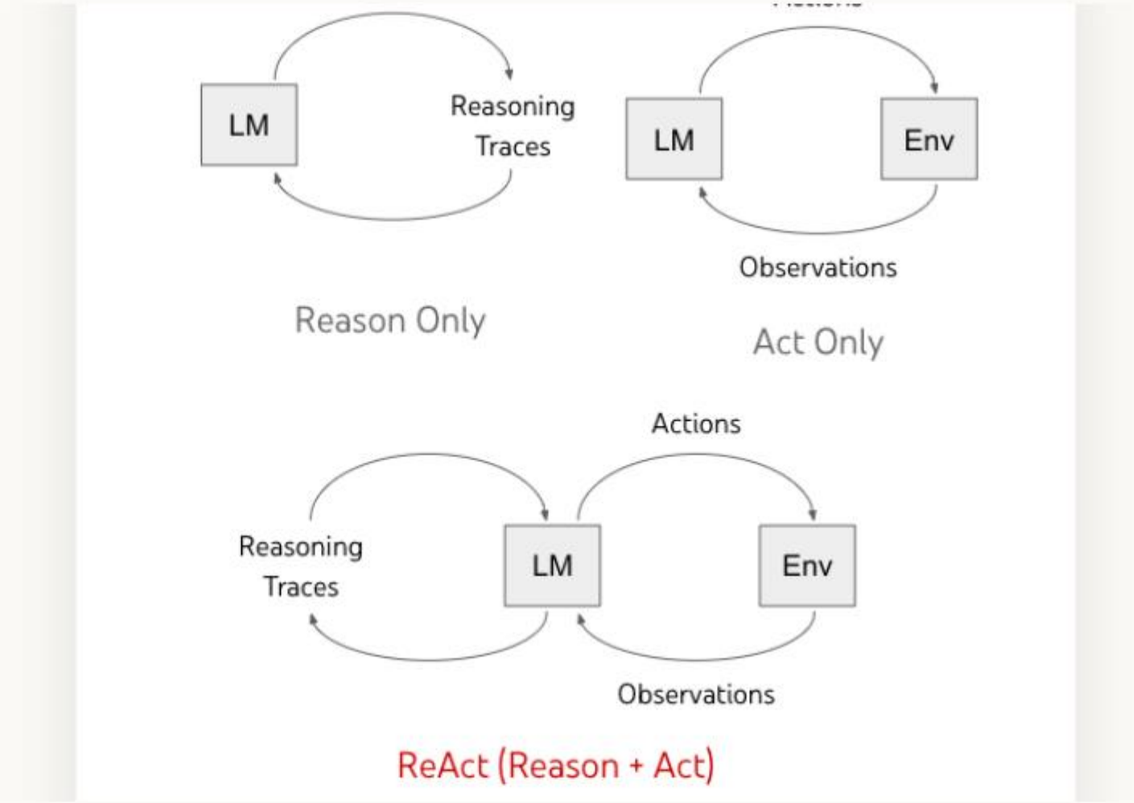
- Understand research
- See the big picture / ML ecosystem
- Build intuition around modern ML paradigms
- Learn to evaluate claims about autonomy

Break & Questions

Lecture 2: From Language Models to Agents

02

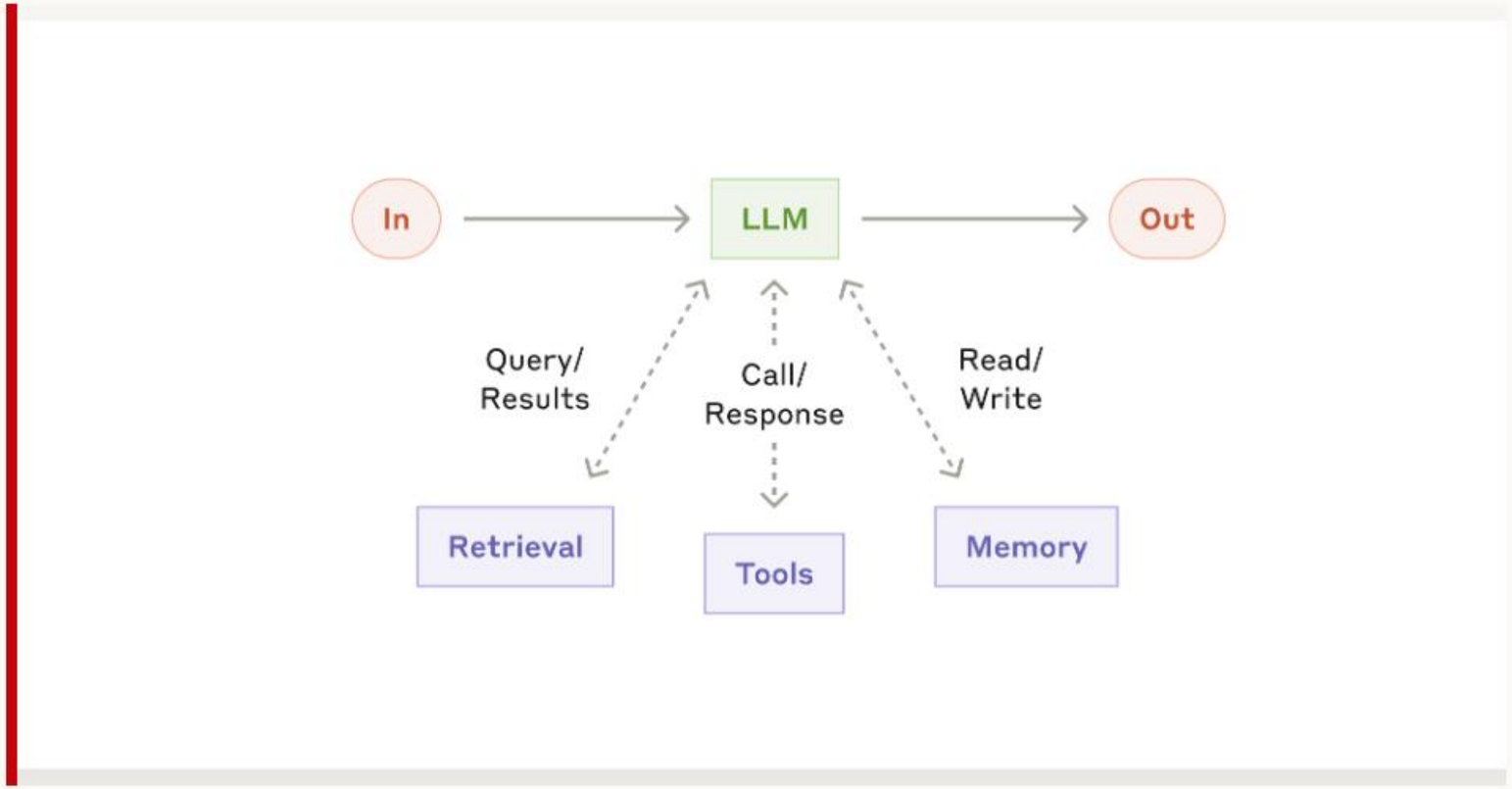
Instruction following, reasoning models, and agent loops.



Lecture 3: Agent Architectures and Frameworks

03

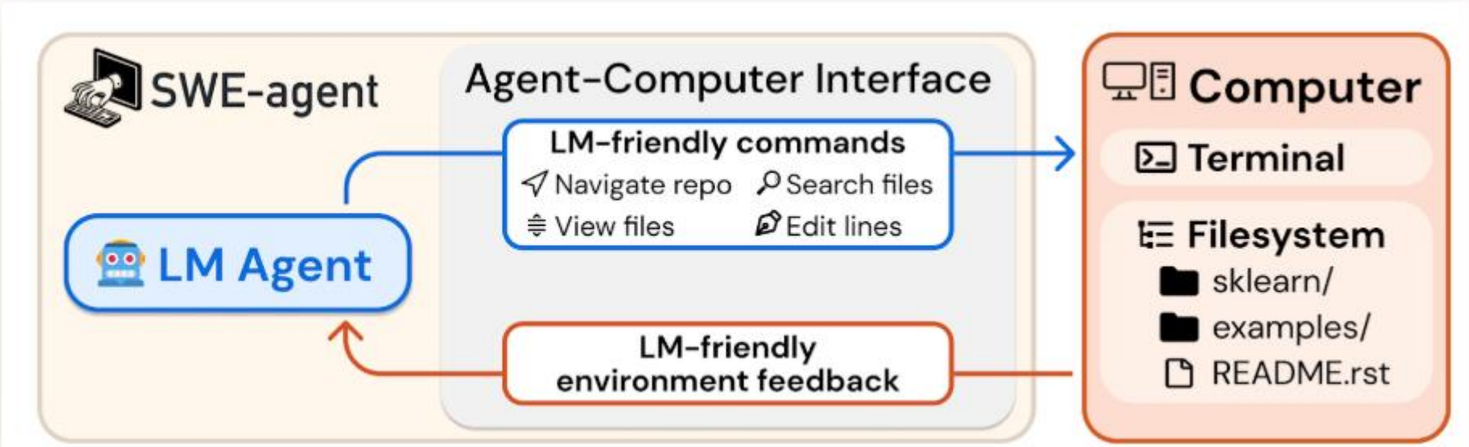
Orchestration, state, control flow, and debugging.



Lecture 4: Tools, Environments, and Agent–Computer Interfaces

04

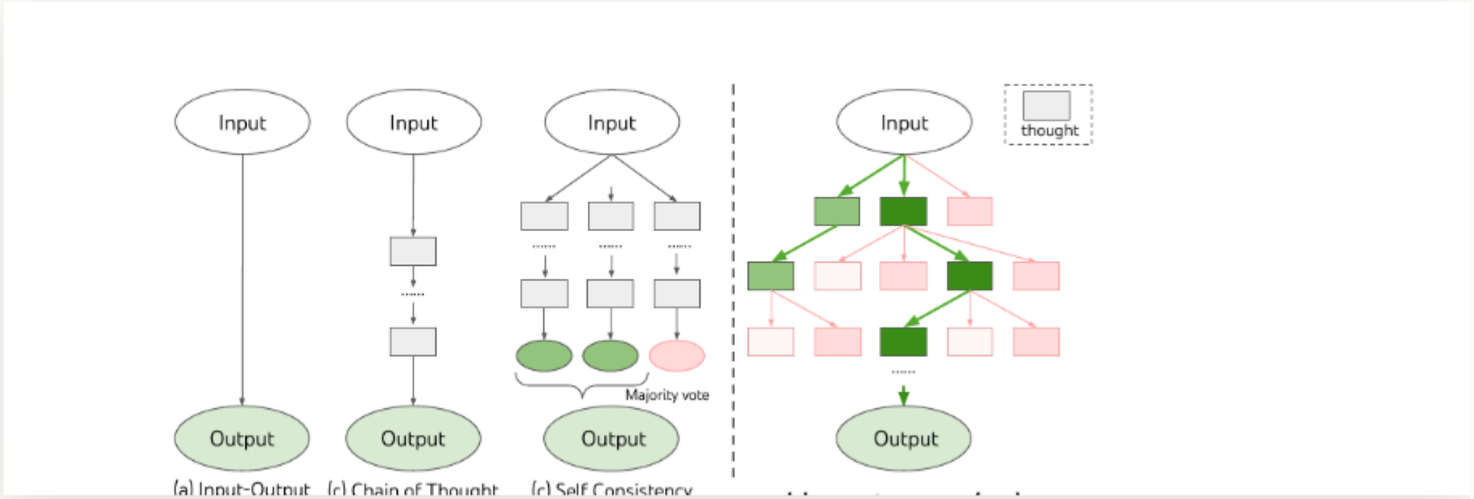
Function calling, code execution, scientific software, and error recovery.



Lecture 5: Planning and Task Decomposition

05

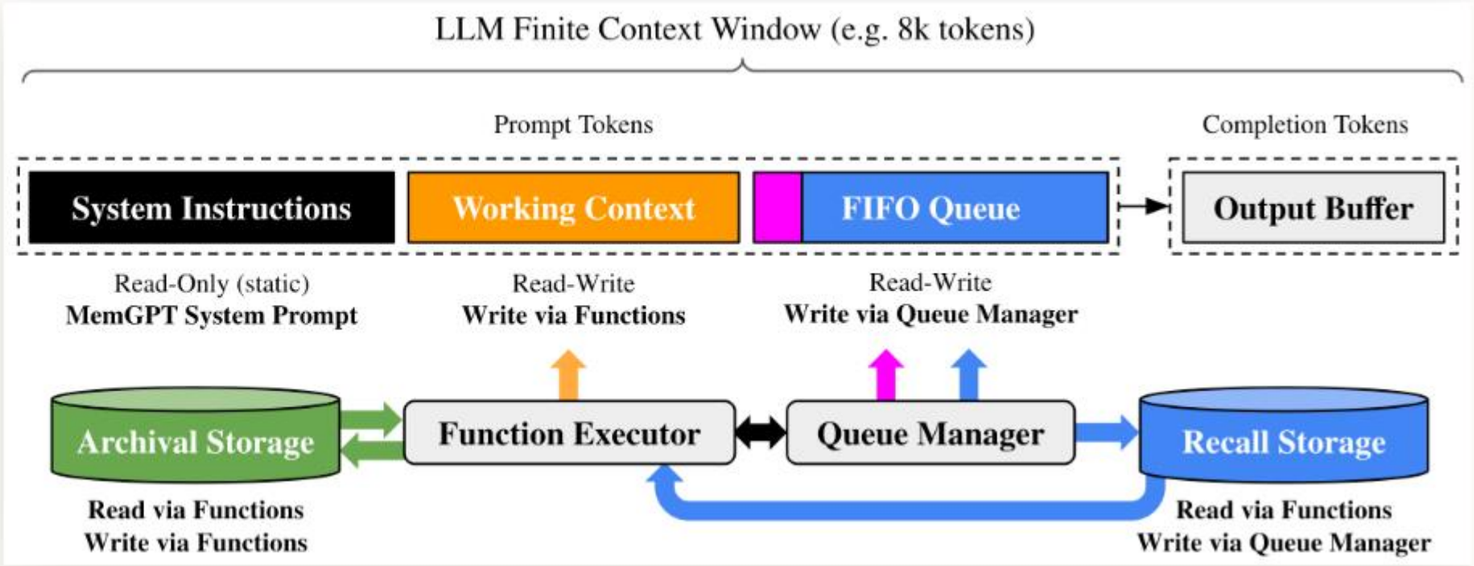
ReAct, search, reflection, verification, and long-horizon tasks.



Lecture 6: Memory, Context, and Retrieval

06

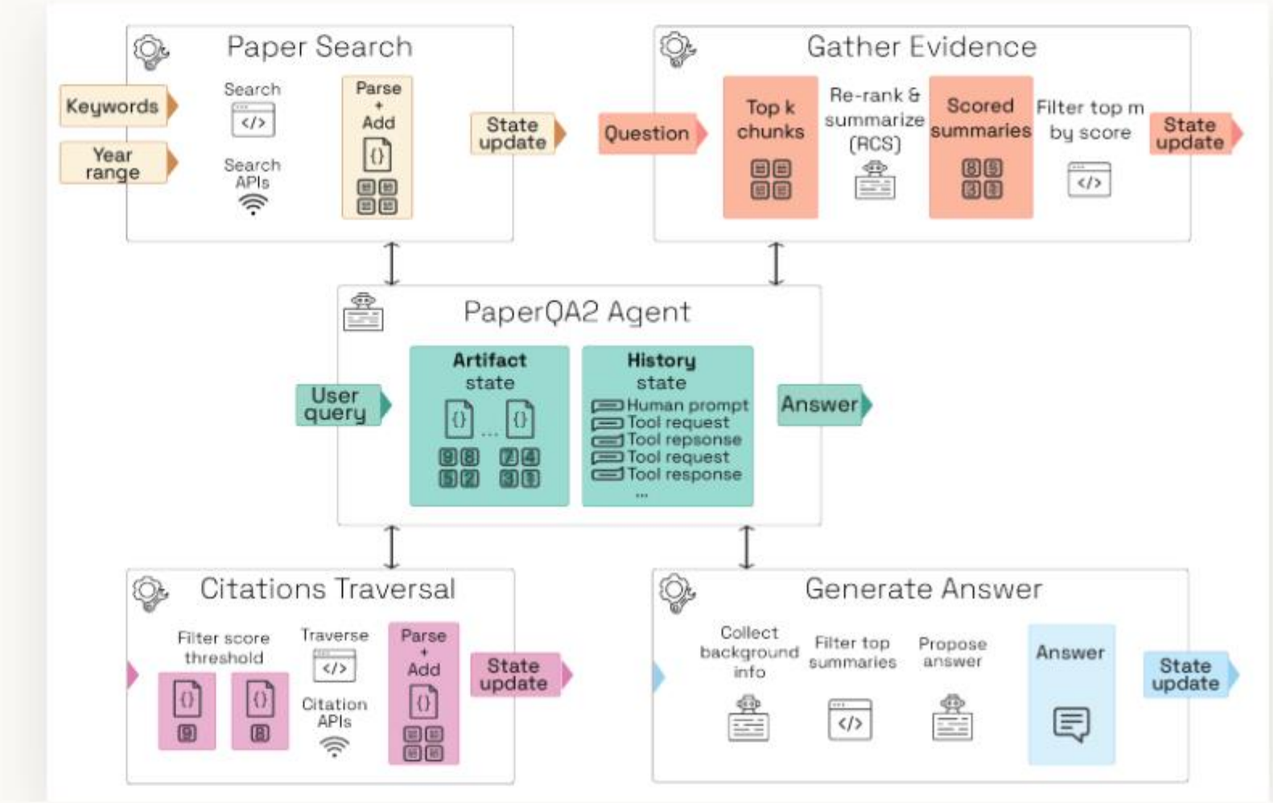
Persistent state, RAG, provenance, and related techniques.



Lecture 7: Agents for Scientific Literature and Evidence Synthesis

07

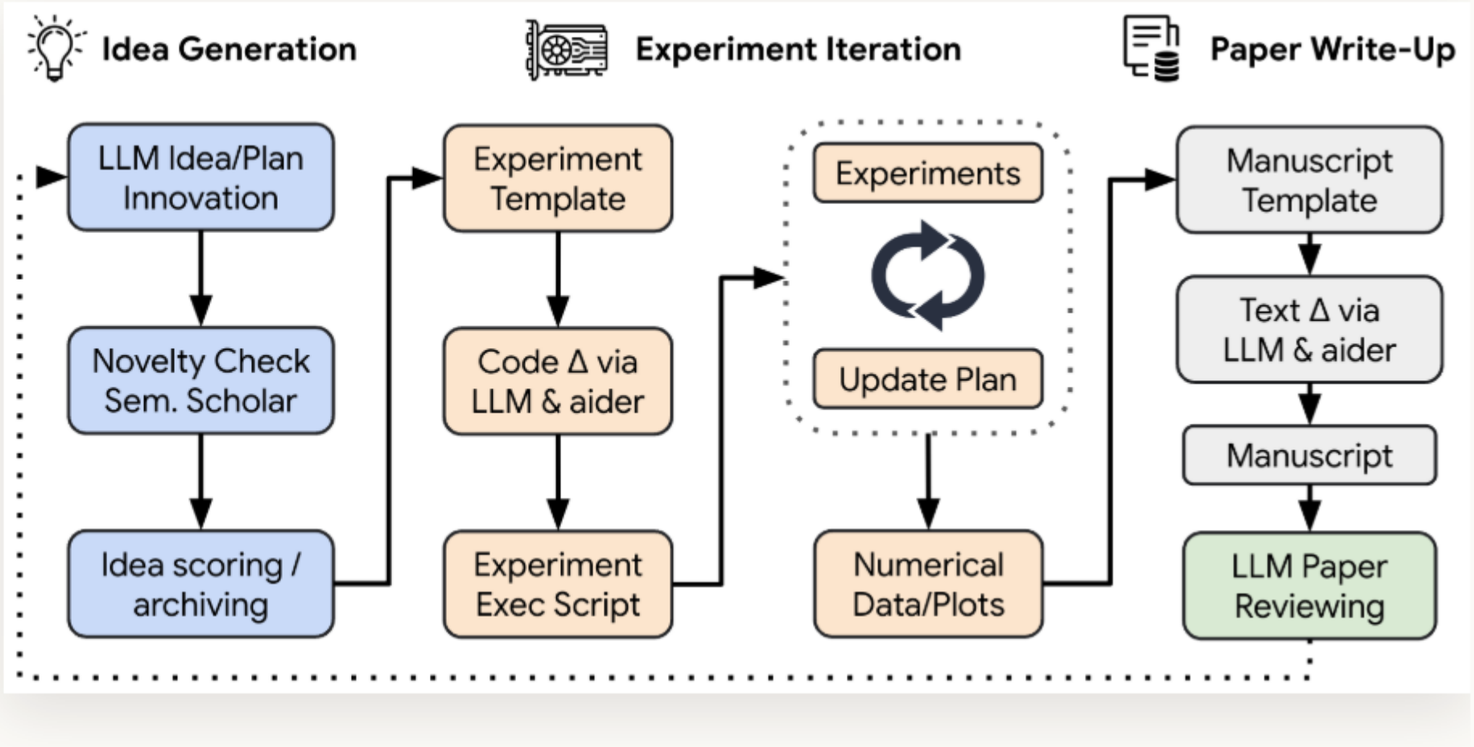
Search, citation verification, systematic reviews, and research gaps.



Lecture 8: Agents for Scientific Coding, Data Analysis, and Simulation

08

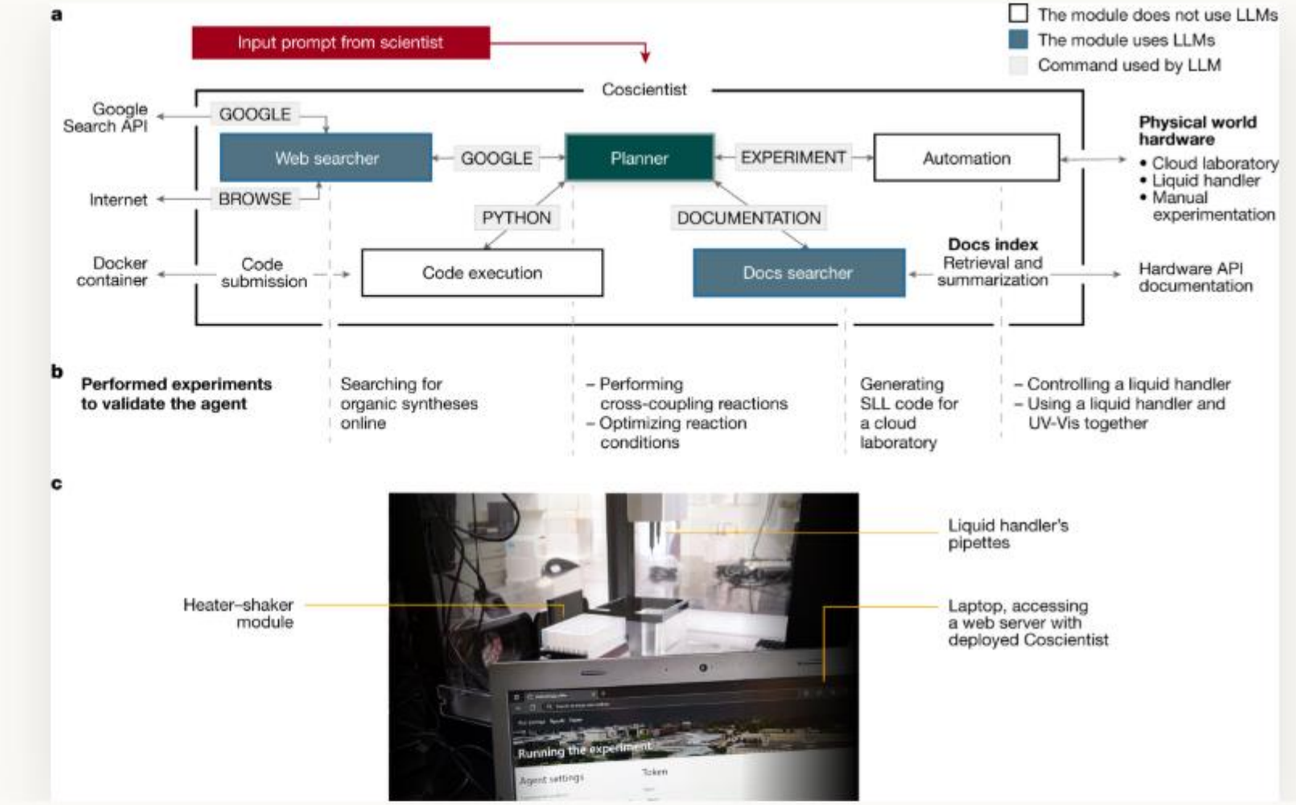
Analysis code, simulators, workflow debugging, and reproducibility.



Lecture 9: Agents for Hypothesis Generation and Experimental Design

09

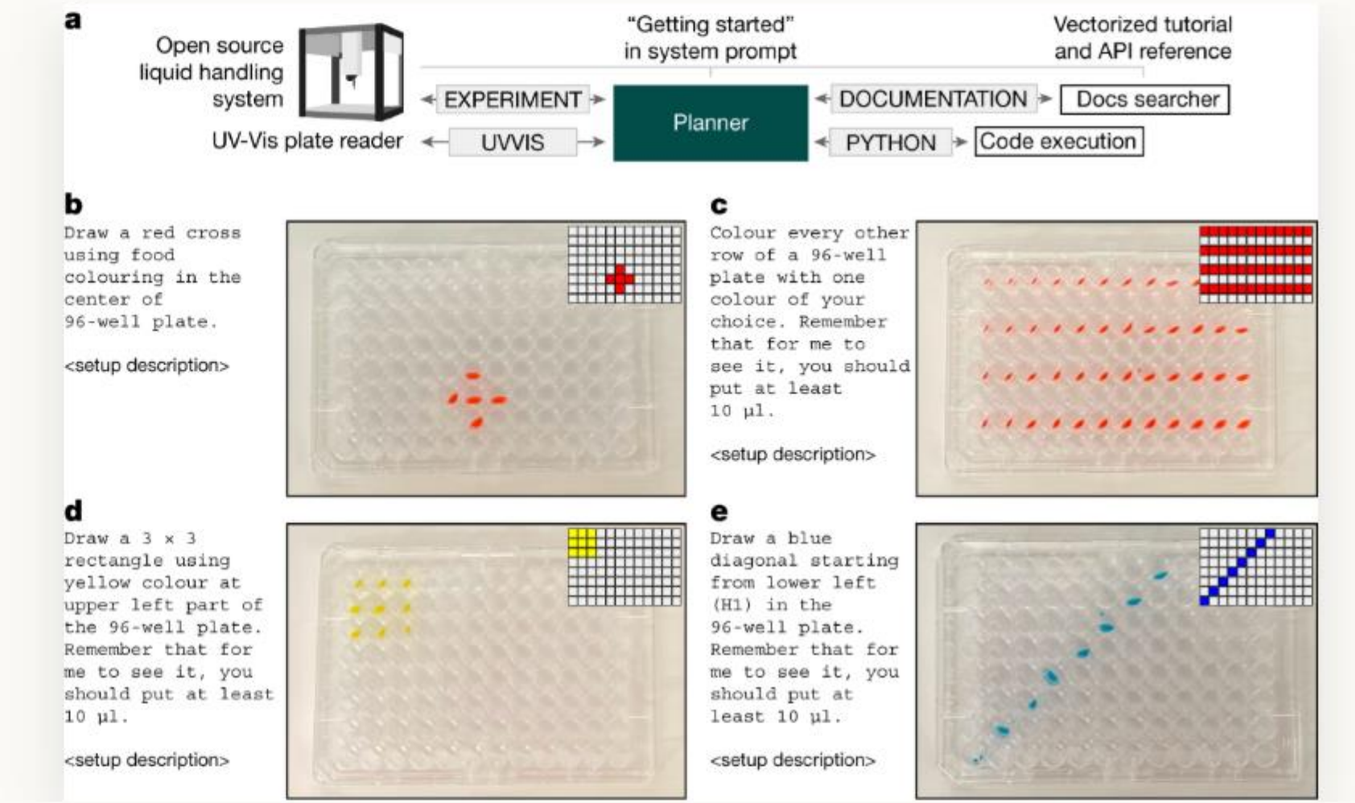
Hypotheses, experiment selection, active learning, and uncertainty.



Lecture 10: Auto-Laboratories and Autonomous Experimentation

10

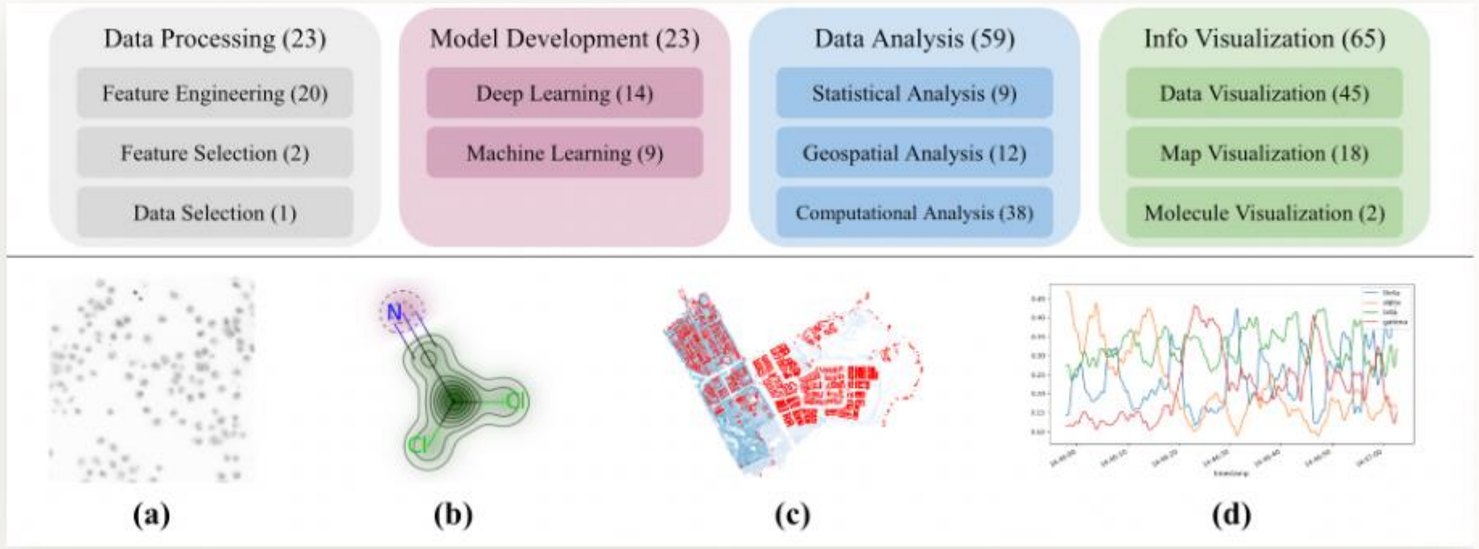
Robotics, closed-loop experimentation, costly actions, and approval.



Lecture 11: Evaluation and Benchmarks for Scientific Agents

11

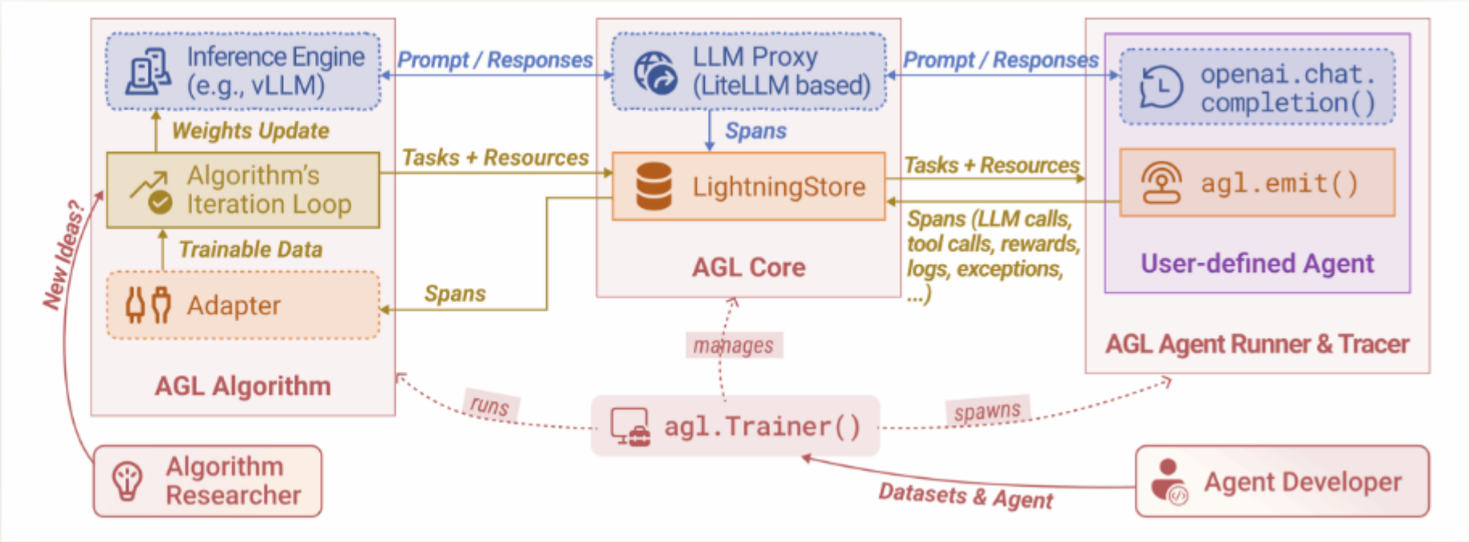
Scientific validity, baselines, reproducibility, cost, and discovery.



Lecture 12: Training and Improving Agents

12

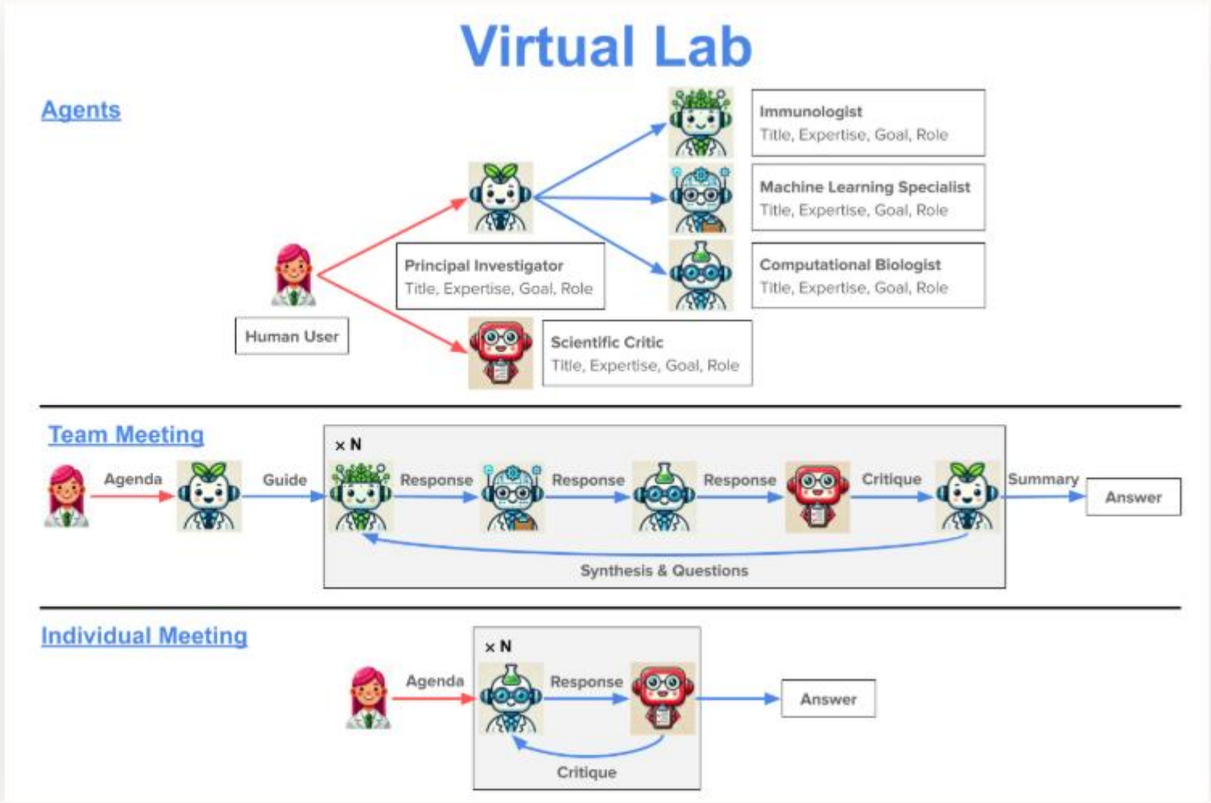
Trajectory data, process supervision, reinforcement learning, and credit assignment.



Lecture 13: Multi-Agent Scientific Discovery

13

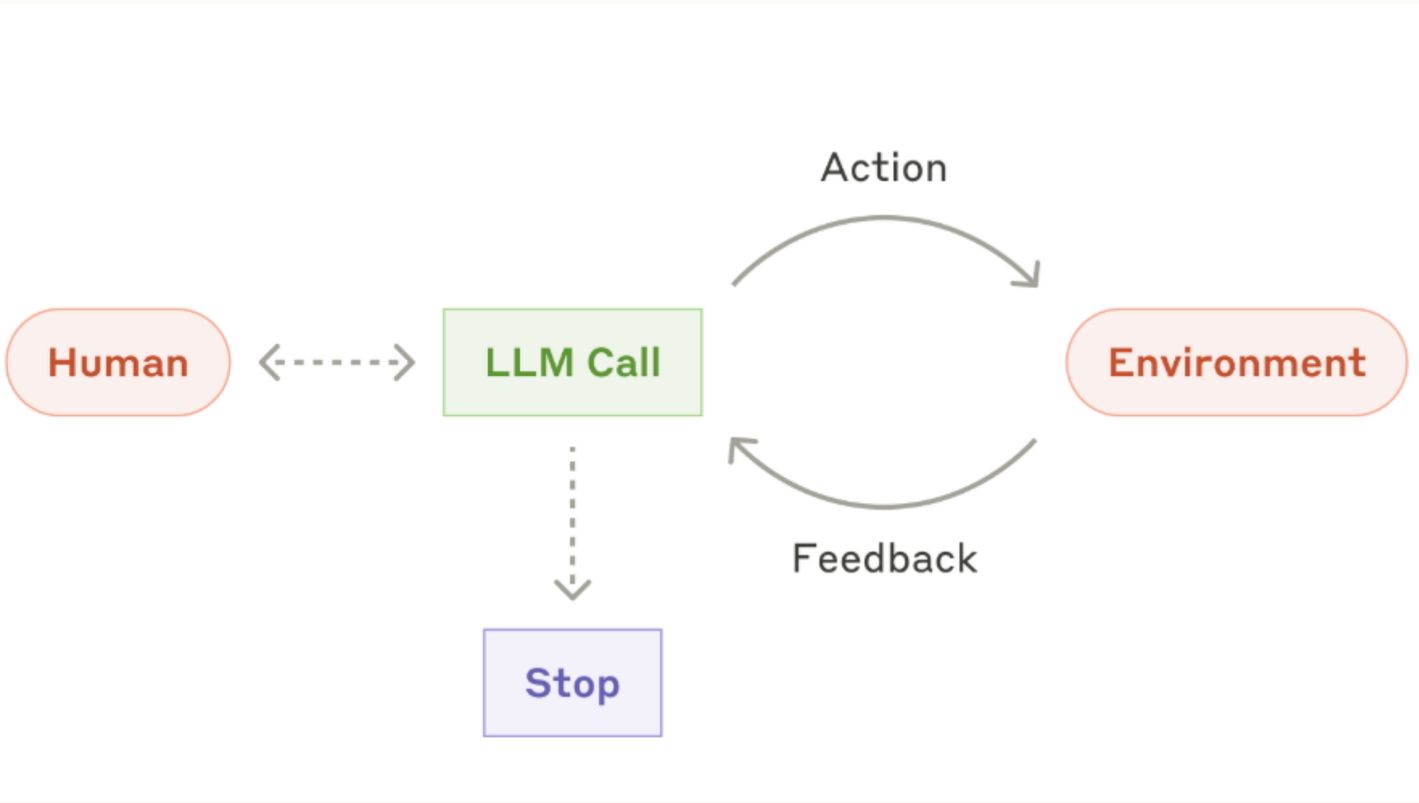
Specialized roles, debate, peer review, coordination, and failure.



Lecture 14: Human–Agent Interaction, Safety, and Scientific Integrity

14

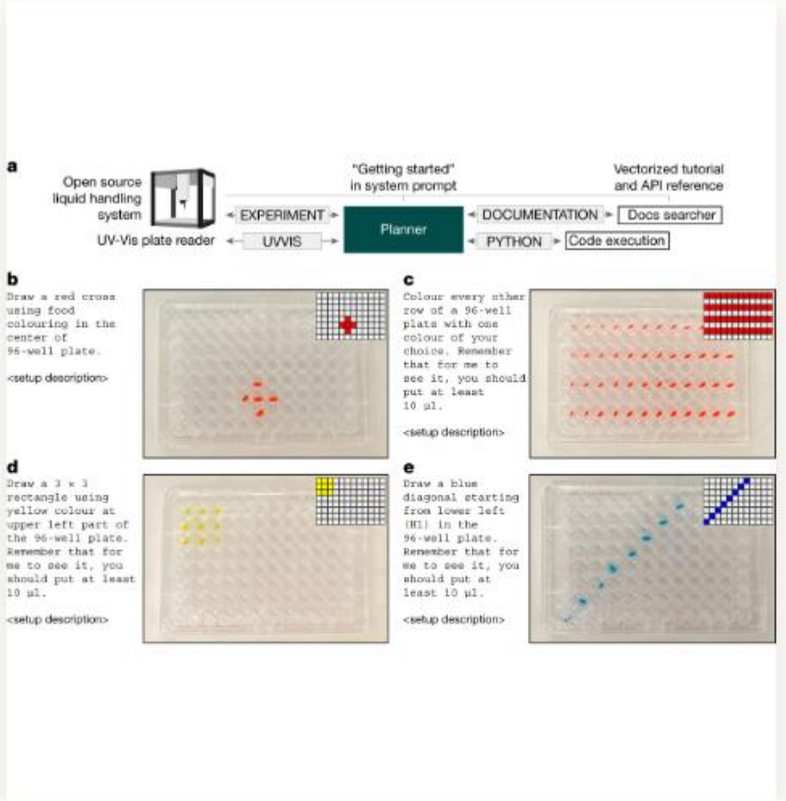
Oversight, sandboxing, credentials, evidence, attribution, and novelty.

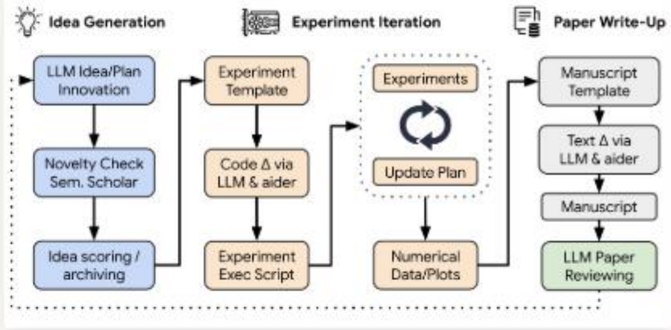


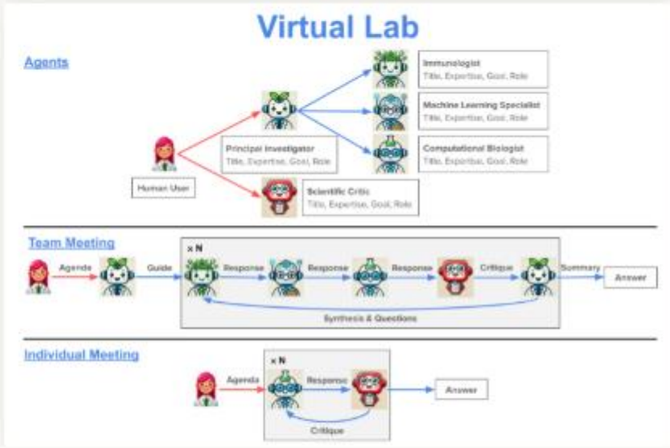
Lecture 15: Scientific Case Studies and Open Frontiers

15

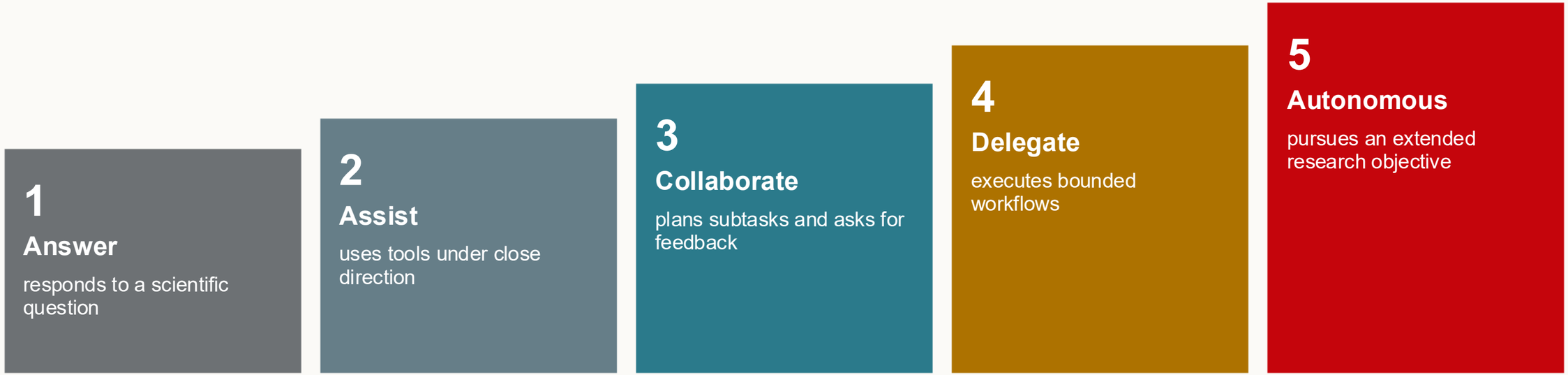
Biology, chemistry, materials, mathematics, climate, and new applications.







Levels of Autonomy: From LLMs to Fully Autonomous Agents



Autonomy: not a single number. Lots of thigns matter: scope, permissions, duration, oversight.

Representative Scientific Agent Systems

System	Domain	What it connects	Autonomy
ReAct	General	Reasoning + actions + observations	Short tool-use loops
Coscientist	Chemistry	Literature, code, hardware, optimization	Bounded laboratory tasks
AI Scientist	ML research	Ideas, code, experiments, writing, review	Long digital workflow
Virtual Lab	Biomedicine	PI agent, specialist agents, human feedback	Collaborative research team

Break & Questions

Brief History of LLM Agents

Three historical threads:

1. **Agents, planning, and tool use** — decades of ideas
2. **Pretraining and foundation models** — reusable language interface and world knowledge
3. **Reasoning + acting with feedback** — models embedded in environments

1. Agents Predate Language Models

- Planning and search
- Reinforcement learning
- Robotics and embodied control
- Multi-agent systems
- Software agents and workflow automation

The old question

How should an intelligent system choose actions in an environment?

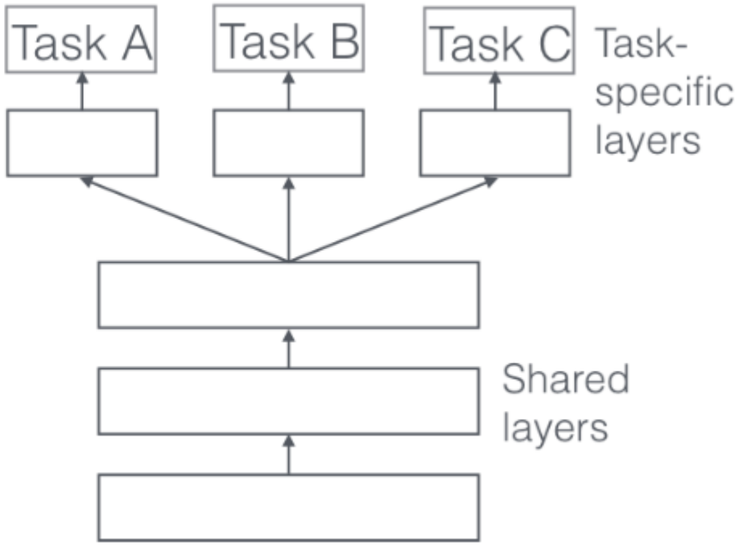
The new ingredient

A broadly capable language model can interpret goals and interfaces expressed in language.

2. Where did LLMs come from? (1) Multitask Models

- Given tasks T_1, …, T_k, train a common base with task-specific "heads"
- Related to transfer learning
- Usually fixed tasks; limited cross-task data

Shared representation: seed of a general-purpose model.



3. Where did LLMs come from? (2) Pretraining and Fine-tuning

Motivation: training from scratch is expensive.

- Larger datasets
- Larger hardware resources
- Larger models

Much of 2010–2015 computer vision scaled to larger CNNs

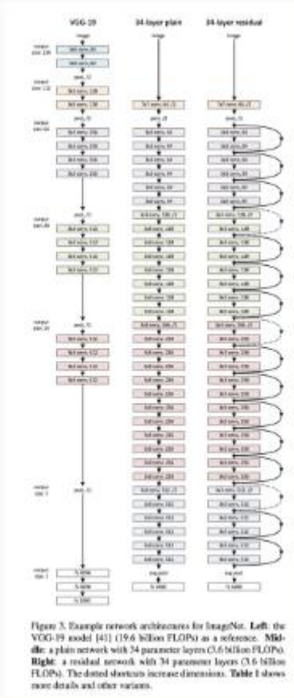


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

3. Where did LLMs come from? (2) Pretraining and Fine-tuning

Idea: pretrain a single model on a dataset, then fine-tune it for a downstream task.

- ImageNet pretraining becomes a standard recipe
- Self-supervised learning reduces the need for labels
- The same recipe transfers to language

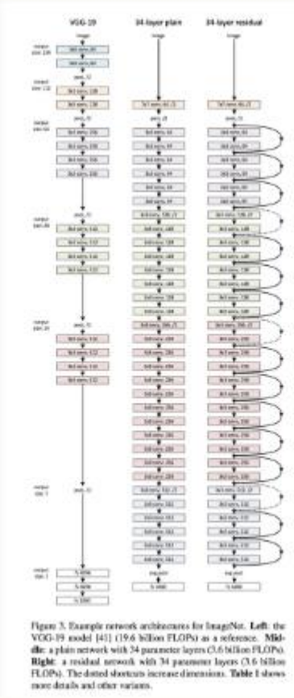


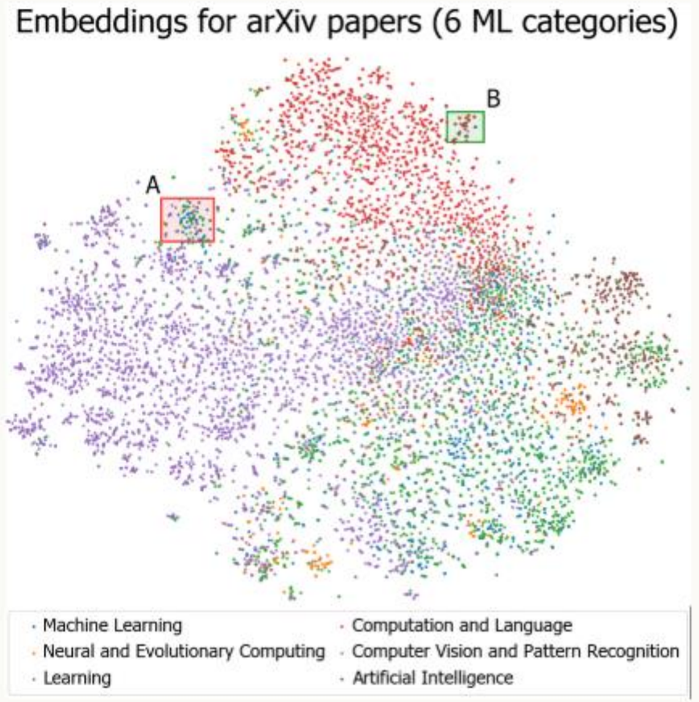
Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

4. Where did LLMs come from? (3) Word Embeddings

Motivation: deep learning advances—can they be applied to NLP?

Three areas:

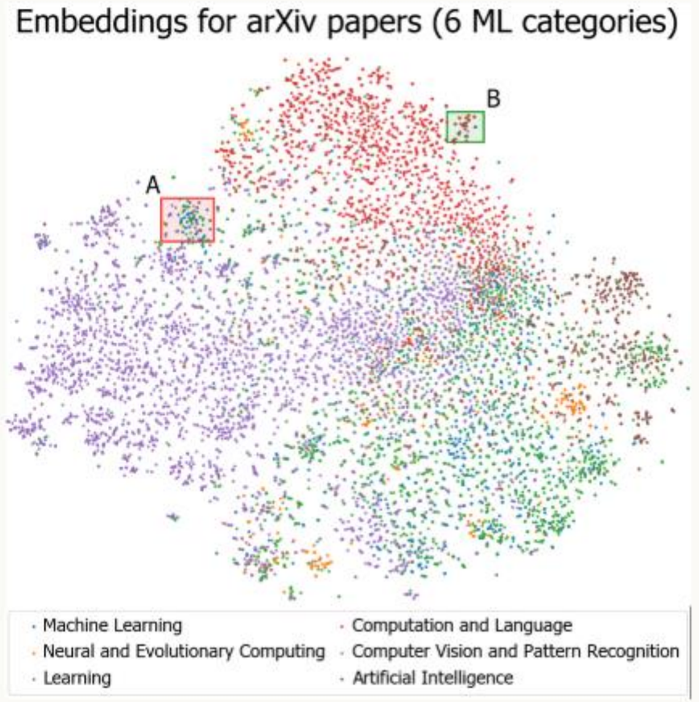
1. General: word embeddings
2. Specific: translation tasks
3. Specific: language modeling tasks



4. Where did LLMs come from? (3) Word Embeddings

- Learn advanced, structured representations of words
- Input to language-specific networks such as LSTMs
- Word embeddings: GloVe, word2vec, etc.

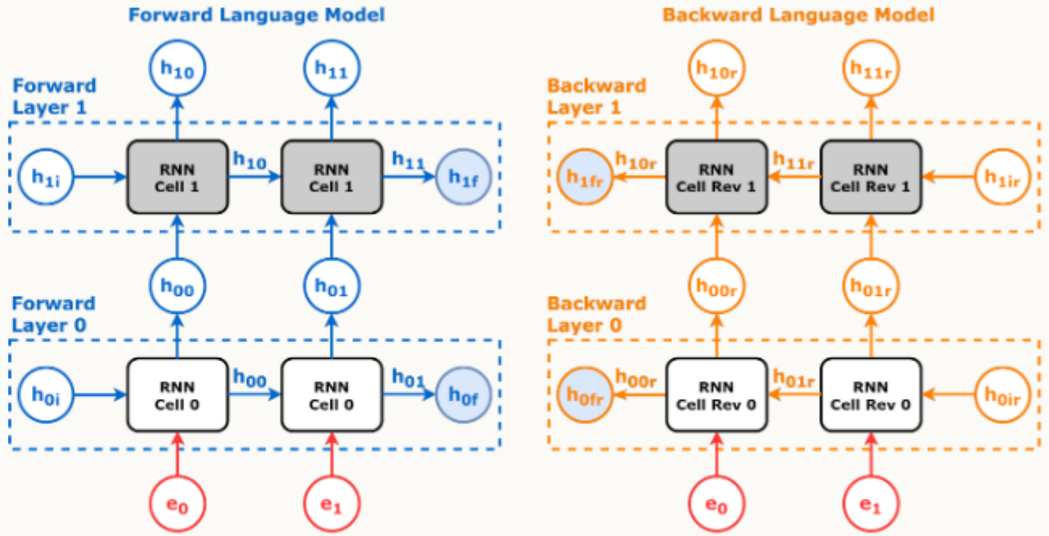
> Issue: static—no context used for words like “bank.”



4. Where did LLMs come from? (3) Word Embeddings

Solution: contextual word embeddings.

- Plug input into a model to obtain a context-dependent embedding
- Include context in both directions
- ELMo makes contextual representations practical

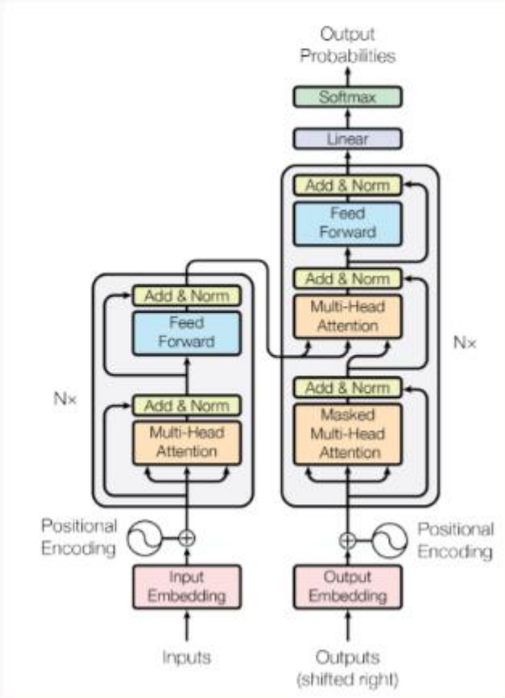


5. Where did LLMs come from? (4) Transformers

So far: embeddings, contextual or otherwise.

Translation is the critical task.

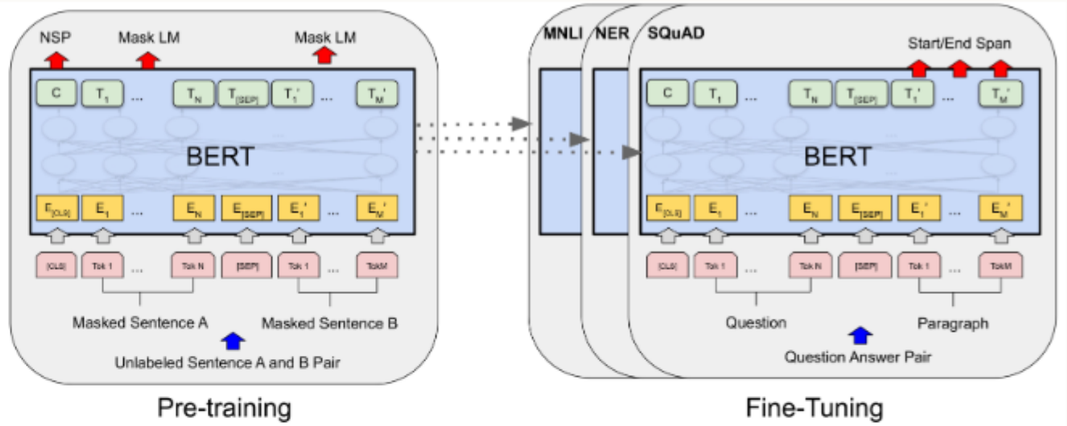
- Transformers arrive in 2017
- Attention replaces recurrent computation
- The architecture is unusually scalable



6. BERT and GPT

- Contextual embeddings + Transformer architecture
- BERT: pretraining followed by fine-tuning
- GPT: autoregressive language modeling
- Train on massive text corpora, then adapt through context

At this point, a language model can be a reusable component inside many systems.



7. Prompts Became Programs

Instruction → **Demonstrations** → **Reasoning** → **Tools** → **Memory**

state a goal → show behavior → decompose → change the world → carry state

Once a model can interpret interfaces expressed in language, adding a tool can add a capability.

8. Reasoning Was Connected to Action

Before

Generate an answer from the model’s current context.

Agent loop

Generate an action, observe the external result, update the plan, and continue.

Thought → Action → Observation → Thought → …

9. Digital Environments Made Actions Cheap

Web

Search, browse, retrieve documents.

Code

Run experiments, simulations, and tests.

Science

Query databases, analyze data, control instruments.

The environment turns language-model output into observable consequences.

10. Representative Systems

2022	→	2023	→	2024	→	2025
ReAct		Coscientist		AI Scientist		Virtual Lab

The direction is toward longer horizons, more heterogeneous tools, and more consequential environments.

Summary

- **Modern agents** build on old ideas about planning, environments, and feedback
- **Foundation models** provide a reusable language-and-code interface
- Prompts, tools, memory, and interaction traces **turn models into systems**
- Scientific agents connect this loop to evidence, experiments, and real-world consequences

> **Central question for the semester:** when does workflow automation become reliable scientific assistance?

Next Two Lectures

2.

From Language Models to Agents

Interface failures, reasoning models, and agent loops.

3.

Agent Architectures and Frameworks

Tools, control flow, orchestration, memory, and debugging.