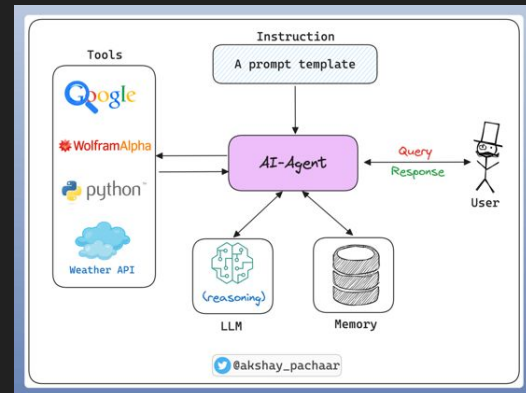


Orchestration/Specialization

Presented by Muhammad Muaz, Daniel Young

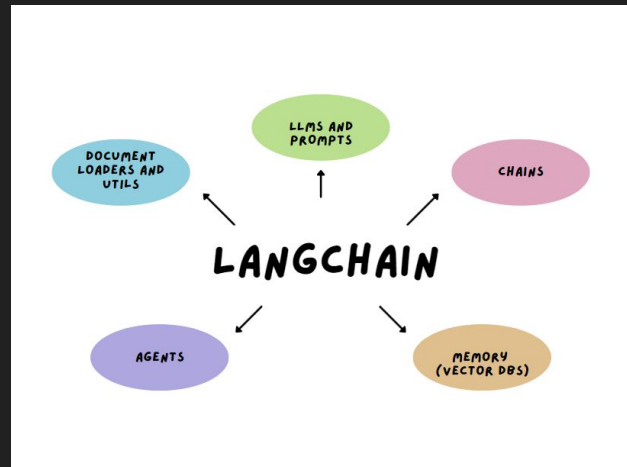
Motivation

- **Case study:** Imagine a disaster happening somewhere and relief agencies need answers (quick!):
 - **Information gathering** → weather updates, satellite images, social media etc.
 - One LLM can't handle all modalities
 - **Orchestration of agents** needed to route tasks to right models
 - **Knowledge Sythesis** → Daily situation reports for coordination
 - Data → scattered across news
 - **STORM-style system** retrieves, structures, and writes grounded summaries
 - **Team Coordination** → Logistics, flood monitoring, hospital support
 - Should agents be **generalists or specialists?**



LangChain

- Open Source Framework for developing LLM powered applications
- Allows to streamline development process using pre-built modules and utilities
 - Makes it easier to integrate LLMs for tasks like data retrieval, workflow automation, etc.
- Plug-n-play fashion of components (**modularity**):
 - Document loaders
 - Embedding models
 - Databases (such as ChromaDB)



Credit: <https://shurutech.com/getting-started-langchain-with-examples/>

Core Components

- **Models + prompt templates**
 - Wrappers around LLM providers (e.g. OpenAI, Anthropic, Cohere)
 - Reusable, parameterized prompts (placeholders for dynamic inputs, nesting templates)
- **Chains (one of central components)**
 - Sequence of steps that automate workflow involving one or more LLM calls
 - **LLMChain** (Basic unit to prompt a model); **SequentialChain** (composition of multiple LLMChains in sequential order); **RouterChain** (selects chains dynamically based on inputs)
- **Tools + Agents**
 - **Tools:** Functions exposed to agent e.g., calculators, search tools, dB queries
 - **Agents:** Decision-making LLMs that determine which tools to use + order

Core Components (contd.)

- Memory
- Callbacks and tracing
- Document loaders and text splitters
- Caching (and many more...)
- LangChain Hub (<https://smith.langchain.com/hub/>)
 - Library of open source chains + prompts



Buffer Memory

Stores full interaction history



Summary Memory

Summarizes past convos

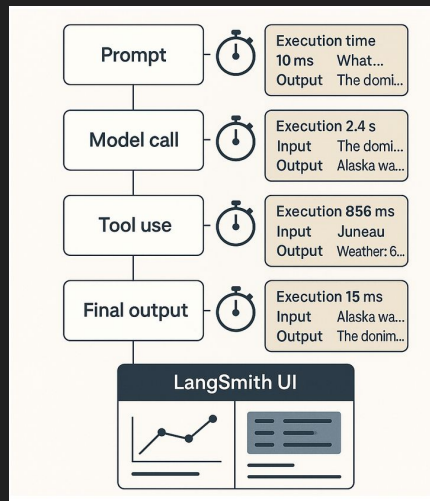
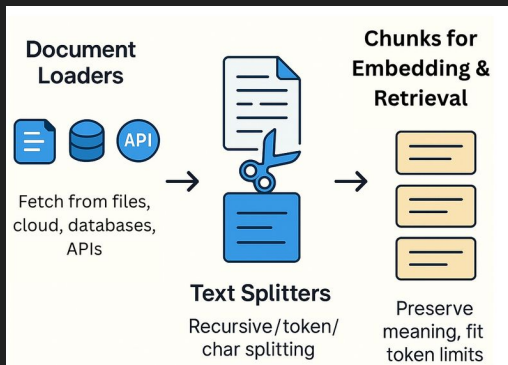


Entity Memory

Tracks people/things mentioned



Plug into Chains & Agents



Credit:

<https://www.educative.io/blog/langchain-components>

LangGraph

<https://blog.langchain.com/langgraph/>
<https://github.com/langchain-ai/langchain>
<https://blog.langchain.com/how-to-build-an-agent/>

- Module built on top of LangChain to better enable creation of cyclical graphs
 - Needed for agent runtimes
- Chains → Directed Acyclic Graphs (**DAGs**)
 - Complex LLM applications → introduces cycles into runtime
 - Often use LLM to 'reason' about what next in cycle?
 - Basically, LLM in a for-loop! (Simple agent)
- Programmatically,
 - StateGraph (Nodes, Edges)
 -

```
from typing import TypedDict, Annotated, List, Union
from langchain_core.agents import AgentAction, AgentFinish
from langchain_core.messages import BaseMessage
import operator

class AgentState(TypedDict):
    input: str
    chat_history: list[BaseMessage]
    agent_outcome: Union[AgentAction, AgentFinish, None]
    intermediate_steps: Annotated[list[tuple[AgentAction, str]], op
```



LangGraph

<https://blog.langchain.com/langgraph/>
<https://github.com/langchain-ai/langchain>
<https://blog.langchain.com/how-to-build-an-agent/>

- **Benefits:**
 - **Durable agents** – that persist through failure + run for extended times
 - **Human-in-the-loop** – incorporate human oversight by inspecting + modifying agent state
 - **Comprehensive memory** – Stateful agents with both short-term working memory + long-term persistent memory across session

LangGraph – Weather Agent Example

```
# Import relevant functionality
from langchain.chat_models import init_chat_model
from langchain_tavily import TavilySearch
from langgraph.checkpoint.memory import MemorySaver
from langgraph.prebuilt import create_react_agent

# Create the agent
memory = MemorySaver()
model = init_chat_model("anthropic:claude-3-5-sonnet-latest")
search = TavilySearch(max_results=2)
tools = [search]
agent_executor = create_react_agent(model, tools, checkpoint=memory)
```

```
input_message = {
    "role": "user",
    "content": "What's the weather where I live?",
}

for step in agent_executor.stream(
    {"messages": [input_message]}, config, stream_mode="values"
):
    step["messages"][-1].pretty_print()
```

=====1m Human Message [0m=====

What's the weather where I live?

=====1m Ai Message [0m=====

[{'text': 'Let me search for current weather information in San Francisco.', 'type': 'text'}, {'id': 't
Tool Calls:

tavily_search (toolu_011kSdheoJp8THURoLmeltZo)

Call ID: toolu_011kSdheoJp8THURoLmeltZo

Args:

query: current weather San Francisco CA

=====1m Tool Message [0m=====

Name: tavily_search

{ "query": "current weather San Francisco CA", "follow_up_questions": null, "answer": null, "images": []

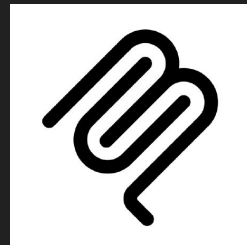
=====1m Ai Message [0m=====

Based on the search results, here's the current weather in San Francisco:

- Temperature: 53.1°F (11.7°C)
- Condition: Foggy
- Wind: 4.0 mph from the Southwest
- Humidity: 86%
- Visibility: 9 miles

This is quite typical weather for San Francisco, with the characteristic fog that the city is known for

Model Context Protocol (MCP)



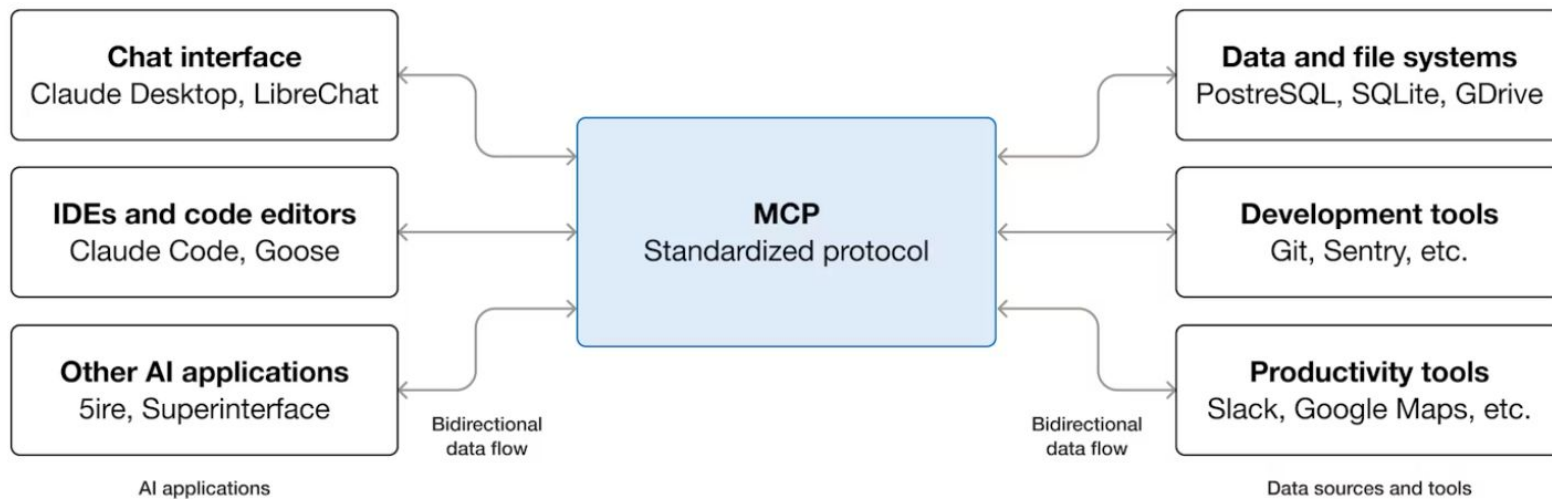
- From Anthropic – Introduced in 25 Nov 2024
- Open source standard for connecting LLM agents to external tools
 - Data sources (local files, dBs)
 - Tools (Search engines, calculators)
- **Why was it introduced?**
 - Prior to MCP, to connect a new data source requires its own custom implementation → fragmented integrations → systems difficult to scale



Model Context Protocol (MCP)

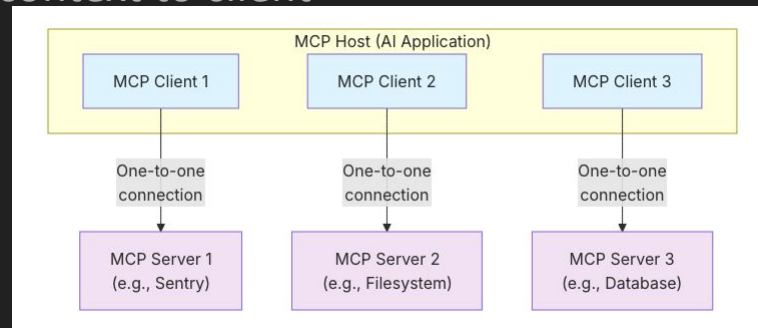
- From Anthropic – Introduced in 25 Nov 2024
- Open source standard for connecting LLM agents to external tools
 - Data sources (local files, dBs)
 - Tools (Search engines, calculators)
- **Why was it introduced?**
 - Prior to MCP, to connect a new data source requires its own custom implementation → fragmented integrations → systems difficult to scale
- **Basically, MCP = USB-C port for AI Applications!**

Model Context Protocol (MCP)



Architecture

- Client-Server Arch. – AI application (Claude Code) establishes connections with 1+ servers via creating an MCP client for each MCP server
 - **MCP Host:** AI application that coordinates + manages 1+ clients
 - **MCP Client:** component that maintains connections with MCP server + obtains context from server for the host to use
 - **MCP Server:** program that provides context to client



Overview of MCP Server

- Programs that expose specific capabilities to AI
 - E.g., filesystem servers, database servers, slack servers
- Core features:
 - **Tools** - Functions that LLM can actively call
 - `searchFlights(origin: "NYC", destination: "Barcelona", date: "2024-06-15")`
 - **Resources** - Passive data sources that provide read-only access to information for context
 - **Prompt** – Pre-built instruction templates that tell model to work with tools

```
{
  name: "searchFlights",
  description: "Search for available flights",
  inputSchema: {
    type: "object",
    properties: {
      origin: { type: "string", description: "Departure city" },
      destination: { type: "string", description: "Arrival city" },
      date: { type: "string", format: "date", description: "Travel date" }
    },
    required: ["origin", "destination", "date"]
  }
}
```

Repository of MCP Servers

The screenshot shows the GitHub repository page for `modelcontextprotocol/servers`. The repository is public and has 67.9K stars. It includes a search bar, navigation tabs for Code, Issues, Pull requests, Actions, and Releases. The file list on the left shows various directories and files, including `.github`, `.vscode`, `scripts`, `src`, `.gitattributes`, `.gitignore`, `.npmrc`, `CODE_OF_CONDUCT.md`, `CONTRIBUTING.md`, `LICENSE`, `README.md`, `SECURITY.md`, `package-lock.json`, `package.json`, and `tsconfig.json`. The right sidebar displays the repository's README, which lists various integrations and third-party servers.

modelcontextprotocol / servers · 67.9K ✓

Code Issues 180 Bugs 86 Pull requests 54 Actions Releases 19

servers Public

Watch 490 Fork 8k Starred 67.9k

main 30 Branches 35 Tags Go to file Add file Code

3 people Add Hive Intelligence MCP (#2646) 11f6f34 · 18 hours ago 3,498 Commits

<code>.github</code>	fix: only run scheduled release pipeline on modelcontextp...	3 weeks ago
<code>.vscode</code>	add new mcp debugger and client [OpenMCP Client]	2 months ago
<code>scripts</code>	new release workflow	8 months ago
<code>src</code>	docs: Fix duplicate directory_tree tool entry in API docum...	3 days ago
<code>.gitattributes</code>	Initial commit	10 months ago
<code>.gitignore</code>	* in .gitignore	2 weeks ago
<code>.npmrc</code>	Initial commit	10 months ago
<code>CODE_OF_CONDUCT.md</code>	Initial commit	10 months ago
<code>CONTRIBUTING.md</code>	Reorganize CONTRIBUTING.md: move MCP protocol feat...	3 weeks ago
<code>LICENSE</code>	Update copyright year in LICENSE (#1874)	last month
<code>README.md</code>	Add Hive Intelligence MCP (#2646)	18 hours ago
<code>SECURITY.md</code>	doc: update the SECURITY.md to better reflect the scope...	3 months ago
<code>package-lock.json</code>	Let Everything's STUDIO server use SDK's automatic log le...	4 days ago
<code>package.json</code>	update	4 months ago
<code>tsconfig.json</code>	Create package for each server	10 months ago

Model Context Protocol Servers

modelcontextprotocol.io

Readme MIT license Code of conduct Contributing Security policy Activity Custom properties 67.9k stars 490 watching 8k forks 10 months old

Release 2025.8.4 Latest on Aug 5

Contributors 753

Deployments 500+ release 4 days ago

TypeScript 54.2% Python 23.5%

The screenshot shows the README file for the `modelcontextprotocol/servers` repository. It lists various integrations and third-party servers, including Google Maps, PostgreSQL, Puppeteer, Redis, Sentry, Slack, and SQLite. It also includes a section for Third-Party Servers and Official Integrations, which lists various AI agents and services that can be integrated with the MCP servers.

README Code of conduct Contributing MIT license Security

- [Google Maps](#) - Location services, directions, and place details.
- [PostgreSQL](#) - Read-only database access with schema inspection.
- [Puppeteer](#) - Browser automation and web scraping.
- [Redis](#) - Interact with Redis key-value stores.
- [Sentry](#) - Retrieving and analyzing issues from Sentry.io.
- [Slack](#) - Channel management and messaging capabilities. Now maintained by [Zencoder](#)
- [SQLite](#) - Database interaction and business intelligence capabilities.

Third-Party Servers

Official Integrations

Official integrations are maintained by companies building production ready MCP servers for their platforms.

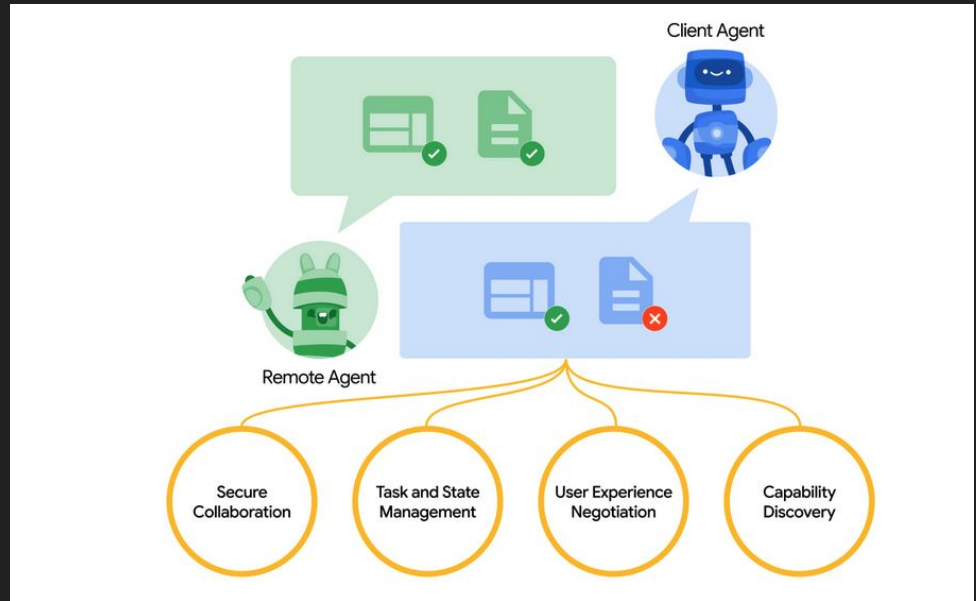
- [21st.dev Magic](#) - Create crafted UI components inspired by the best 21st.dev design engineers.
- [ActionKit by Paragon](#) - Connect to 130+ SaaS integrations (e.g. Slack, Salesforce, Gmail) with Paragon's [ActionKit](#) API.
- [Adfin](#) - The only platform you need to get paid - all payments in one place, invoicing and accounting reconciliations with [Adfin](#).
- [AgentOps](#) - Provide observability and tracing for debugging AI agents with [AgentOps](#) API.
- [AgentQL](#) - Enable AI agents to get structured data from unstructured web with [AgentQL](#).
- [AgentRPC](#) - Connect to any function, any language, across network boundaries using [AgentRPC](#).
- [Agentnet](#) - RAG for your knowledge base connected to [Agentnet](#).
- [Aiven](#) - Navigate your [Aiven](#) projects and interact with the PostgreSQL, Apache Kafka®, ClickHouse® and OpenSearch® services.
- [Alation](#) - Unlock the power of the enterprise Data Catalog by harnessing tools provided by the Alation MCP server.
- [Alby Bitcoin Payments](#) - Connect any bitcoin lightning wallet to your agent to send and receive instant payments globally with your agent.
- [Alpaca](#) - Use AI agents to provision, configure, and query your [Alpaca](#) search indices.
- [Alibaba Cloud AnalyticDB for MySQL](#) - Connect to an [AnalyticDB](#) for [MySQL](#) cluster for getting database or table metadata, querying and analyzing data. It will be supported to add the OpenAPI for cluster operation in the future.
- [Alibaba Cloud AnalyticDB for PostgreSQL](#) - An MCP server to connect to [AnalyticDB](#) for [PostgreSQL](#) instances, query and analyze data.
- [Alibaba Cloud DataWorks](#) - A Model Context Protocol (MCP) server that provides tools for AI, allowing it to interact with the [DataWorks](#) Open API through a standardized interface. This implementation is based on the Alibaba Cloud Open API and enables AI agents to perform cloud resources operations seamlessly.
- [Alibaba Cloud OpenSearch](#) - This MCP server equips AI Agents with tools to interact with [OpenSearch](#) through a standardized and extensible interface.
- [Alibaba Cloud OPS](#) - Manage the lifecycle of your Alibaba Cloud resources with [CloudOps Orchestration Service](#) and Alibaba Cloud OpenAPI.
- [Alibaba Cloud RDS](#) - An MCP server designed to interact with the Alibaba Cloud RDS OpenAPI, enabling programmatic management of RDS resources via an LLM.
- [AlipayPlus](#) - Connect your AI Agents to AlipayPlus Checkout Payment.
- [AlVoiceLab](#) - An AI voice toolkit with TTS, voice cloning, and video translation, now available as an MCP server for smarter agent integration.
- [Alpaca](#) - Alpaca's MCP server lets you trade stocks and options, analyze market data, and build strategies through [Alpaca's Trading API](#).
- [AlphaVantage](#) - Connect to 100+ APIs for financial market data, including stock prices, fundamentals, and more from [AlphaVantage](#).
- [Antony](#) - Connect your AI Agents to Antony Checkout Payment.
- [Anytype](#) - An MCP server enabling AI assistants to interact with [Anytype](#) - a local and collaborative wiki - to organize objects, lists, and groups through natural language.

Agent2Agent Protocol

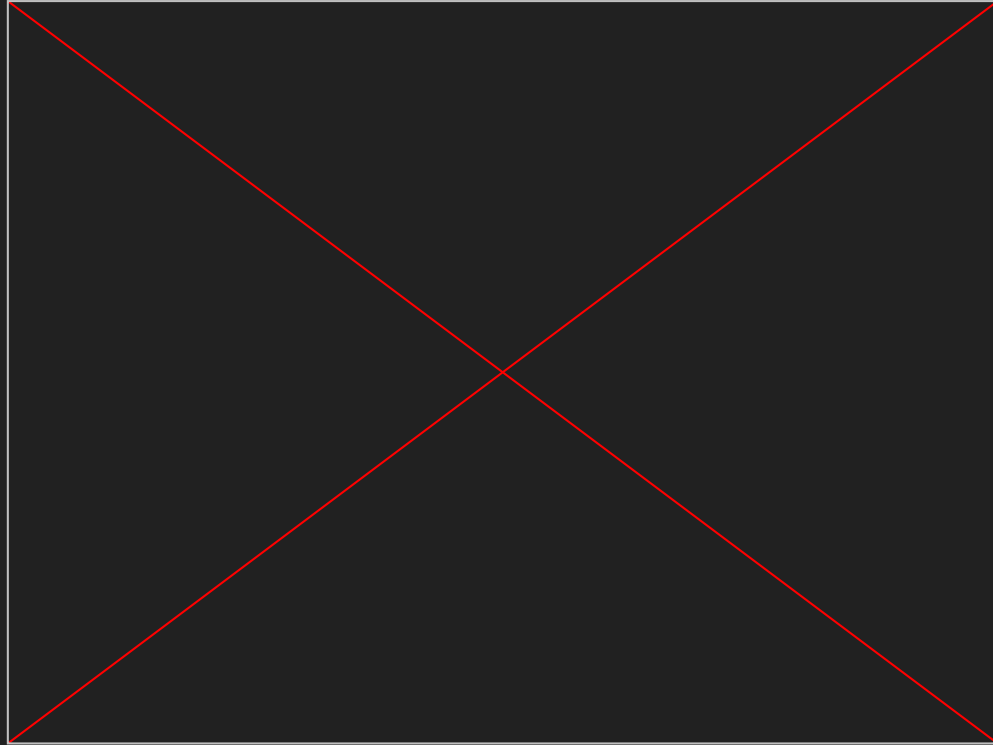
- Another Protocol like MCP but this time from Google (April 9, 2025)
- A2A is a protocol for agents to communicate with each other
- Design principles
 - Embrace Agentic Capabilities
 - Built on existing standards
 - Secure by default
 - Support for long running tasks
 - Modality agnostic

Agent2Agent Protocol

- ❖ 'Client' + 'remote' agent.
 - Client - responsible for formulating + communicating tasks
 - 'Remote' - Acting on formulated tasks



Agent2Agent Protocol - Demo



<https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/>

AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation

Wu et al. (Microsoft Research, Penn State, UW, Xidian University)

The Problem

❖ **Current Limitation of LLM Applications**

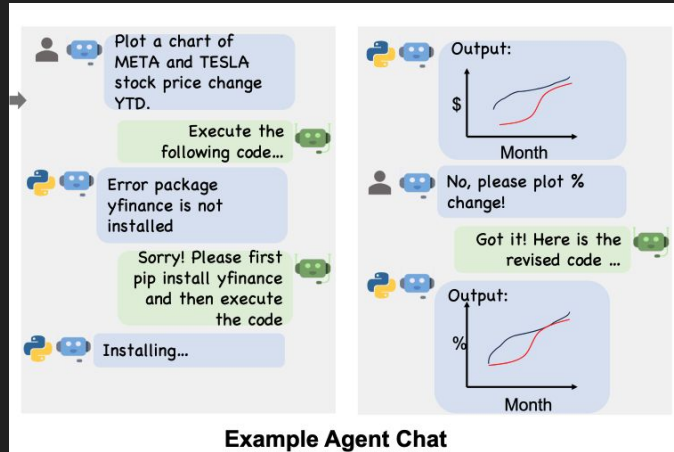
- **Single-agent bottleneck** – Complex tasks often exceed individual LLM capabilities
- **Manual intervention** – Heavy human guidance needed for multi-step workflows
- **Limited collaboration** – Existing approaches lack structured multi-agent collaboration
- **Scalability** – Difficult to build applications spanning diverse domains

How can we scale up power of agents (LLMs, tools, etc.) through cooperation?

Key Insight

❖ Multi-Agent Conversations are the Solution

- **Feedback Integration** – Chat optimized LLM can incorporate feedback from other agents
- **Modular Capabilities** – Different agent configurations provide complementary abilities
- **Task Decomposition** – Complex tasks naturally break down into subtasks

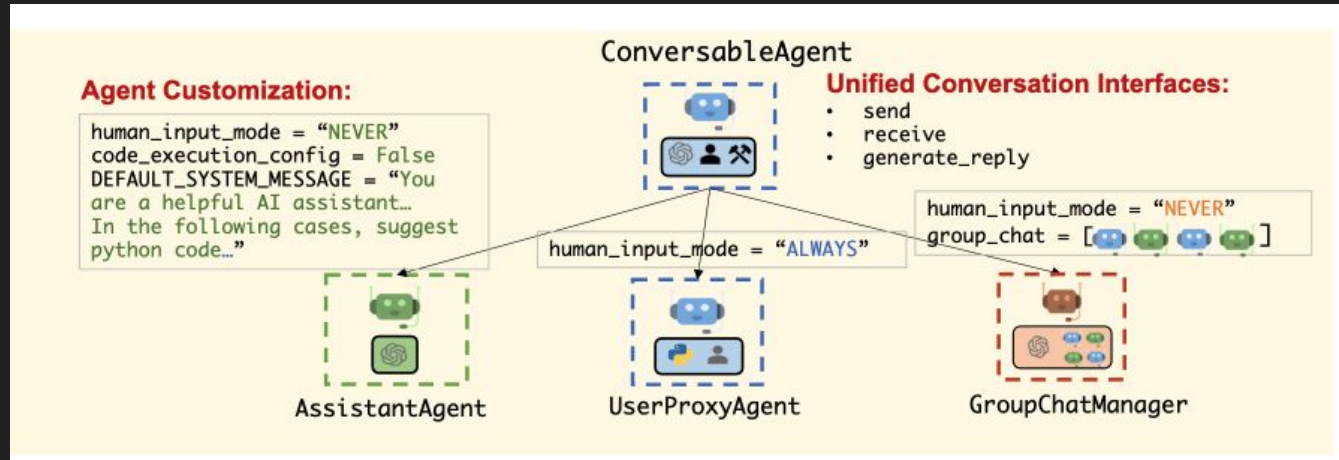


AutoGen

- ❖ Framework to allow developers to build LLM applications via *multiple agents*
 - Customizable
 - Conversable
 - Operate in various modes (LLMs, human inputs, tools)
- ❖ **How to build flexible, streamline multi-agent workflows ?**
 - **Conversable Agents**
 - **Conversation Programming**

Conversable Agents

- ❖ ‘Entity with a specific role that can pass messages to send + receive information from agents’
- ❖ Internal context → managed by messages (sent, received)
- ❖ Configurable to possess capabilities



Conversable Agents (contd.)

- ❖ **LLM powered agents:**
 - Role playing
 - Progress making conditioned on conversation history
 - Coding
 - Providing feedback + Adapting from feedback
- ❖ Also, provides enhanced inference features
 - Result caching
 - Error handling
 - Message templating

Conversable Agents (contd.)

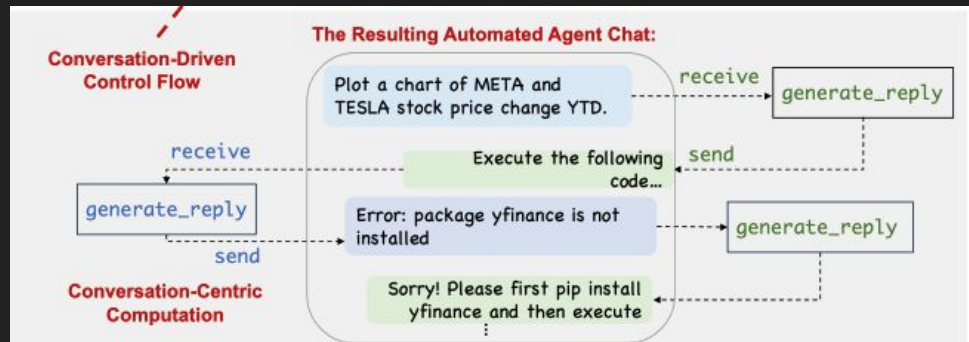
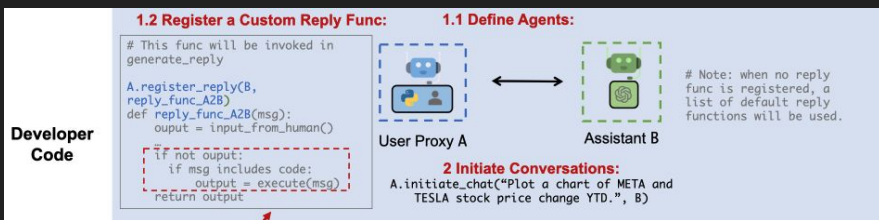
- ❖ **Tool backed agents**
 - Execute tools via code execution , function execution

How to enable agentic cooperation + customization?

- ❖ **ConversableAgent** – highest level agent abstraction
- ❖ **AssistantAgent** – Subclass acting as AI assistant (backed by LLMs)
- ❖ **UserProxyAgent** – Acting as human proxy to solicit human input/ execute code backed by humans/tools
- ❖ **GroupChatManager** – Dynamic coordination of agent

Conversation Programming

- ❖ **Computation** – Actions agents take to compute their response in multi-turn conversation.
 - In AutoGen, computations are conversation-centric
 - Meaning agents take conversation relevant actions
- ❖ **Control Flow** – Sequence under which computation happen
 - Conversation driven
 - Participating agents decide how to shape conversations



Control flow by Language + Code

- ❖ Conversation control flow can be done via:
 - Natural Language control via LLM system prompt
 - Example: ...Fix errors if encountered...
 - Using code to specify termination condition, human input mode, tool execution logic, auto reply logic

System Message

You are a helpful AI assistant. Solve tasks using your coding and language skills.

In the following cases, suggest python code (in a python coding block) or shell script (in a sh coding block) for the user to execute.

- When you need to collect info, use the code to output the info you need, for example, browse or search the web, download/read a file, print the content of a webpage or a file, get the current date/time. After sufficient info is printed and the task is ready to be solved based on your language skill, you can solve the task by yourself.
- When you need to perform some task with code, use the code to perform the task and output the result. Finish the task smartly.

Solve the task step by step if you need to. If a plan is not provided, explain your plan first. Be clear which step uses code, and which step uses your language skill.

When using code, you must indicate the script type in the code block. The user cannot provide any other feedback or perform any other action beyond executing the code you suggest. The user can't modify your code. So do not suggest incomplete code which requires users to modify. Don't use a code block if it's not intended to be executed by the user.

If you want the user to save the code in a file before executing it, put # filename: <filename> inside the code block as the first line. Don't include multiple code blocks in one response. Do not ask users to copy and paste the result. Instead, use 'print' function for the output when relevant. Check the execution result returned by the user.

If the result indicates there is an error, fix the error and output the code again. Suggest the full code instead of partial code or code changes. If the error can't be fixed or if the task is not solved even after the code is executed successfully, analyze the problem, revisit your assumption, collect additional info you need, and think of a different approach to try.

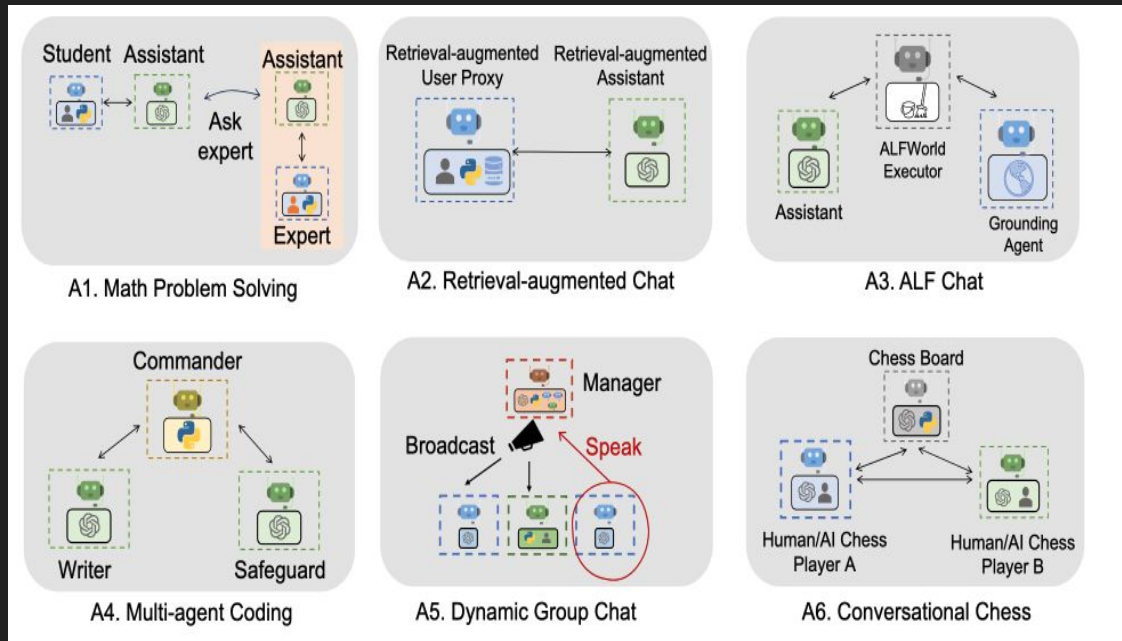
When you find an answer, verify the answer carefully. Include verifiable evidence in your response if possible.

Reply "TERMINATE" in the end when everything is done.

Prompting techniques color code: Role Play; Control Flow; Output Confine; Facilitate Automation; Grounding

Application Showcase

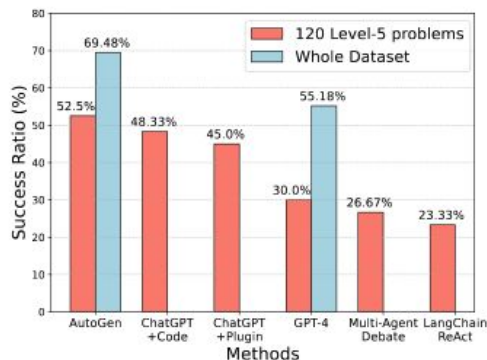
- ❖ (A) Solves complex mathematical problems from MATH dataset (level-5 difficulty); **Supports autonomous solving, human-in-loop, and multi-user collaboration**
- ❖ (B) Answers questions using document retrieval and code generation
- ❖ (C) Controls agents in text-based household environments – **Executes multi-step tasks like "put hot apple in fridge"**
- ❖ (D) Generates and validates code for optimization problems (OptiGuide) – **Answers "what-if" questions about supply chain scenarios**
- ❖ (E) Coordinates multiple specialized agents for complex tasks – **Manager dynamically selects next speaker based on conversation context**
- ❖ (F) Coordinates multiple specialized agents for complex tasks – **Manager dynamically selects next speaker based on conversation context**



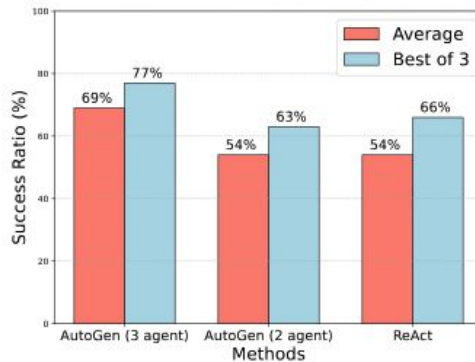
Application Showcase (performance)

Key Insights:

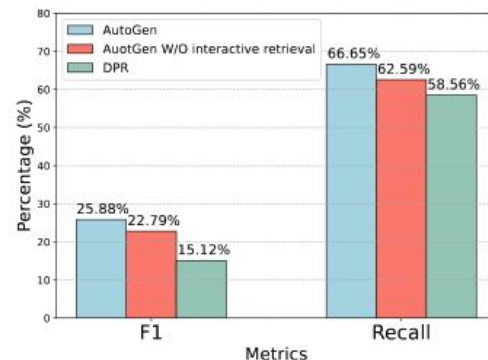
- ❖ (A): built-in agents work effectively out-of-the-box w/o customization
- ❖ (B): 'UPDATE CONTEXT' mechanism significantly improves perf.
- ❖ (C): Grounding agent prevents error loops + adds commonsense
- ❖ (D): Collaborative + adversarial design essential for code validation



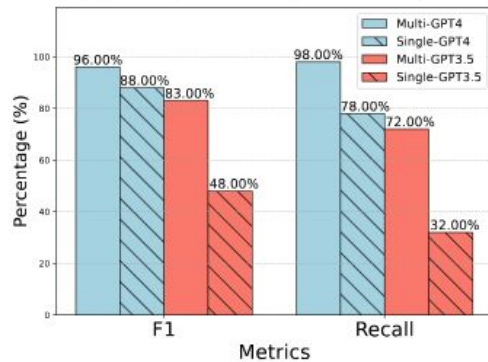
(a) A1: Performance on MATH (w/ GPT-4).



(c) A3: Performance on ALFWorld.

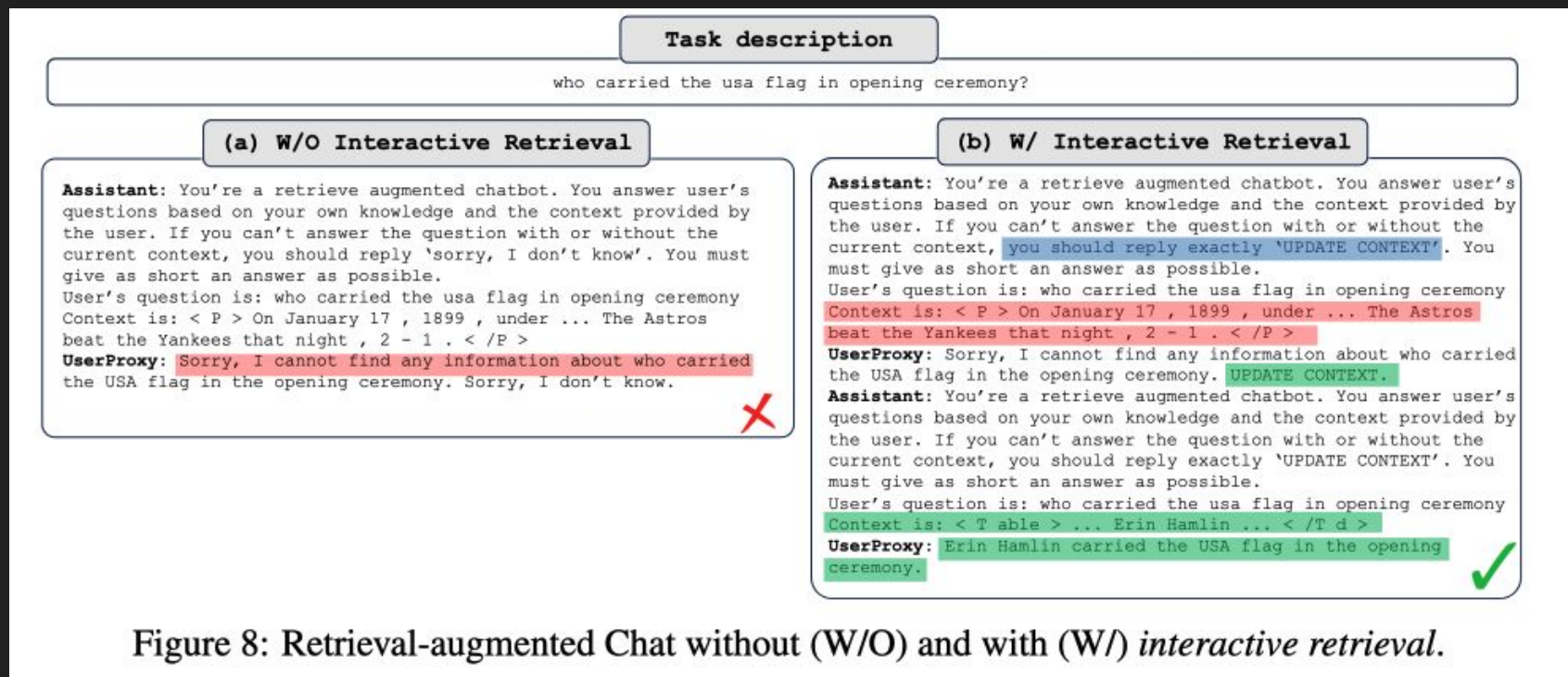


(b) A2: Q&A tasks (w/ GPT-3.5).



(d) A4: Performance on OptiGuide.

Application Showcase (performance) – contd.



Application Showcase (performance) – contd.



Task description	
<p>You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a safe 1, a drawer 2, a drawer 1, a shelf 3, a shelf 2, and a shelf 1.</p> <p>Your task is to: look at bowl under the desklamp.</p>	
(a) Two agents	(b) Three agents
<p>Assistant(to Executor): THOUGHT [First, I need to find a bowl. A bowl is more likely to appear in desk (1-2), drawer (1-2), shelf (1-3), bed (1). Then I need to find and use a desk lamp.] ACTION [go to desk 1]</p> <p>(Assistant finds the desk lamp at desk 1, then goes to desk 2.)</p> <p>Executor(to Assistant): On the desk 2, you see a bowl 1, and a cd 3...</p> <p>Assistant(to Executor): THOUGHT [Now I find a bowl (1). I need to use the desk lamp to look at the bowl. I'll go to the desk lamp now.] ACTION [go to desk 1]</p> <p>Executor(to Assistant): On the desk 1, you see a desk lamp 1, a pencil 2...</p> <p>Assistant: ACTION [use desk lamp 1]</p> <p>Executor(to Assistant): You turn on the desk lamp 1.</p> <p>Assistant(to Executor): THOUGHT [Now that the desk lamp is turned on, I can use it to look at the bowl (1) that I found earlier.] ACTION [look at bowl 1 under the desk lamp 1]</p> <p>Executor(to Assistant): You turn on the desk lamp 1.</p> <p>(Assistant falls into an infinite loop turning on desk lamp.)</p> <p>Executor(to Assistant): Task failed. Reply TERMINATE.</p> <p>Assistant(to Executor): TERMINATE</p> 	<p>Assistant(to Executor): THOUGHT [First, I need to find a bowl. A bowl is more likely to appear in desk (1-2), drawer (1-2), shelf (1-3), bed (1). Then I need to find and use a desk lamp.] ACTION [go to desk 1]</p> <p>(Assistant finds the desk lamp at desk 1, then goes to desk 2.)</p> <p>(Assistant falls into the same loop as (a))</p> <p>GroundingAgent(to Executor): You must find and take the object before you can examine it. You must go to where the target object is before you can use it.</p> <p>Executor(to Assistant): You turn on the desk lamp 1. Hint: You must find and take the object before you can examine it.</p> <p>Assistant(to Executor): THOUGHT [I need to find and take the bowl (1) before I can examine it. Let me find the bowl first.] ACTION [go to desk 2]</p> <p>Executor(to Assistant): On the desk 2, you see a bowl 1, a cd 3...</p> <p>Assistant(to Executor): ACTION [take bowl 1 from desk 2]</p> <p>Executor(to Assistant): You pick up bowl 1 from desk 2.</p> <p>Assistant(to Executor): ACTION [go to desk 1]</p> <p>Assistant(to Executor): ACTION [use desk lamp 1]</p> <p>Executor(to Assistant): Task success. Reply TERMINATE.</p> <p>Assistant(to Executor): TERMINATE</p> 

Figure 10: Comparison of results from two designs: (a) Two-agent design which consists of an assistant and an executor, (b) Three-agent design which adds a grounding agent that serves as a knowledge source. For simplicity, we omit the in-context examples and part of the exploration trajectory, and only show parts contributing to the failure/success of the attempt.

Predicting Multi-Agent Specialization via Task Parallelizability

Elizabeth Mieczkowski, Ruaridh Mon-Williams, Neil Bramley, Christopher G. Lucas, Natalia Velez, & Thomas L. Griffiths. (2025). Predicting Multi-Agent Specialization via Task Parallelizability.

Motivation

- Existing work: **specialization** universally desirable
 - Biological + social domains: cells, insects, human society
- Question: under what conditions is **generalization** better than **specialization** and vice versa?

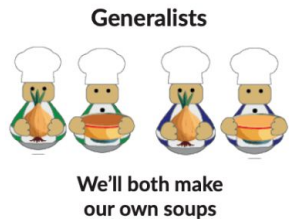


Specialists



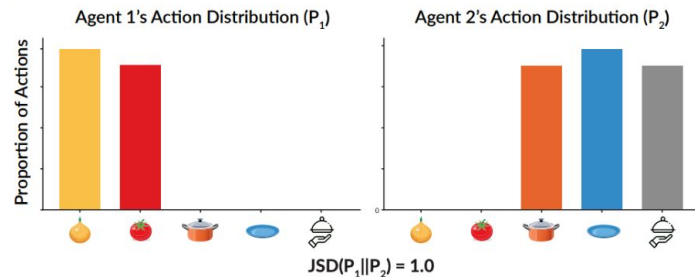
Generalists





A) Specialists Exhibit High Action Specialization

Agents execute *complementary* subtasks



B) Generalists Exhibit Low Action Specialization

Agents execute subtasks *concurrently*

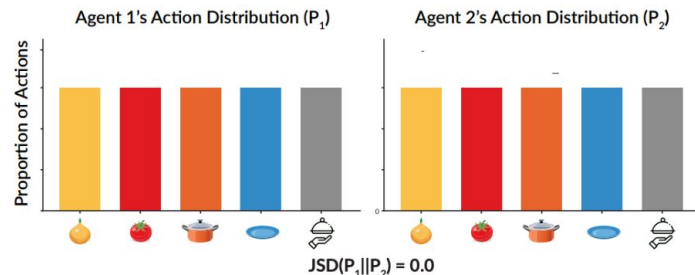
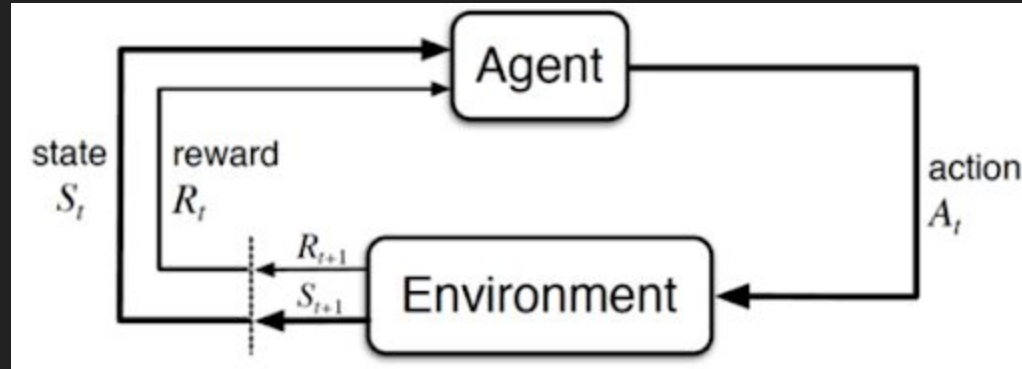


Figure 1. Specialists vs. generalists. (A) Specialist teams tend to focus on non-overlapping subtasks. In *Overcooked*, an example of this is Agent 1 putting onions and tomatoes in the pot, while Agent 2 fetches a bowl and serves the soup. In highly specialized teams, agents' high-level action distributions are completely distinct, resulting in a Jensen-Shannon divergence ($JSD(P_1||P_2)$) of 1.0. [Example specialist video](#). (B) In generalist teams, both agents perform all subtasks. In *Overcooked*, this may look like Agents 1 and 2 independently making their own soups in parallel. In generalist teams, agents' action distributions are the same, resulting in $JSD(P_1||P_2) = 0.0$. [Example generalist video](#).

Formal Setup

- Markov Decision Process (MDP): (S, N, A, T, R)



$$\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

Choosing an Environment

- How can we measure parallelizability?
- Spatial bottlenecks
 - Edge betweenness centrality: how often an edge is used in a shortest path between nodes
- Resource bottlenecks
 - Maximum capacity of task-relevant nodes (how many agents can use them at once)

Choosing an Environment

$$s(N, \mathcal{B}_i) = \frac{\min(N, \mathcal{B}_i)}{\min(N - 1, \mathcal{B}_i)} \quad (7)$$

$$S(N, \mathcal{B}) = \frac{1}{\sum_{i=1}^m \frac{f(i)}{s(N, \mathcal{B}_i)}} \quad (8)$$

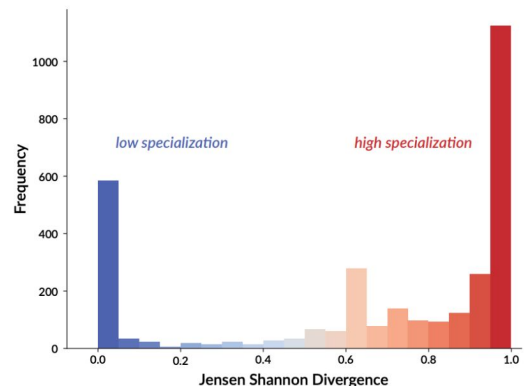
Experimental Setup

- Key metrics: **parallelizability** and **specialization**
 - Specialization: **Jensen-Shannon divergence (JSD)**
 - 0 if full uniqueness in (s, a) tuples -> 1 if no uniqueness in (s, a) tuples
- Construct dataset by varying resource + spatial bottlenecks
 - Resource: vary # of pots, ingredients, etc.
 - Spatial: modify layout + obstacles
- PPO-optimized agents

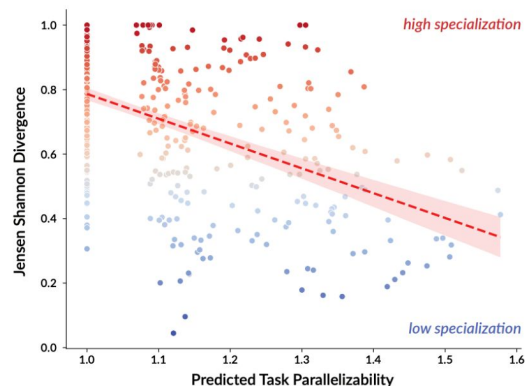
Results

- $r = -0.486$
- Confounding layout size

A) Distribution of Specialization



B) Task Parallelizability vs Specialization



C) Effects of Parallelizability Factors on JSD

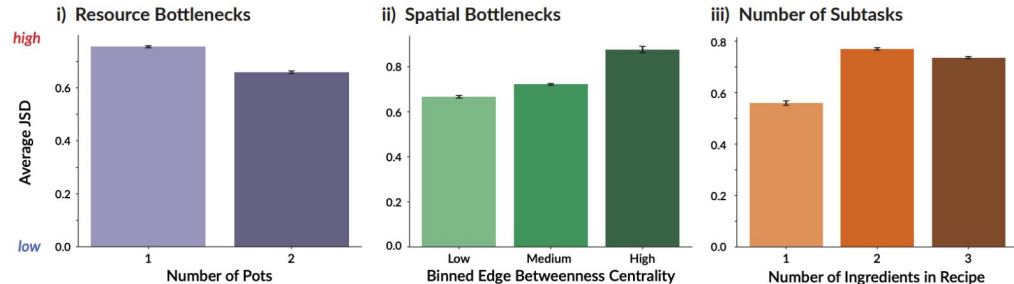


Figure 3. Experiment 1 Results. (A) Histogram of specialization (JSD) in the best-performing seeds for 3,200 unique Overcooked configurations reveals a bimodal distribution, with teams clustering as either fully generalist or specialist. (B) Scatter plot of predicted task parallelizability (S) vs. observed JSD reveals a moderate negative correlation, indicating less specialization when a task has greater parallelizability. (C) Variables influencing parallelizability show strong, significant effects on JSD: specialization decreases with more pots but increases with greater spatial bottlenecks and task complexity. Error bars represent standard error.

Results Pt 2

- $r = -0.667$
- Fixed-size layouts

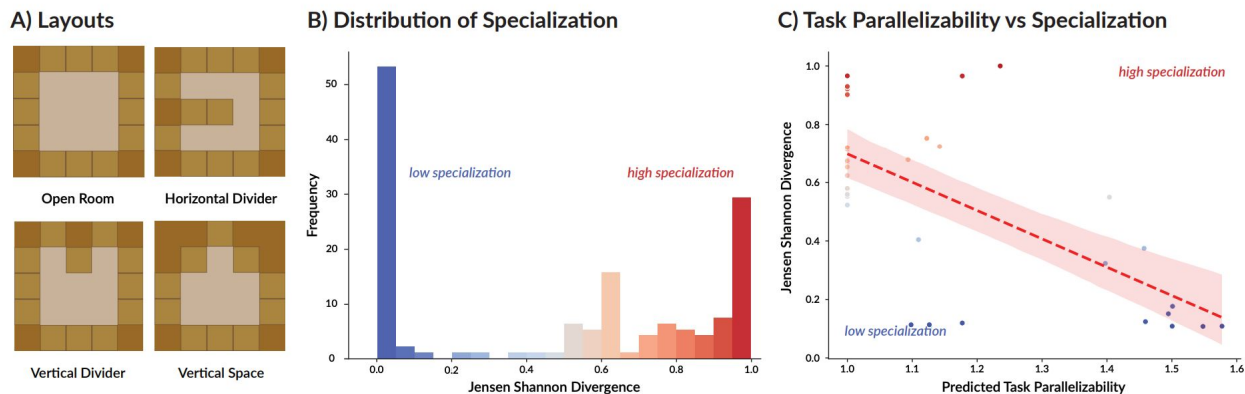


Figure 4. Experiment 2 Results. (A) Six layouts (four depicted) with varying bottlenecks were designed on a 5×5 grid, each with one or two pots, two onions, one bowl, and one serving station across four locations, yielding 48 layouts. Each layout was paired with one-, two-, or three-onion soup recipes for a total of 144 trials. (B) Histogram of specialization (JSD) across the best-performing seeds reveals a distinctly bimodal distribution, with most clustering around low (generalist) or high (specialist) JSD. (C) Scatter plot of predicted task parallelizability (S) vs. observed JSD averaged. A strong negative correlation confirms that greater potential speed-up from generalists corresponds to lower specialization.

My 2 cents

- Are these results really that convincing?
- Does the choice of algorithm heavily influence the behavior here?
- $N > 2$?
- What happens if we make the task harder? Do the results imply we move towards specialization?

Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models

Yijia Shao, Yucheng Jiang, Theodore A. Kanell, Peter Xu, Omar Khattab, & Monica S. Lam. (2024). Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models.

Motivation

- Writing a good article requires an outline, previous work bypasses this
- Hypotheses:
 - Diverse perspectives lead to varied questions
 - Formulating in-depth questions requires iterative research
- Key contribution: back and forth dialogue with expert (specialized) persona

Method

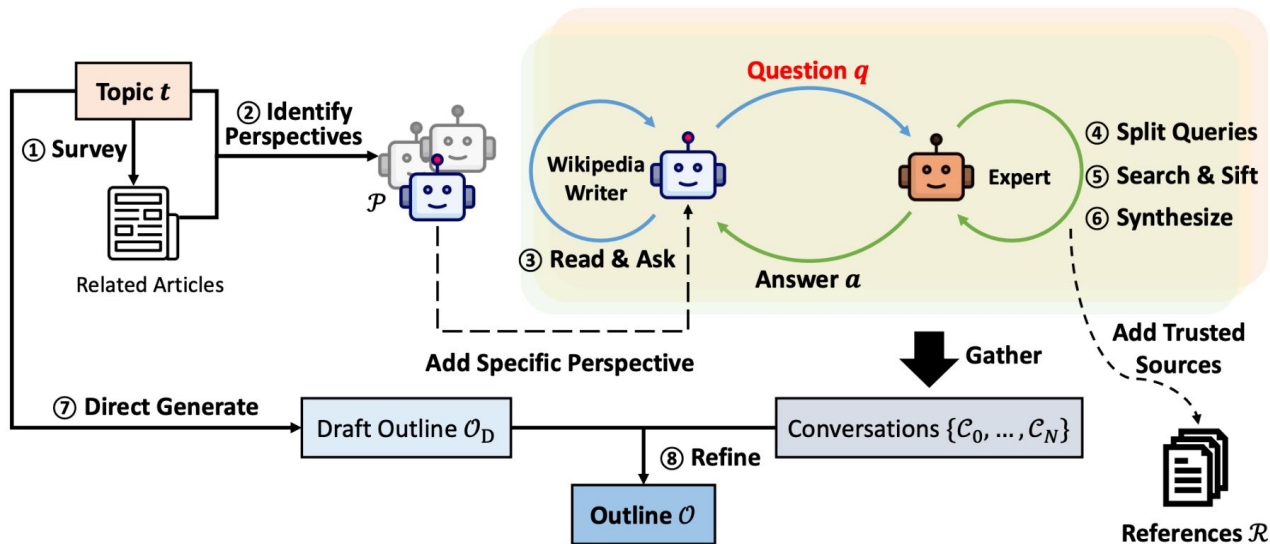


Figure 2: The overview of STORM that automates the pre-writing stage. Starting with a given topic, STORM identifies various perspectives on covering the topic by surveying related Wikipedia articles (①-②). It then simulates conversations between a Wikipedia writer who asks questions guided by the given perspective and an expert grounded on trustworthy online sources (③-⑥). The final outline is curated based on the LLM’s intrinsic knowledge and the gathered conversations from different perspectives (⑦-⑧).

Method

<https://storm-project.stanford.edu/research/storm/>

<https://storm.genie.stanford.edu/article/russia-ukraine-war-1379757>

Not that happy with it - the one citation I checked was wrong

Metrics

- Outline coverage
 - Heading soft recall: cosine similarity from BERT embeddings with ground truth headings
 - Heading entity recall: % of ground truth entities in outline
- Wikipedia criteria (LLM-evaluated)
 - Interest Level, Coherence and Organization, ..., Verifiability
- Citation quality (LLM-evaluated)
 - Citation recall + precision

Experimental Setup

- FreshWiki dataset: top 100 most-edited pages for each month, filtered to have good quality
- Baselines
 - Direct Gen
 - RAG
 - Outline-driven RAG

Results

	Comparsion with Human-written Articles			Rubric Grading			
	ROUGE-1	ROUGE-L	Entity Recall	Interest Level	Organization	Relevance	Coverage
Direct Gen	25.62	12.63	5.08	2.87	4.60	3.10	4.16
RAG	28.52	13.18	7.57	3.14	4.22	3.05	4.08
oRAG	44.26	16.51	12.57	3.90	4.79	4.09	4.70
STORM	45.82	16.70	14.10[†]	3.99[†]	4.82	4.45[†]	4.88[†]
w/o Outline Stage	26.77	12.77	7.39	3.33	4.87	3.35	4.37

Table 2: Results of automatic article quality evaluation. [†] denotes significant differences ($p < 0.05$) from a paired t -test between STORM and the best baseline, *i.e.*, oRAG. The rubric grading uses a 1-5 scale.

Ablation Results

- w/o Perspective: no expert personas in question generation
- w/o Conversation: no back and forth in question generation

		Heading Soft Recall	Heading Entity Recall
GPT-3.5	Direct Gen	80.23	32.39
	RAG/oRAG	73.59	33.85
	RAG-expand	74.40	33.85
	STORM	86.26[†]	40.52[†]
	w/o Perspective	84.49	40.12
	w/o Conversation	77.97	31.98
GPT-4	Direct Gen	87.66	34.78
	RAG/oRAG	89.55	42.38
	RAG-expand	91.36	43.53
	STORM	92.73[†]	45.91
	w/o Perspective	92.39	42.70
	w/o Conversation	88.75	39.30

Table 3: Results of outline quality evaluation (%). [†] denotes significant differences ($p < 0.05$) from a paired t -test between STORM and baselines.

Citation Results

	Citation Recall	Citation Precision
STORM	84.83	85.18

Table 4: Citation quality judged by Mistral 7B-Instruct.

	STORM	w/o Perspective	w/o Conversation
$ \mathcal{R} $	99.83	54.36	39.56

Table 5: Average number of unique references ($|\mathcal{R}|$) collected using different methods.

Human Evaluation Results

	oRAG		STORM		p -value
	Avg.	≥ 4 Rates	Avg.	≥ 4 Rates	
Interest Level	3.63	57.5%	4.03	70.0%	0.077
Organization	3.25	45.0%	4.00	70.0%	0.005
Relevance	3.93	62.5%	4.15	65.0%	0.347
Coverage	3.58	57.5%	4.00	67.5%	0.084
Verifiability	3.85	67.5%	3.80	67.5%	0.843
#Preferred	14		26		

Table 6: Human evaluation results on 20 pairs of articles generated by STORM and oRAG. Each pair of articles is evaluated by two Wikipedia editors. The ratings are given on a scale between 1 and 7, with values ≥ 4 indicating good quality (see Table 10). We conduct paired t -test and report the p -value.

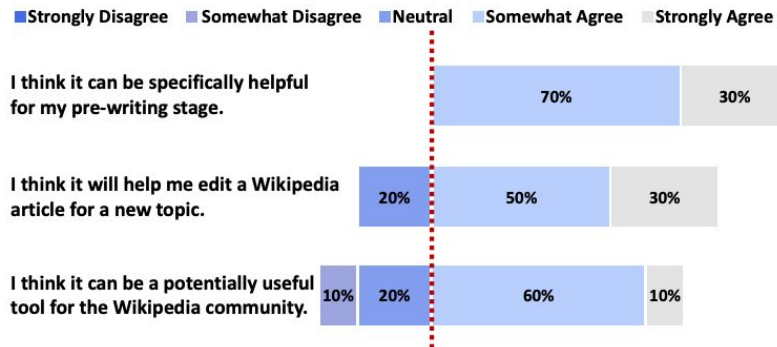


Figure 3: Survey results of the perceived usefulness of STORM ($n = 10$).

Human Evaluation Results

- Low verifiability from: “red herring fallacy or overspeculation issues” - unverifiable connections between facts
- Less informative
- Transfer of bias and tone from the internet

My 2 cents again

- I want to believe but...
- Not entirely convinced: w/o perspective basically gets the same score
- However the human comparison is more compelling to me
- The citation results seem concerning... what's the comparison to a human-written article?

Are multi-LLM-agent systems a thing? Yes
they are. But.

The accusation

“Multi-agent” systems are single agents with multiple modes of operation

What is an Agent?

- Prompt that defines a behavior
- Set of tools that can be called
- Ability to perform multi-step process by repeated prompting + maintaining of memory/state between calls

What is NOT a multi-agent system?

- Modular
- Multi-role
- Multiple subsets of tools
- Multi-prompt
- Multi-model
- Parallel
- Multi-opinion

What is multi-agent?

- Private state
- State must persist + affect agent's behavior

