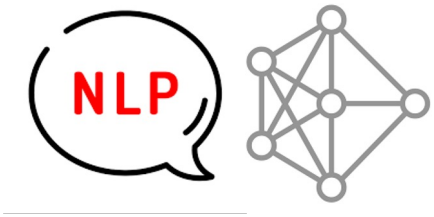
CHULA ENGINEERING
Foundation toward Innovation

COMPUTER



RAG & Agentic LLMs

2110572: Natural Language Processing Systems

Assoc. Prof. Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University

Credit: TA.Pluem & TA.Knight

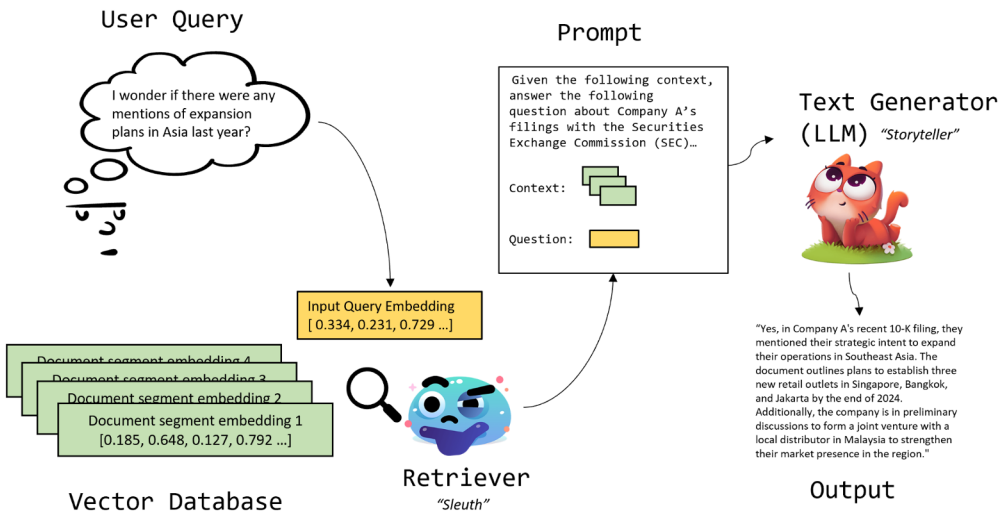
Outline

- 1) Introduction
- Retrieval-augmented Generation
  - 2) Typical Workflow
    - Hybrid Search
    - Document Reranking
  - 3) Advanced RAG
  - 4) Evaluation
- 5) Agentic LLMs
  - Motivation
  - Reasoning
  - Memory
  - 6) Real-World Agentic Applications
  - 7) Evaluation
- 8) LLM Tools

Introduction

Retrieval-Augmented Generation (RAG)

- Combining a retriever with an LLM to generate highly informed and contextually relevant outputs.
  - Given a query, the retriever finds the most relevant documents.
  - The LLM takes the documents to answer the query.



Text Retrieval System

A system that finds the most relevant text (such as an answer, paragraph, or passage) given a query (which could be a question, keywords, or any relevant text).

(1) Sparse embedding

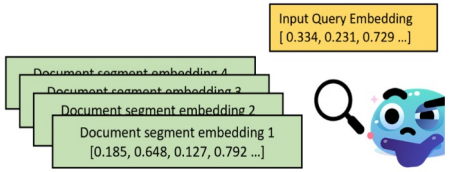
{'mil': 0.27, '##vus': 0.39, 'is': 0.16, 'a': 0.11, 'vector': 0.41, 'database': 0.24, 'built': 0.17, 'for': 0.09, 'scala': 0.19, '##ble': 0.11, 'similarity': 0.30, 'search': 0.19}

idf BM25 tf

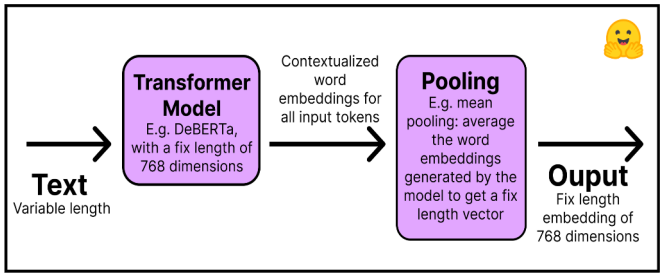
$$BM25 = \sum_{t \in q} \log\left[\frac{N}{df(t)}\right] \cdot \frac{(k_1+1)\cdot tf(t,d)}{k_1 \cdot \left[(1-b)+b\cdot\frac{dl(d)}{dl_{avg}}\right] + tf(t,d)}$$

- k_1, b – parameters
- $dl(d)$ – length of document d
- dl_{avg} – average document length

(2) Dense embedding



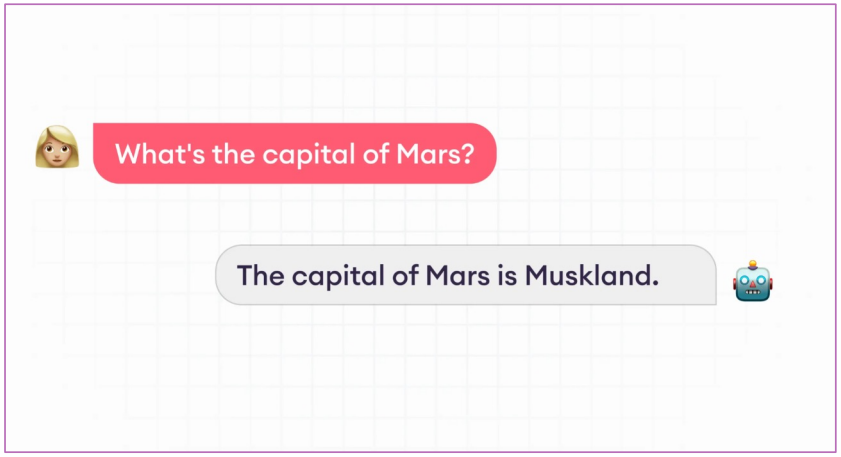
Sentence embedding



Why use RAG?

- Let's you use your own data. This includes:
  - private data
  - new data
  - domain-specific data

- Reduces hallucinations
  - by grounding the response in factual data



RAG vs Fine-tuning

- Use RAG if data is dynamic.
- Use fine-tuning if you want to change the behavior of the model.
- You can also use both!

Common use cases

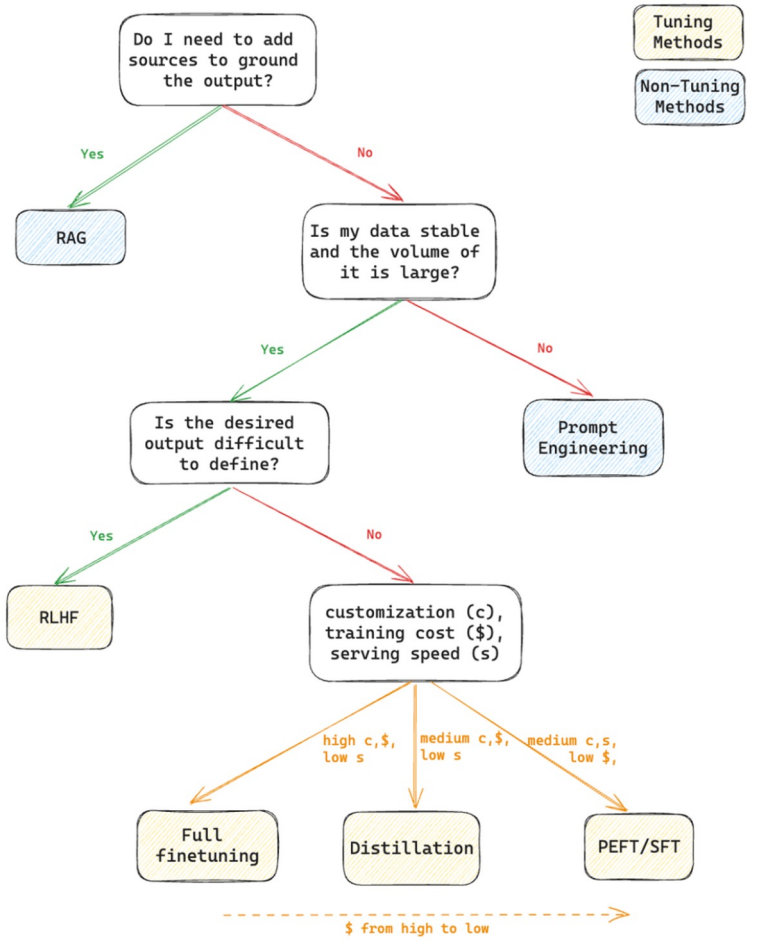
Some common use cases where fine-tuning can improve results:

- Setting the style, tone, format, or other qualitative aspects
- Improving reliability at producing a desired output
- Correcting failures to follow complex prompts
- Handling many edge cases in specific ways
- Performing a new skill or task that's hard to articulate in a prompt

One high-level way to think about these cases is when it's easier to "show, not tell". In the sections to come, we will explore how to set up data for fine-tuning and various examples where fine-tuning improves the performance over the baseline model.

Another scenario where fine-tuning is effective is reducing cost and/or latency by replacing a more expensive model like `gpt-4o` with a fine-tuned `gpt-4o-mini` model. If you can achieve good results with `gpt-4o`, you can often reach similar quality with a fine-tuned `gpt-4o-mini` model by fine-tuning on the `gpt-4o` completions, possibly with a shortened instruction prompt.

https://platform.openai.com/docs/guides/fine-tuning



https://cloud.google.com/blog/products/ai-machine-learning/to-tune-or-not-to-tune-a-guide-to-leveraging-your-data-with-llms

Retrieval-Augmented Generation

Typical RAG Workflow (1)

Step 1: Split your documents into manageable-sized chunks.

- You will need to find a good chunk size based on your documents.
- If structure matters in your document, try using a markdown splitter.

Markdown splitter example

For example, if we want to split this markdown:

```
md = '# Foo\n\n ## Bar\n\nHi this is Jim  \nHi this is Joe\n\n ## Baz\n\n Hi this is Molly'
```

We can specify the headers to split on:

```
[("#", "Header 1"),("##", "Header 2")]
```

And content is grouped or split by common headers:

```
{'content': 'Hi this is Jim  \nHi this is Joe', 'metadata': {'Header 1': 'Foo', 'Header 2': 'Bar'}}
{'content': 'Hi this is Molly', 'metadata': {'Header 1': 'Foo', 'Header 2': 'Baz'}}
```

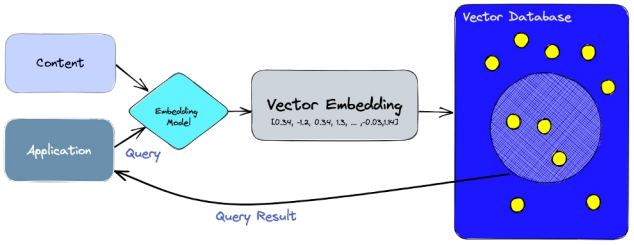
Name	Classes	Splits On	Adds Metadata	Description
Recursive	RecursiveCharacterTextSplitter, RecursiveJsonSplitter	A list of user defined characters		Recursively splits text. This splitting is trying to keep related pieces of text next to each other. This is the `recommended way` to start splitting text.
HTML	HTMLHeaderTextSplitter, HTMLSectionSplitter	HTML specific characters	✅	Splits text based on HTML-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the HTML)
Markdown	MarkdownHeaderTextSplitter	Markdown specific characters	✅	Splits text based on Markdown-specific characters. Notably, this adds in relevant information about where that chunk came from (based on the Markdown)
Code	many languages	Code (Python, JS) specific characters		Splits text based on characters specific to coding languages. 15 different languages are available to choose from.
Token	many classes	Tokens		Splits text on tokens. There exist a few different ways to measure tokens.
Character	CharacterTextSplitter	A user defined character		Splits text based on a user defined character. One of the simpler methods.
[Experimental] Semantic Chunker	SemanticChunker	Sentences		First splits on sentences. Then combines ones next to each other if they are semantically similar enough. Taken from Greg Kamradt
AI21 Semantic Text Splitter	AI21SemanticTextSplitter	✅	Identifies distinct topics that form coherent pieces of text and splits along those.	

https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/

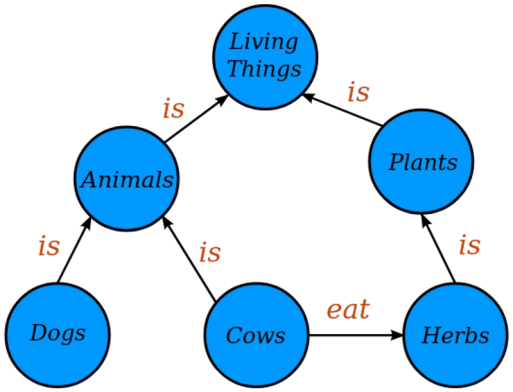
Typical RAG Workflow (2)

Step 2: Index your chunks into a database or knowledge base of some kind.

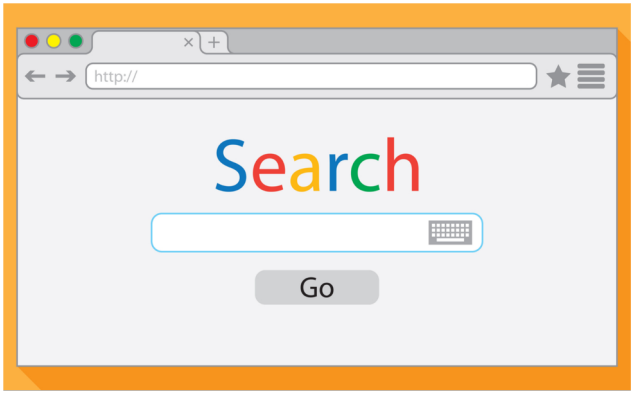
Vector Databases



Knowledge Graphs



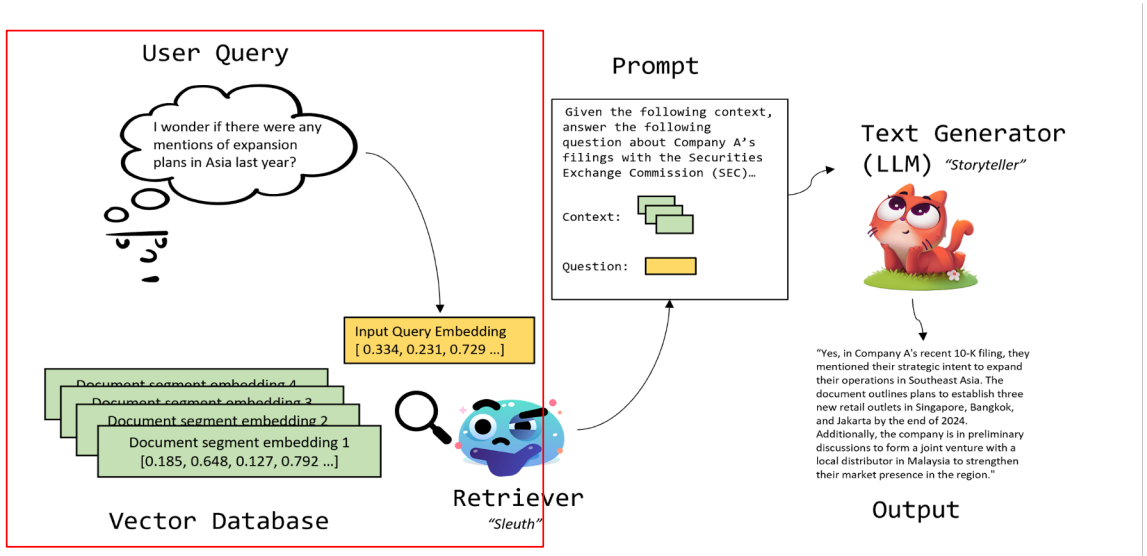
Search Engines



Typical RAG Workflow (3)

Step 3: Given a query, **find the most similar documents** in your knowledge base. E.g., perform vector search in a vector database.

- Use hybrid search for better performance
- (Optiona



There are 3 search paradigms

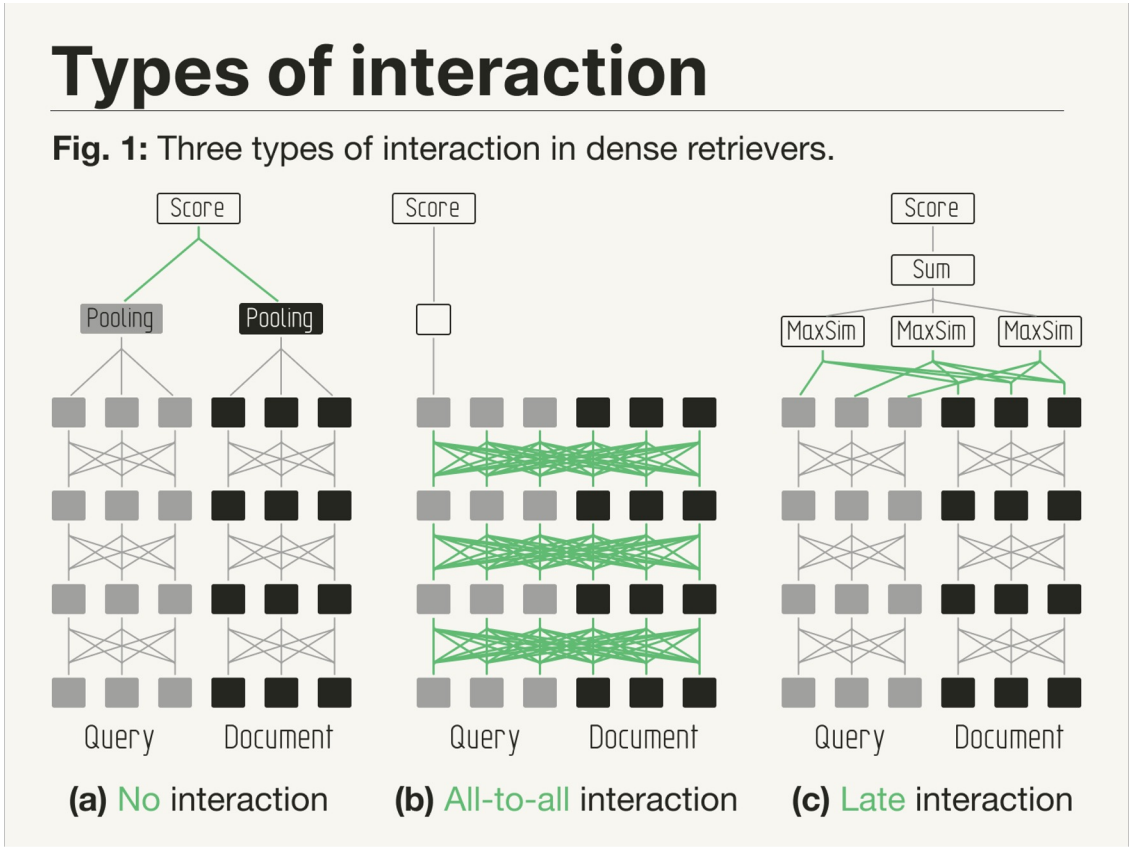
(a) is the most efficient but least performance.

(b) is the most computationally expensive but performs best.

(a) and (c) can be precomputed offline.

To trade-off between speed and accuracy, two-stage ranking is often used:

- (a) (and optionally (c)) is used to find a small group of candidates
- then run (b) for the final score.

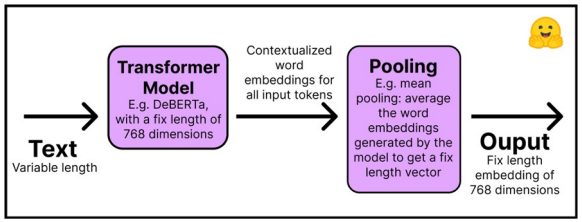


Hybrid Search

- Step3: IR
- **3.1) Use hybrid search for better performance**
- 3.2) (Optional) - perform document reranking

- Basically **contextual search (dense embedding) + keyword search (sparse embedding)**
- Supported by many vector DBs.

Embedding from NN+BM25

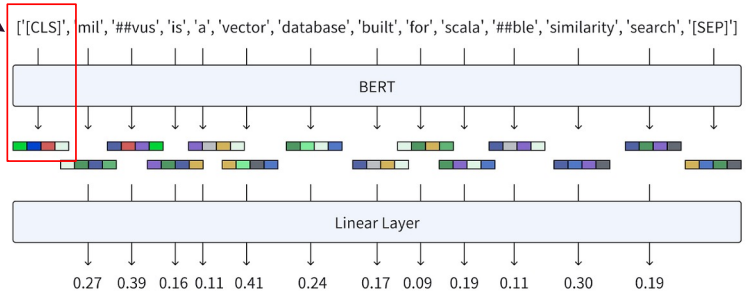


$$BM25 = \sum_{t \in q} \log\left[\frac{N}{df(t)}\right] \cdot \frac{(k_1+1)\cdot tf(t,d)}{k_1 \cdot \left[(1-b)+b\cdot\frac{dl(d)}{dl_{avg}}\right] + tf(t,d)}$$

- k_1, b – parameters
- $dl(d)$ – length of document d
- dl_{avg} – average document length

BGE-M3

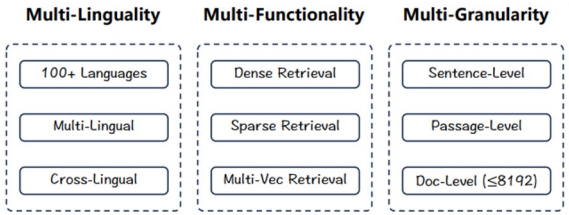
[CLS] token embedding is the dense embedding



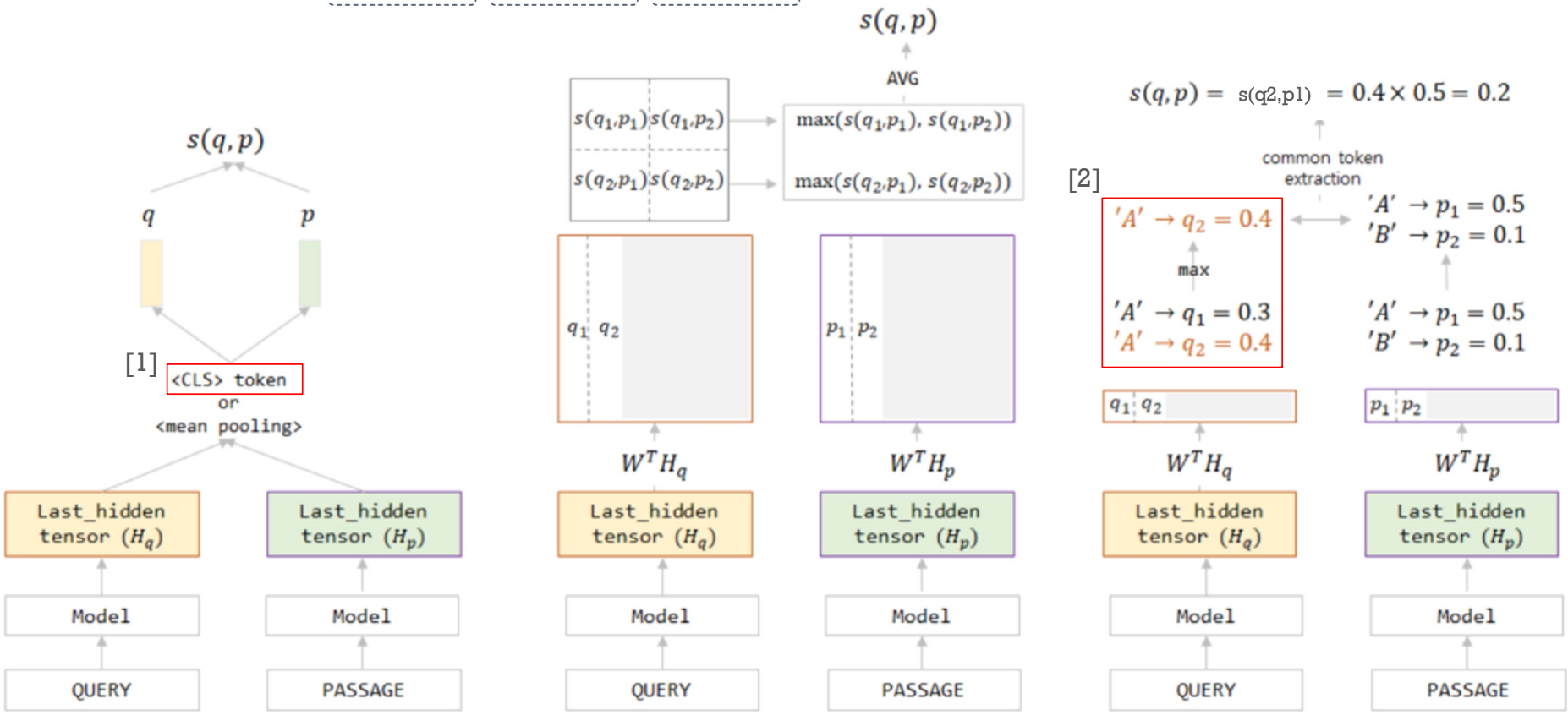
The sparse embedding

{'mil': 0.27, '##vus': 0.39, 'is': 0.16, 'a': 0.11, 'vector': 0.41, 'database': 0.24, 'built': 0.17, 'for': 0.09, 'scala': 0.19, '##ble': 0.11, 'similarity': 0.30, 'search': 0.19}

BGE-M3



[1] Use the [CLS] token as the dense embedding
[2] Take the max score in case a term appears multiple times. This is the sparse embedding. (the score is the importance of each term, like BM25)
[3] Late interaction paradigm



https://www.linkedin.com/pulse/bge-m3-retriever-junhong-kim-stfjc

Hybrid Search: Reranking Score

- Reciprocal Rank Fusion (RRF) and Distribution-based Score Fusion (DBSF) are used to combine the score between multiple search results

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

D - set of docs
R - set of rankings as permutation on 1..|D|
K - typically set to 60 by default

Retrieval Modules	Execution Time(seconds)	Ragas Context Precision@10
Bm25	0.274728	0.649015
VectorDB	0.496673	0.522239
Hybrid RRF (RRF-k: 10)	0.771401	0.676157
Hybrid RRF (RRF-k: 3)	0.771401	0.640295
Hybrid RRF (RRF-k: 5)	0.771401	0.668342
Hybrid CC (Weights: 0.7, 0.3)	0.771401	0.652625
Hybrid DBSF (Weights: 0.7, 0.3)	0.771401	**0.696401**

Table 3: Table of Execution Time and Ragas Context Precision by Retrieval Experiments

DBSF

$$S\left(\bigcup_{i \in E} \frac{x_i - (\mu_i - 3\sigma_i)}{(\mu_i + 3\sigma_i) - (\mu_i - 3\sigma_i)}\right)$$

where:
- E is a set of embeddings from retrievers
- mu and sigma are the mean and std. of each set of embedding

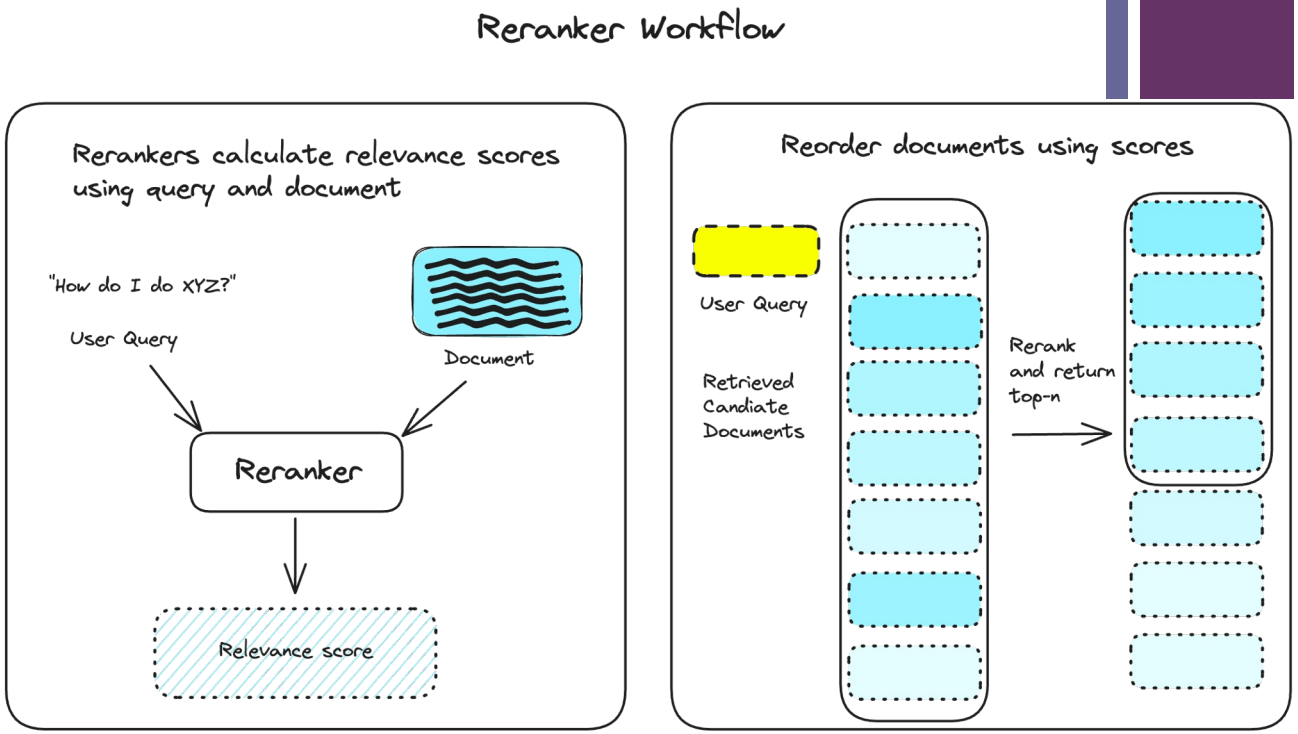
https://medium.com/plain-simple-software/distribution-based-score-fusion-dbsf-a-new-approach-to-vector-search-ranking-f87c37488b18
https://arxiv.org/pdf/2410.20878v1

Document Reranking

- Step3: IR
- 3.1) Use **hybrid search** for better performance
- 3.2) (Optional) - perform **document reranking**

Uses a reranker model or a "cross-encoder" to **rerank the retrieved documents by the relevancy of the query to the documents**.

This process often improves the RAG pipeline performance by weeding out unrelated documents.



Performance on MIRACL dataset

model	inference length	avg	ar	bn	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh	de	yo
bge-m3	512+512	**67.91**	78.4	80	59.6	55.5	57.7	78.6	57.8	59.3	56	72.8	69.9	70.1	78.6	86.2	82.6	61.7	56.8	60.7
bge-reranker-v2-m3	1024	**72.84**	81.7	84.63	63.45	63.71	62.49	82.41	63.26	68.25	62.71	79.96	73.79	76.93	82.27	89.36	85.3	64.2	62.64	64.03
bge-reranker-v2-gemma	1024	**73.39**	82.26	85.15	66.62	64.29	62.03	82.58	64.26	68.68	61.33	79.72	74.83	78.36	81.46	89.22	86.06	65.61	64.25	64.37
bge-reranker-v2-minicpm-20	1024	**62.26**	71.8	62.15	62.51	56.45	45.28	74.73	52.77	50.64	57.3	70.57	69.12	60.58	67.4	76.98	67.87	61.61	49.74	63.2
bge-reranker-v2-minicpm-28	1024	**67.75**	77.4	64.87	66.03	61.89	51.91	80.02	63.42	56	60.5	78.25	73.39	72.5	72.09	78.41	74.46	66.09	59.15	63.07
bge-reranker-v2-minicpm-40	1024	**67.77**	77.49	64.8	66.02	62.05	51.78	80.12	62.98	55.7	60.51	78.07	73.01	72.64	72.81	78.51	74.24	65.82	59.43	63.91

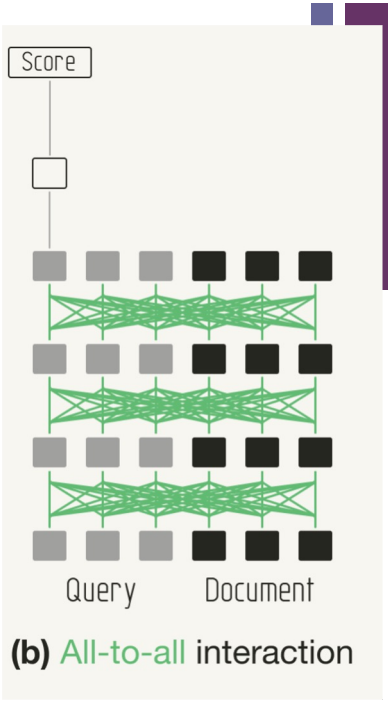
https://huggingface.co/BAAI/bge-m3

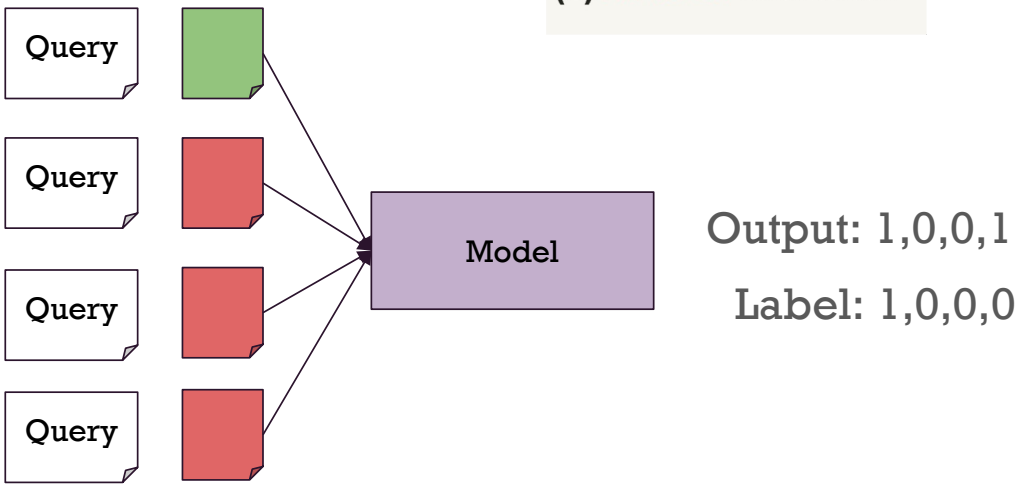
How to train Reranker (conceptual)

A reranker takes in the query and a document and outputs a score of [0,1] where 1 means the doc is most relevant to the query.

To train a reranker, you will need to feed it:

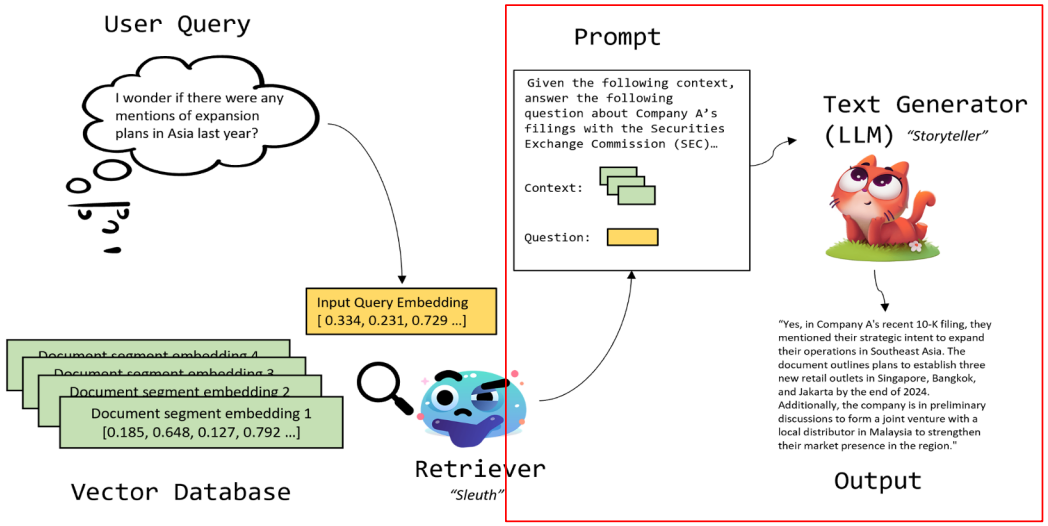
- A query
- Positive documents
- (Hard) Negative documents
- Labels





Typical RAG Workflow (4)

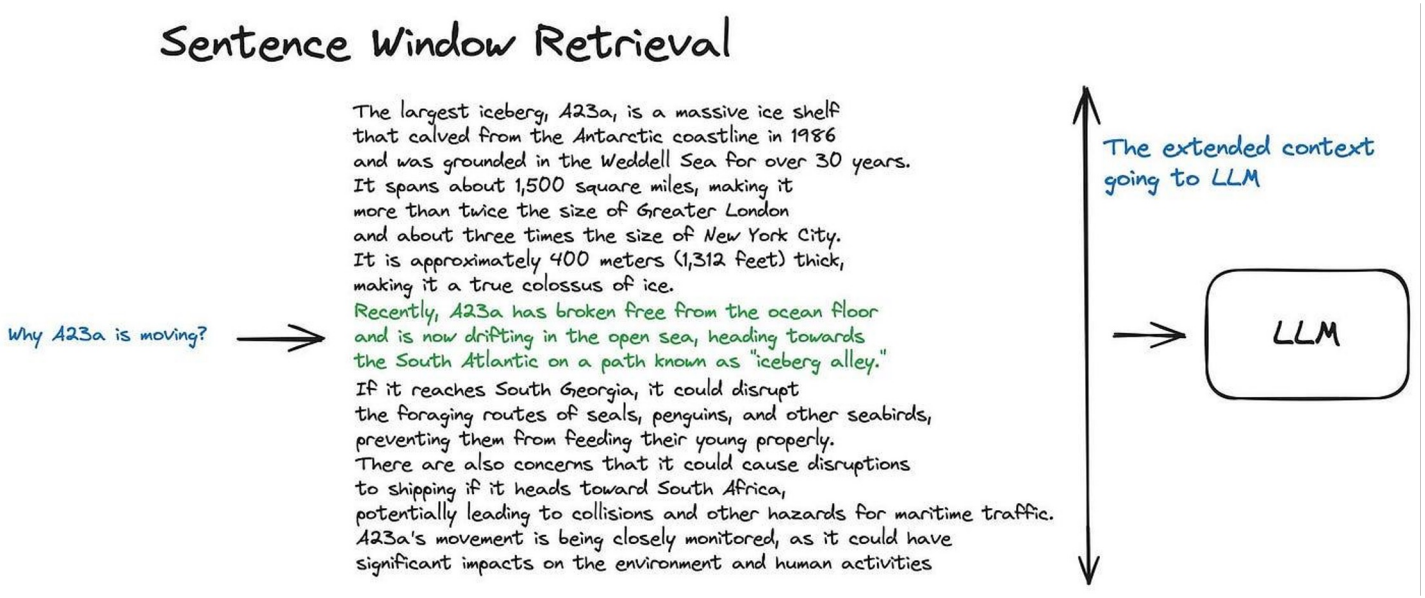
Step 4: Feed the query and documents to the LLM to generate an answer.



Advanced RAG

1) Sentence Window Retriever

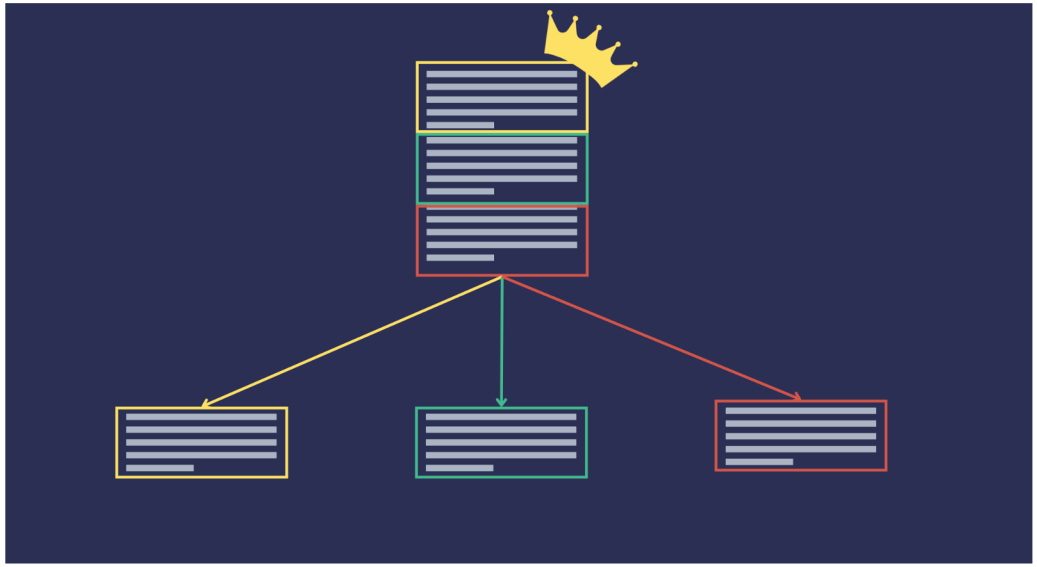
Instead of chunk-based RAG, sentence window retriever instead looks for the sentence most relevant to the query and take **the entire window of sentences** around it as the context for the LLM.



https://haystack.deepset.ai/tutorials/42_sentence_window_retriever

2) Auto-Merging Retriever

- Auto-Merging is a retrieval technique that leverages a hierarchical document structure.
  - Parent nodes are the original document and children nodes are chunks of their parent.
- Return top_k docs normally.
- However, if the no. of children nodes that belong to the same parent returned is more than a threshold, return the parent node instead.



e.g. if threshold = 50% and 2 children are retrieved, return the parent instead

https://docs.llamaindex.ai/en/stable/examples/retrievers/auto_merging_retriever/

RAG Tools & Framework

Frameworks

LlamaIndex

LangChain

microsoft/**graphrag**

A modular graph-based Retrieval-Augmented Generation (RAG) system

59 Contributors · 227 Used by · 204 Discussions · 21k Stars · 2k Forks

Vector DBs

drant

Pinecone

Chroma

facebookresearch/**faiss**

A library for efficient similarity search and clustering of dense vectors.

167 Contributors · 4k Used by · 225 Discussions · 32k Stars · 4k Forks

Load data and build an index

In the same folder where you created the `data` folder, create a file called `starter.py` file with the following:

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader("data").load_data()
index = VectorStoreIndex.from_documents(documents)
```

This builds an index over the documents in the `data` folder (which in this case just consists of the essay text, but could contain many documents).

Your directory structure should look like this:

```
├── starter.py
└── data
    └── paul_graham_essay.txt
```

Query your data

Add the following lines to `starter.py`

```
query_engine = index.as_query_engine()
response = query_engine.query("What did the author do growing up?")
print(response)
```

This creates an engine for Q&A over your index and asks a simple question. You should get back a response similar to the following: `The author wrote short stories and tried to program on an IBM 1401.`

Evaluation

Retrieval

- **Hit-rate**: assesses whether the relevant documents are being retrieved.
- **MRR (Mean reciprocal rank)**: gauges if the relevant documents are ranked high when retrieved.
- **NDCG@K (Normalized Discounted Cumulative Gain):** evaluates the quality of ranked results by considering the relevance and the position of each relevant item in the entire ranked list

Generation

- **Faithfulness**: checks for hallucination (whether the response is grounded to the retrieved contexts.)
- **Relevancy**: evaluates the relevance of both the retrieved context and the generated answer to the initial query.
- **Correctness**: determines if the generated answer is correct based on the reference answer.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$$

$$NDCG@K = \frac{DCG@K}{IDCG@K} = \frac{\sum_{i=1}^{k\ (\text{actual order})}\frac{Gains}{log_2(i+1)}}{\sum_{i=1}^{k\ (\text{ideal order})}\frac{Gains}{log_2(i+1)}}$$

- Actual List: ["Apple", "Banana", "Orange"]
- Retrieved List: ["Banana", "Apple", "Orange"]

$Apple : 2\ ,\ Banana : 1\ ,\ Orange : 0$

Using them, We calculate:

$$DCG@3 = \frac{1}{log_2(1+1)} + \frac{2}{log_2(1+2)} + \frac{0}{log_2(1+3)} = 2.26$$

$$IDCG@3 = \frac{2}{log_2(1+1)} + \frac{1}{log_2(1+2)} + \frac{0}{log_2(1+3)} = 2.63$$

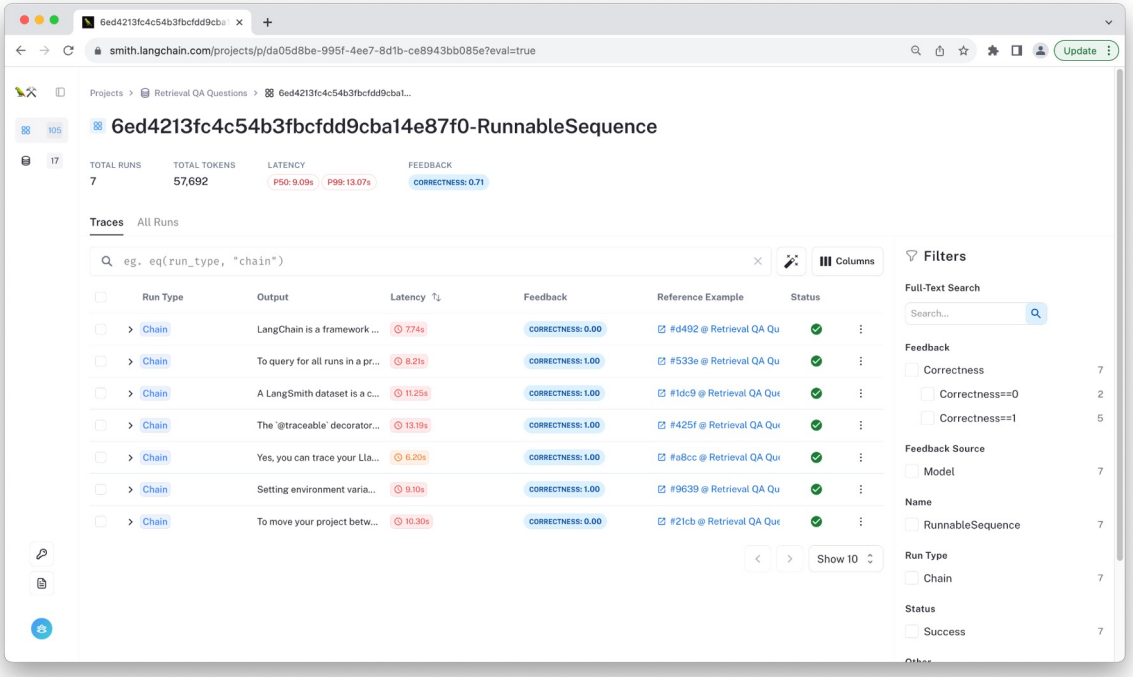
$$NDCG@3 = \frac{2.3}{2.6} = 0.86$$

Evaluation

LangSmith

You can also specify your own metric(s) such as:

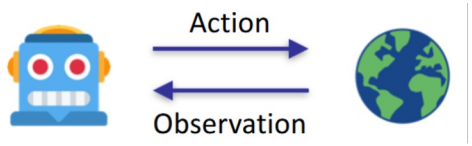
- answer conciseness, fluency, etc.; these metrics can be evaluated by a strong LLM like GPT-4o (refer to LLM-as-a-judge).
- or even the number of tokens spent.



https://arxiv.org/abs/2306.05685 (LLM-as-a-judge)
https://www.langchain.com/langsmith

Agentic LLMs

What is "Agent"



An "intelligent" system that interacts with some "environment"

- Physical environments: autonomous car
- Digital environments: ChatGPT

AI Agent Example

ELIZA (1966): Symbolic AI agent

```
  Welcome to
                EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                EE      LL       II       ZZ   AA   AA
                EEEEE   LL       II     ZZZ    AAAAAAA
                EE      LL       II    ZZ      AA   AA
                EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

- Domain specific!
- Requires manual design
- Cannot work beyond simple domains

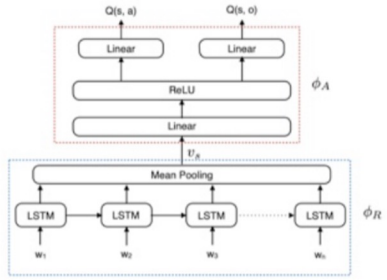
AI Agent Example

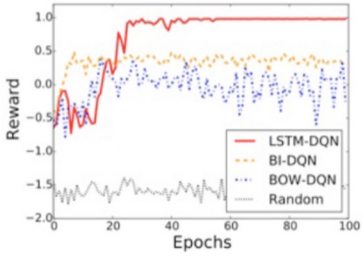
LSTM-DQN (2015): Deep RL agent

State 1: The old bridge
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

Command: **Go east**

State 2: Ruined gatehouse
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.
Exits: Standing archway, castle corner, Bridge over the abyss



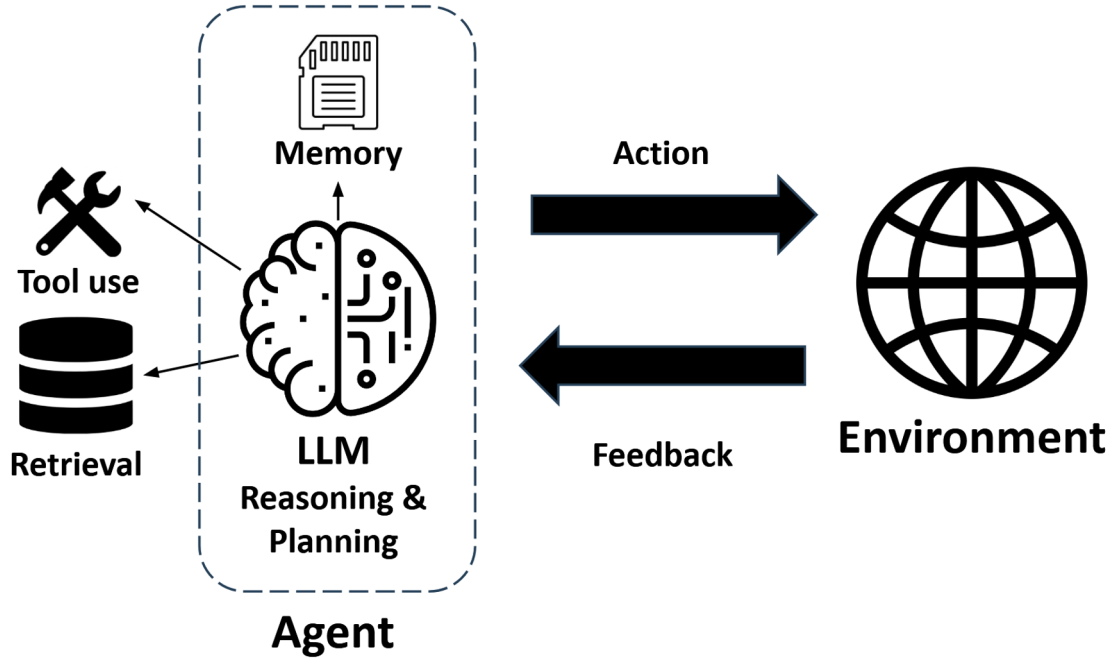


- Domain specific!
- Requires scalar reward signals
- Requires extensive training

Language understanding for text-based games using deep reinforcement learning. EMNLP 2015.

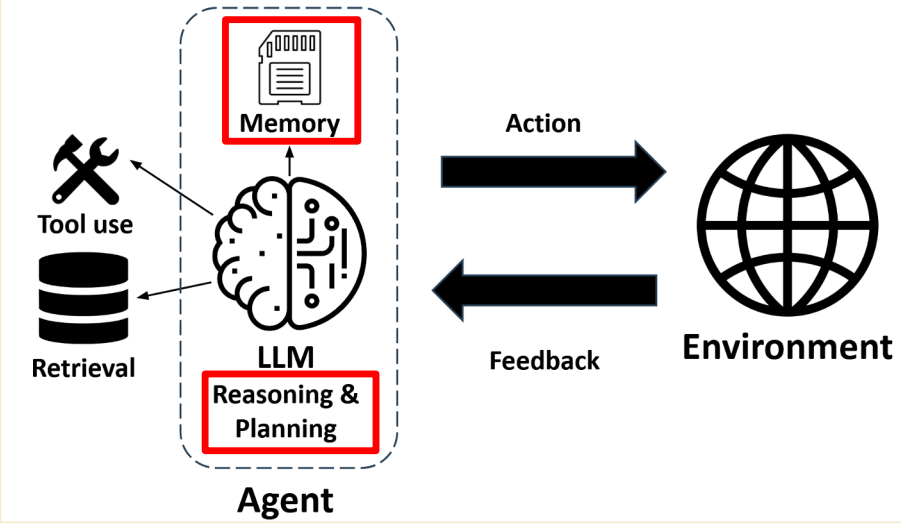
Agentic LLMs

An agentic LLM is a large language model (LLM) that operates as an autonomous agent, capable of performing tasks, making decisions, and interacting with external tools or APIs to achieve the specified goals.



Benefit of Agentic LLM

- Enables LLMs to be able to solve complex real-world tasks through:
  - Task decomposition (planning)
  - Allocation of subtasks to specialized modules (tools, expert models, etc.)
  - Multi-agent generation (improves responses quality)
- Expands LLMs' capabilities with external tools and knowledge.



[Reasoning1: Basic Reasoning Model]
Reasoning - Why do LLMs need reasoning?

Humans can learn new tasks with only a handful of examples because **we can reason** [1].

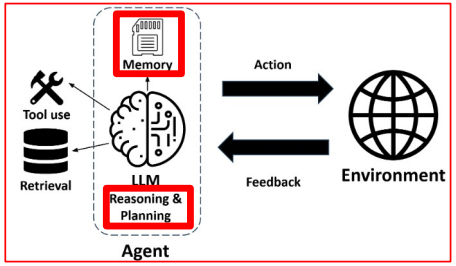
Suppose we have this simple task:

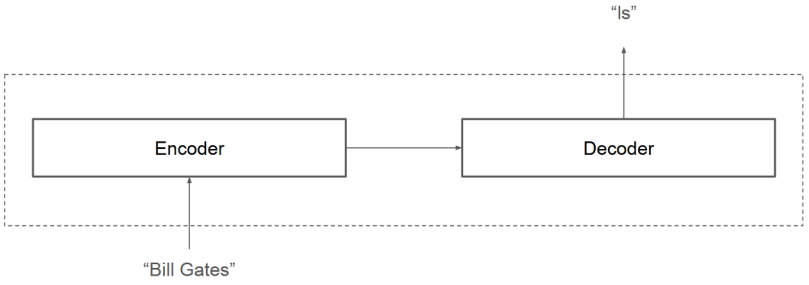
Last Letter Concatenation

Rule: Take the last letter of each word, and then concatenate them

Input	Output
"Elon Musk"	"nk"
"Bill Gates"	"ls"
"Barack Obama"	?

Solve it by ML? Tons of labeled data needed!

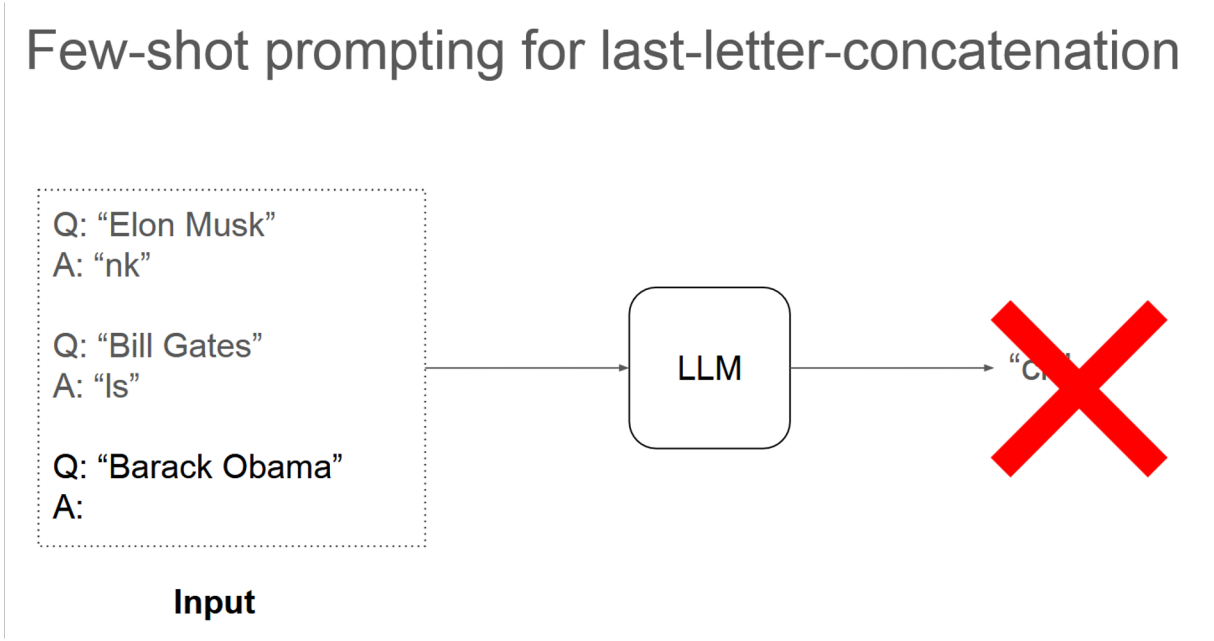




[1] http://llmagents-learning.org/slides/llm-reasoning.pdf

Reasoning - Why do LLMs need reasoning?

Simply prompting LLMs with few-shot examples probably won't work as well.



[1] http://llmagents-learning.org/slides/llm-reasoning.pdf

Reasoning - Why do LLMs need reasoning?

However, giving the reasoning demonstration to the LLM allows it to complete the task easily.

One demonstration is enough, like humans

Q: “Elon Musk”
A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk". so the output is "nk".

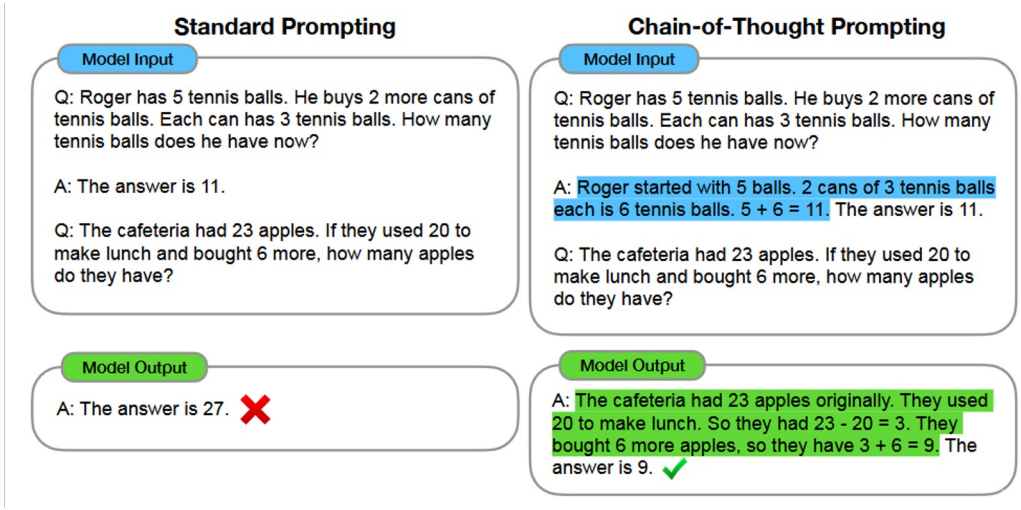
Q: “Barack Obama"
A: the last letter of "Barack" is "k". the last letter of "Obama" is "a". Concatenating "k", "a" leads to "ka". so the output is "ka".

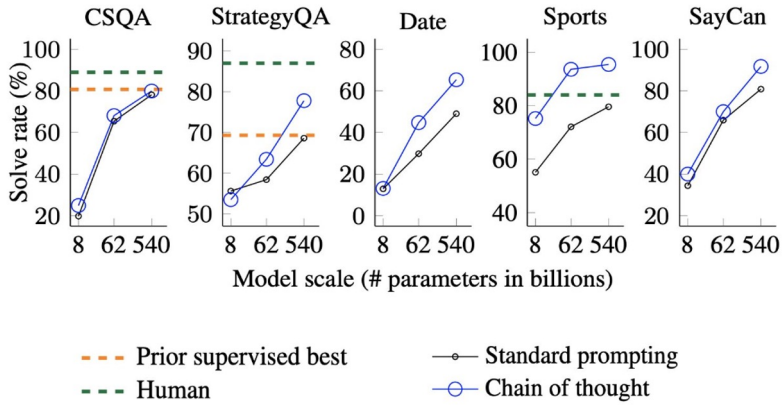
100% accuracy with only one demonstration example

[1] http://llmagents-learning.org/slides/llm-reasoning.pdf

Reasoning - How do we achieve reasoning? (1)

Key Idea: Derive the Final Answer through **Intermediate Steps**.





https://arxiv.org/abs/2201.11903

Reasoning - How do we achieve reasoning? (2)

[1] proves, using the computational complexity theory, that CoT allows transformer to solve significantly more complex problems.

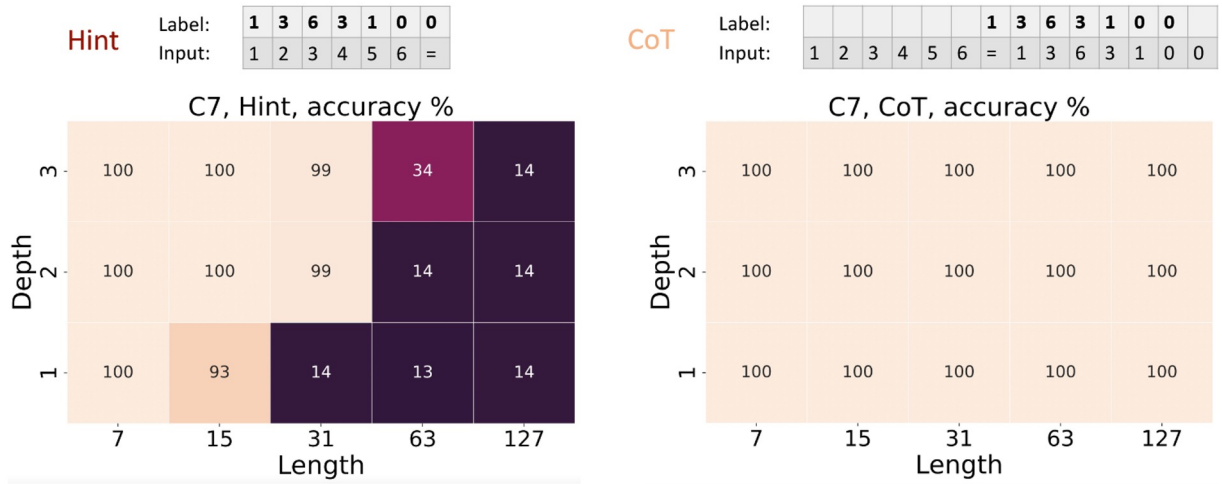
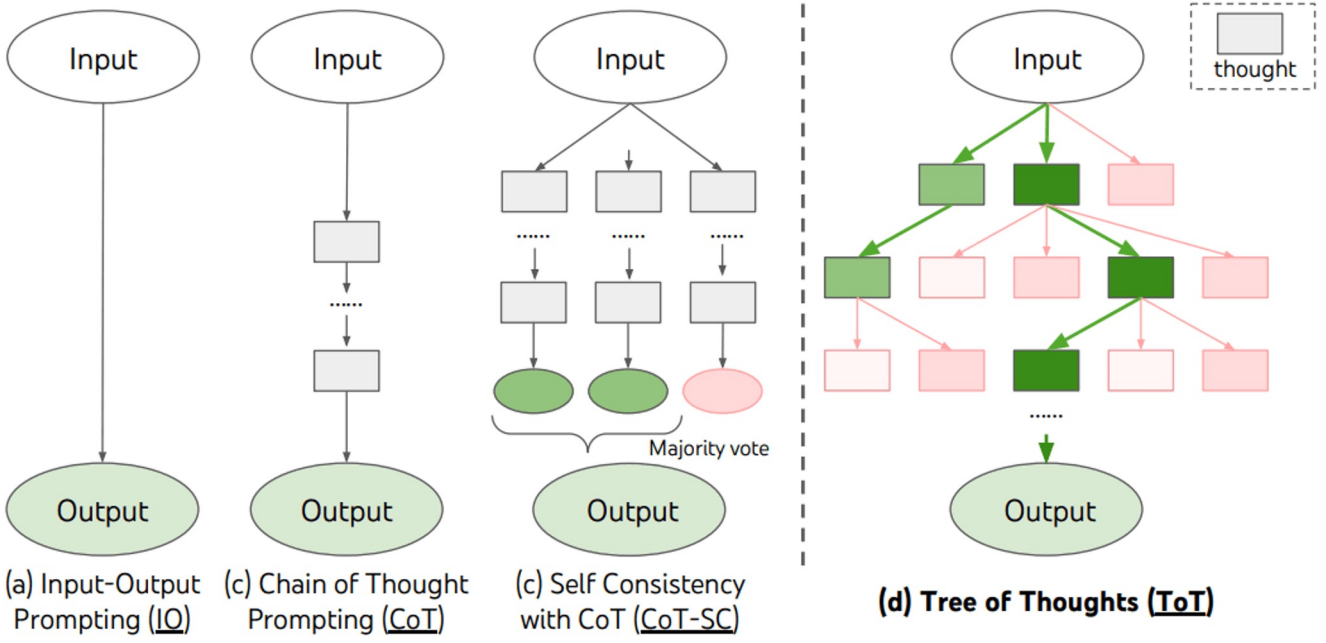


Figure 2: Modular Addition(C_7). The label is the sum of the inputs modulo a positive integer, which is 7 in this case. The chain of thoughts and hints are the partial modular sum. Low-depth transformers with hint can solve this task well for a reasonable input sequence length, but with cot the performance is much better, especially with a long input sequence, as predicted by our Theorem 3.3. See experiments for C_2 in Figure 5.

[1] https://openreview.net/forum?id=3EWTEy9MTM

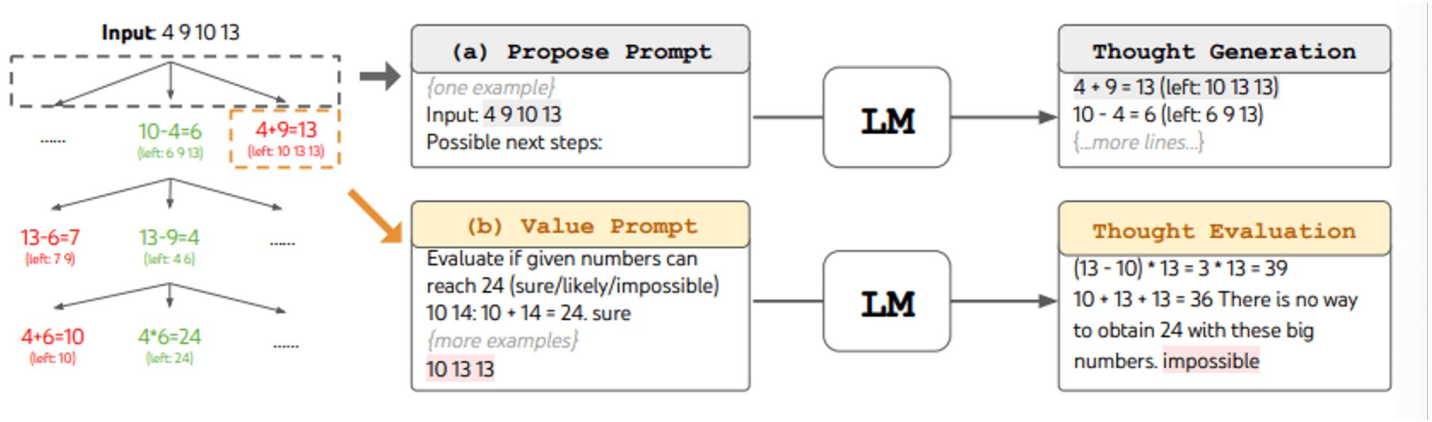
Reasoning - Tree-of-Thought Prompting (1)

ToT maintains a tree of thoughts, where thoughts represent coherent language sequences that serve as intermediate steps toward solving a problem.



https://arxiv.org/abs/2305.10601

Reasoning - Tree-of-Thought Prompting (2)



Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

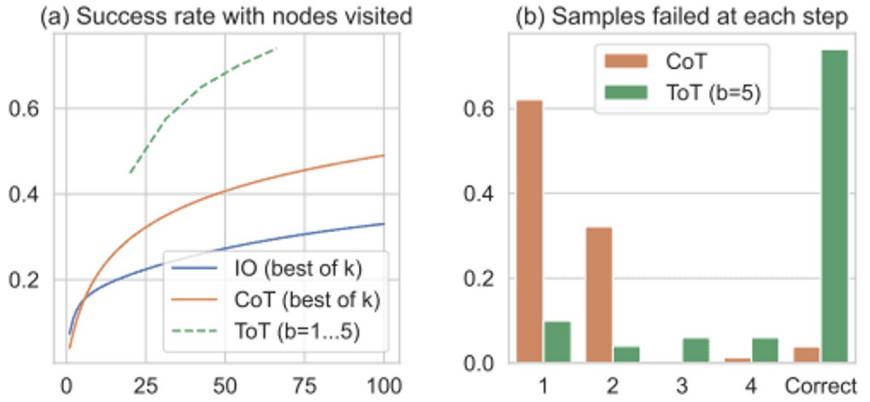


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Reasoning - Problems

- LLMs are not trained for predicting reasoning paths but instead just the next token - not true for reasoning models!

- LLMs Can Be Easily Distracted by Irrelevant Context [1]
- LLMs Cannot Self-Correct Reasoning Yet [2]
- Premise Order Matters in LLM Reasoning [3]

Adding irrelevant contexts to GSM8K leads to 20+ points performance drop

[Prompt] Lucy has $65 in the bank. She made a $15 deposit and then followed by a $4 withdrawal. Maria's monthly rent is $10. What is Lucy's bank balance?

[Response] Lucy's bank balance is $65 + $15 - $4 - $10 = $66. The answer is $66.

Terry eats 2 yogurts a day. They are currently on sale at 4 yogurts for $5.00. How much does he spend on yogurt over 30 days?

... So, Terry spends $75 on yogurt over 30 days. (Correct)

Review your previous answer and find problems with your answer.

Upon reviewing my previous answer, I realized that I made a mistake in calculating Terry's spending on yogurt over 30 days. I incorrectly stated that Terry spends $2.50 per day for 30 days…

Based on the problems you found, improve your answer.

... the final answer is Terry spends $37.5 on yogurt over 30 days. (Wrong)

[Original GSM8K Problem] Thomas withdraws $1000 in 20 dollar bills from the bank account. He loses 10 bills while getting home. After that, he uses half of the remaining bills to pay for a bill. Thomas then triples his money. He then converts all his bills to 5 dollar bills. How many 5 dollar bills does he have?

[Reordered GSM8K Problem] Thomas withdraws $1000 in 20 dollar bills from the bank account. After getting home, he uses half of the remaining bills to pay for a bill. Thomas then triples his money. He then converts all his bills to 5 dollar bills. He loses 10 bills while getting home. How many 5 dollar bills does he have?

About 10 points drop on solving rates across all frontier LLMs

[1] https://arxiv.org/abs/2302.00093
[2] https://arxiv.org/abs/2310.01798
[3] https://arxiv.org/abs/2402.08939

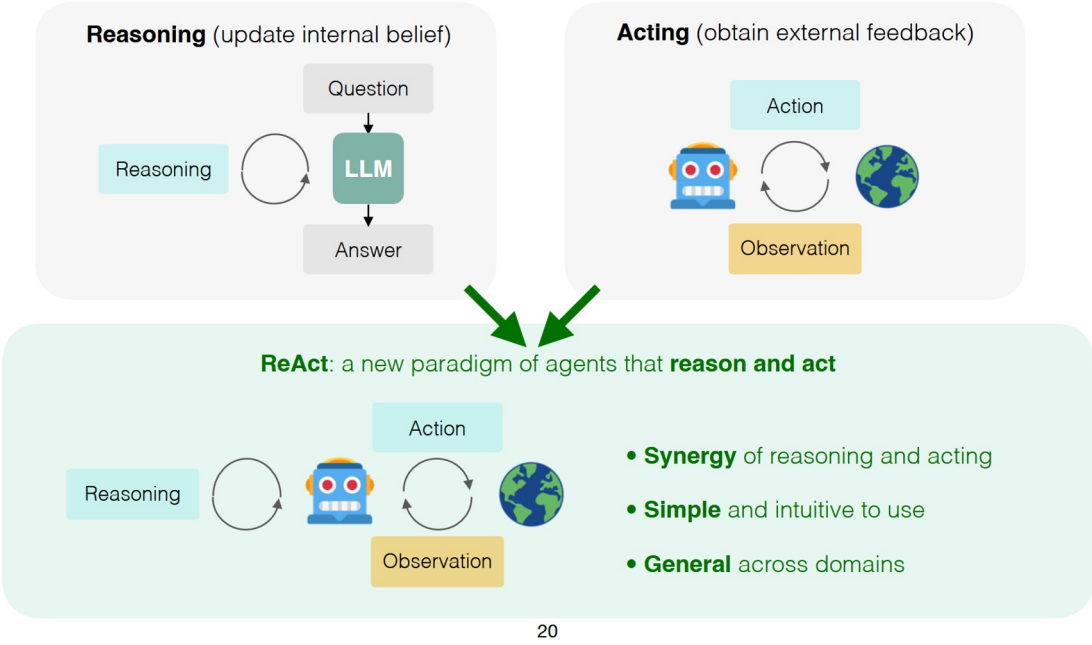
[Reasoning2: ReAct]

ReAct: Synergizing Reasoning and Acting in Language Models

ReAct = Reasoning + Acting

Allows LLMs perform dynamic reasoning to **create, maintain, and adjust plans** for acting while also enabling interaction to external environments.

Unlocks **agentic capability**.



https://arxiv.org/abs/2210.03629

ReAct Example

Question

Aside from the Apple Remote, what other devices can control the program Apple Remote was originally designed to interact with?

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys]

ReAct Benefit

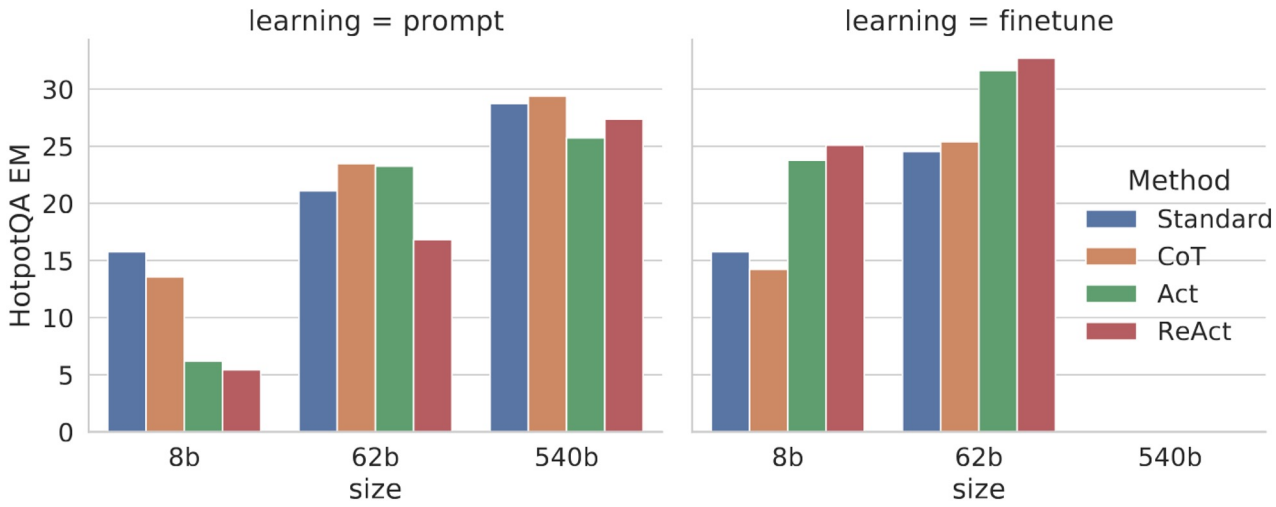


Figure 3: Scaling results for prompting and finetuning on HotPotQA with `ReAct` (ours) and baselines.

	Type	Definition	ReAct	CoT
Success	True positive	Correct reasoning trace and facts	94%	86%
	False positive	Hallucinated reasoning trace or facts	6%	14%
Failure	Reasoning error	Wrong reasoning trace (including failing to recover from repetitive steps)	47%	16%
	Search result error	Search return empty or does not contain useful information	23%	-
	Hallucination	Hallucinated reasoning trace or facts	0%	56%
	Label ambiguity	Right prediction but did not match the label precisely	29%	28%

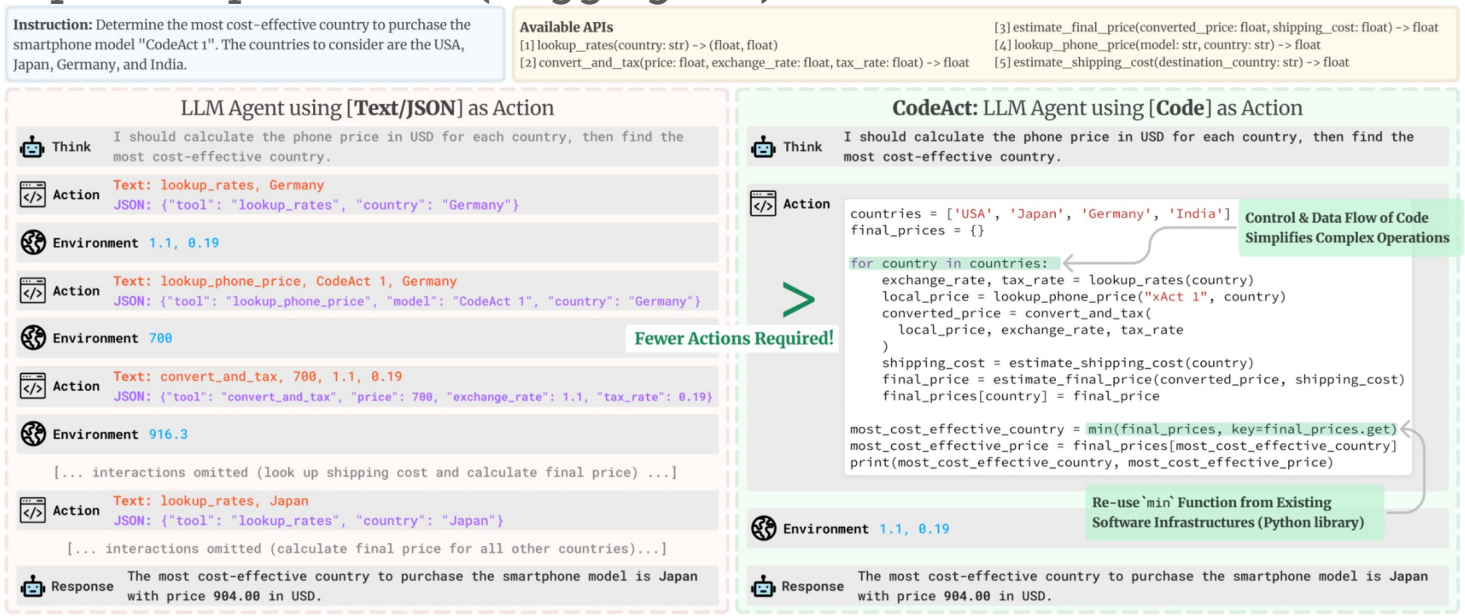
ReAct: How's it work?

Pretty much the standard on which AI agents nowadays operate **using ReAct**.

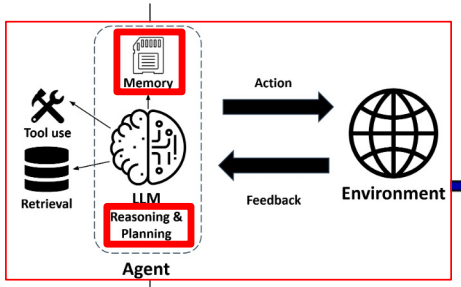
How it works — OpenAI Deep Research

Deep research was trained using end-to-end reinforcement learning on hard browsing and reasoning tasks across a range of domains. Through that training, it learned to plan and execute a multi-step trajectory to find the data it needs, backtracking and reacting to real-time information where necessary. The model is also able to browse over user uploaded files, plot and iterate on graphs using the python tool, embed both generated graphs and images from websites in its responses, and cite specific sentences or passages from its sources. As a result of this training, it reaches new highs on a number of public evaluations focused on real-world problems.

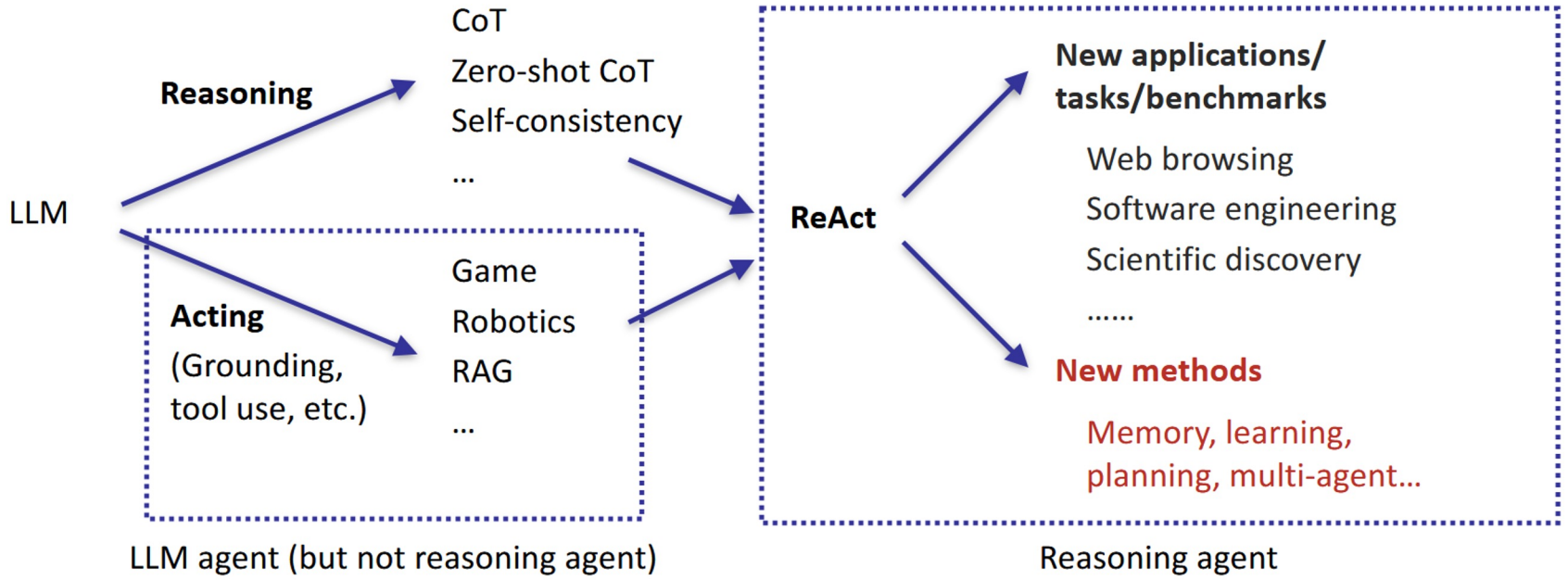
Open Deep Research (Huggingface)



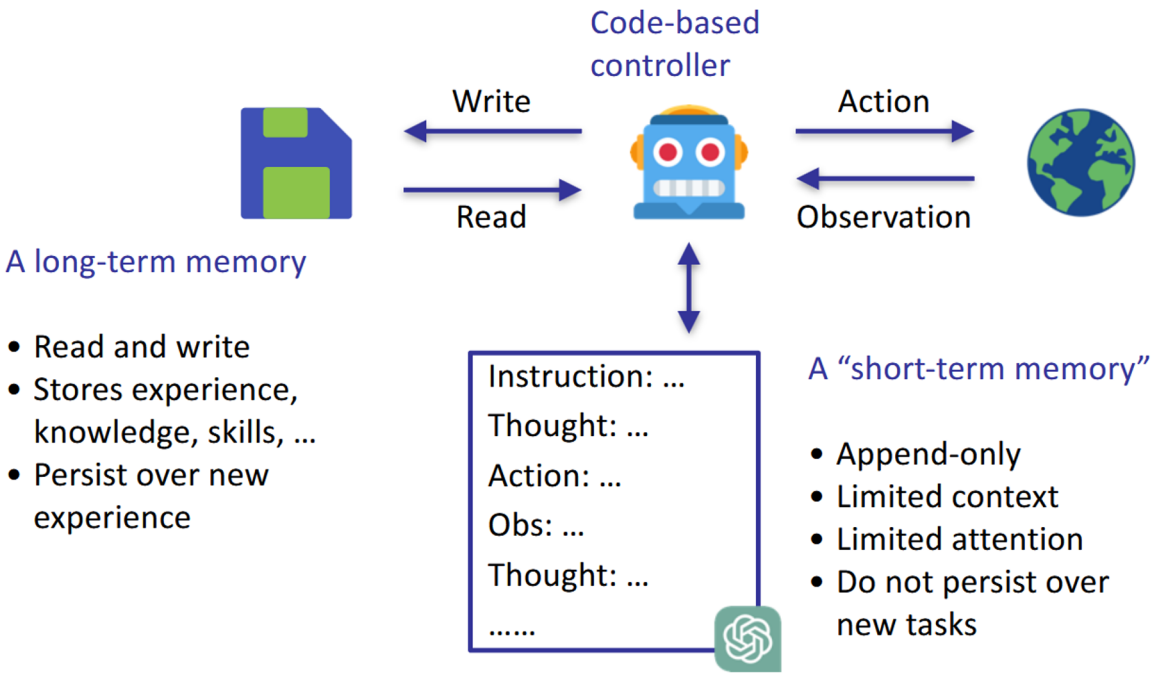
+ Agentic LLMs: Memory [Memory]



A brief history of LLM agents



Long-term memory



Long-term memory

As opposed to short-term memory, e.g.in context

- Text-based, like diary
- Code-based
- Episodic memory - like log stream

Long-term memory persists over new tasks, essentially making LLMs look like they are learning

Real-World Agentic Applications

Digital automation

File reports

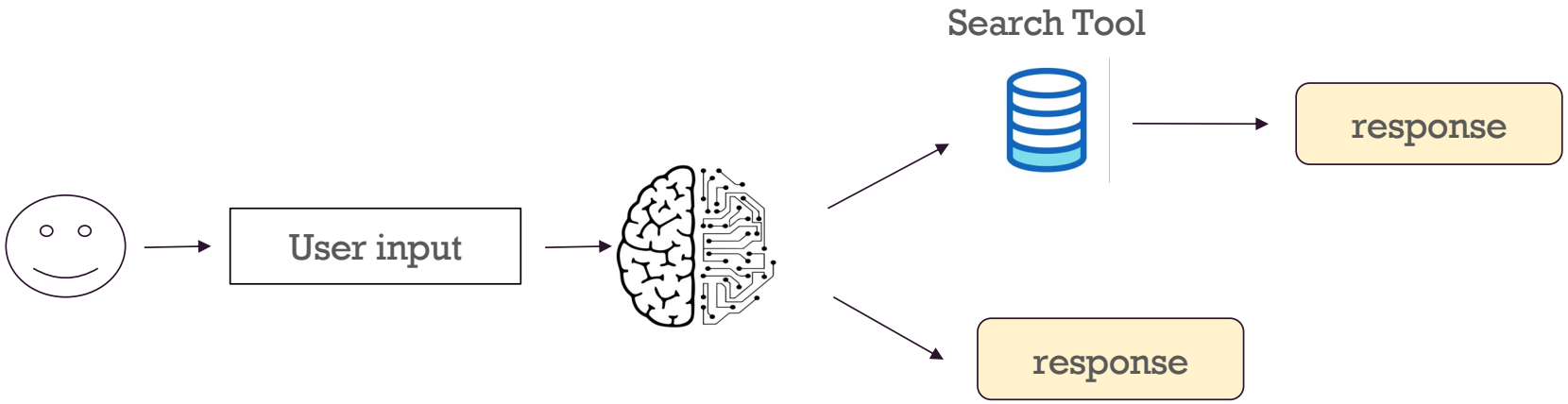
Code experiments

Explore papers

- Tremendous practical values, but little progress (think about Siri)
- **Underlying research challenges**:
  - Reasoning over real-world language (and other modalities)
  - Decision making over open-ended actions and long horizon

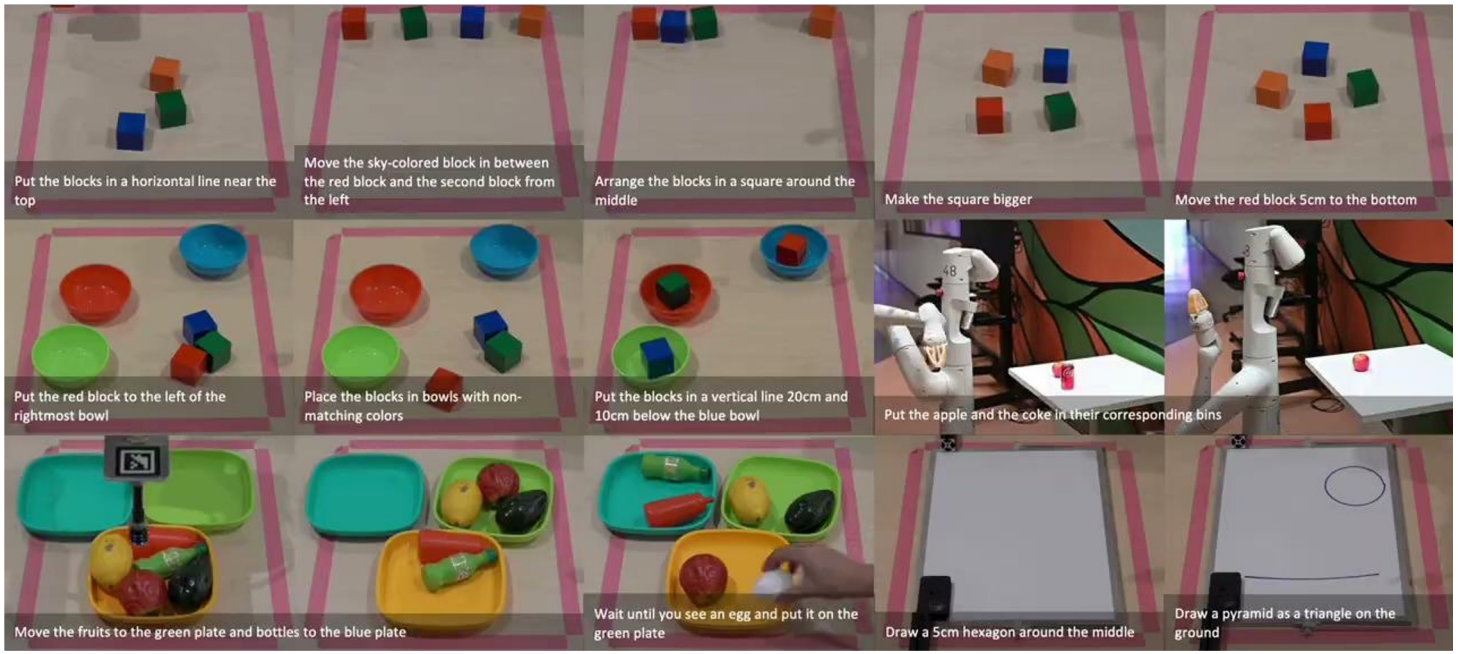
Example 1: Agentic RAG

- Not all messages need supporting documents for LLMs to respond.
- Instead, give the LLM a vector search "tool" (also sometimes referred to as function calling).
- Now, it can automatically decide when to use the tool.



Example 2: Robotics

- Give an LLM access to functions that control robots, such as move, grab, etc.
- The LLM will decide which functions are required (and when to use each) for each query.

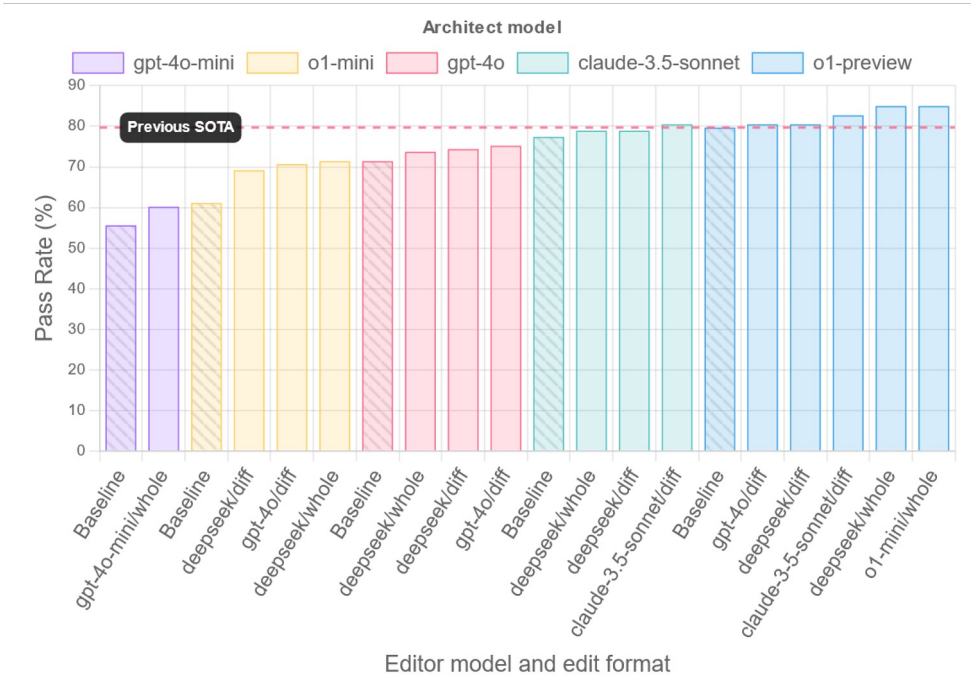


https://code-as-policies.github.io/

Example 3: Multi-agent

- Why stop at just a single AI? We can also use multiple of them to complete a task.

```
> /code Add a function to calculate the factorial of a number
Certainly! I'll add a function to calculate the factorial of a number.
Here's the change:

def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage
print(factorial(5))  # Output: 120

This function uses recursion to calculate the factorial. It's been added
to your code.

> /architect Can we make this simpler?
Yes, you could replace the factorial() function with a call to the
math.factorial() builtin. This way you don't have to maintain a custom
factorial implementation, and the builtin function is well optimized.

Edit the files? (Y)es/(N)o [Yes]: Yes

<<<<<<< SEARCH
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
=======
from math import factorial
>>>>>>> REPLACE
```



Tools & Framework

Agentic

Microsoft AutoGen

An Open-Source Programming Framework for Agentic AI
autogen@microsoft.com

Monitoring

LangSmith

Challenges

- Reasoning and planning
  - LLM agents tend to make mistakes when performing complex tasks end-to-end
- Embodiment and learning from environment feedback
  - LLM agents are not yet efficient at recovering from mistakes for long-horizon tasks
  - Continuous learning, self-improvement
  - Multimodal understanding, grounding and world models
- Multi-agent learning, theory of mind
- Safety and privacy
  - LLMs are susceptible to adversarial attacks, can emit harmful messages and leak private data
- Human-agent interaction, ethics
  - How to effectively control the LLM agent behavior, and design the interaction mode between humans and LLM agents

Overall LLM Challenges

Air Canada has been held liable for a negligent misrepresentation made to a customer by one of its chatbots in a case that one expert said highlights broader risks businesses must consider when adopting AI tools.

According to Moffat's screenshot of a conversation with the chatbot, the British Columbia resident was told he could apply for the refund "within 90 days of the date your ticket was issued" by completing an online form.

Moffatt then booked tickets to and from Toronto to attend the funeral of a family member. But when he applied for a refund, Air Canada said bereavement rates did not apply to completed travel and pointed to the bereavement section of the company's website.

https://www.theguardian.com/world/2024/feb/16/air-canada-chatbot-lawsuit

Evaluation Tools/Benchmark for Agentic LLMs

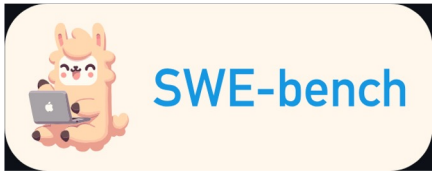
SWE bench

τ -bench

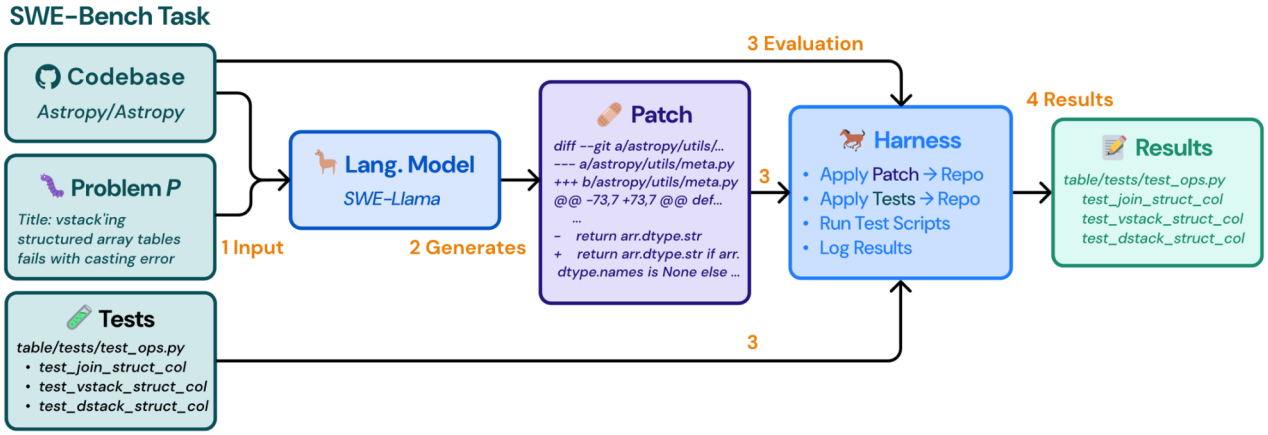
WebArena

GAIA

1) SWE-bench

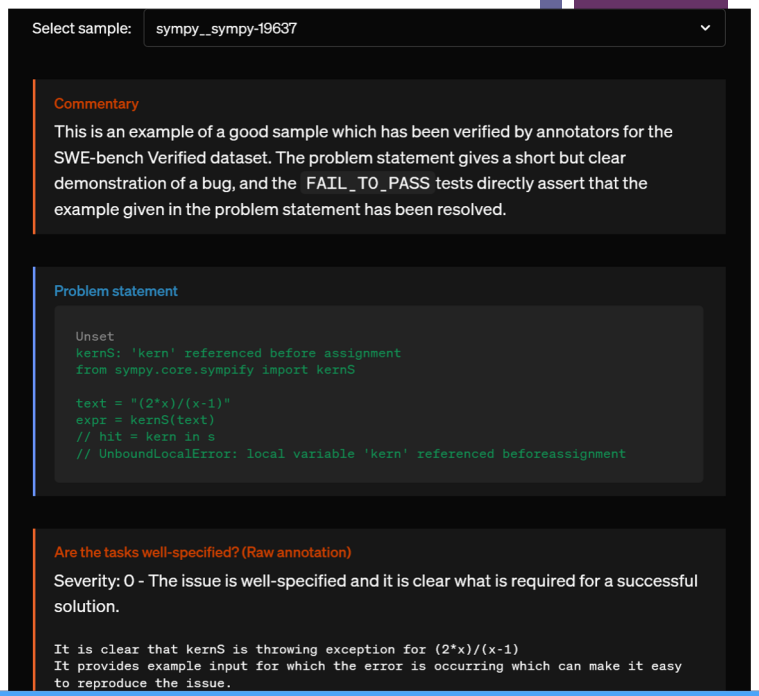


Each sample in the SWE-bench test set is created from a resolved GitHub issue in one of 12 open-source Python repositories on GitHub, such as scikit-learn.

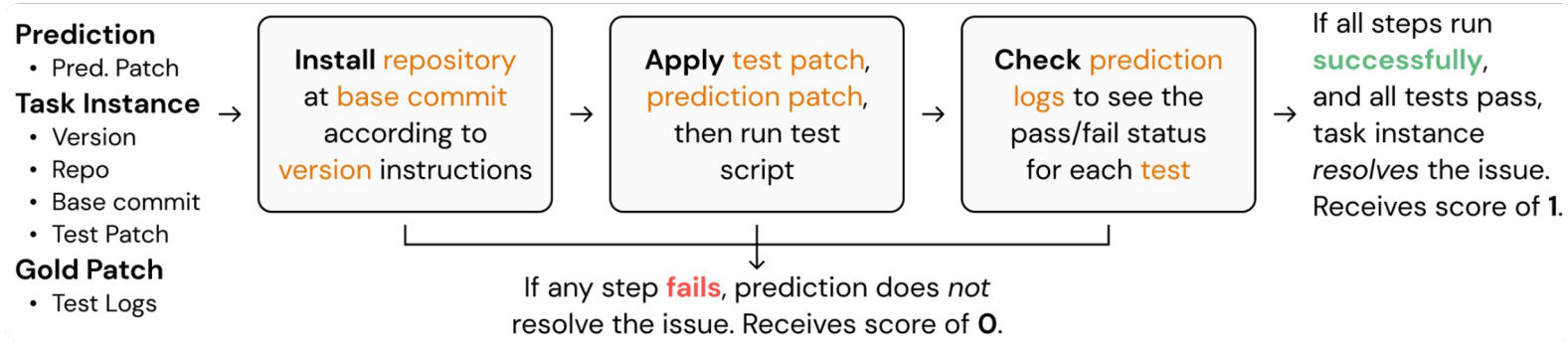


Also a multimodal version!

http://www.swebench.com/



1) SWE-bench (cont.)



Leaderboard

Lite | Verified | Full | Multimodal

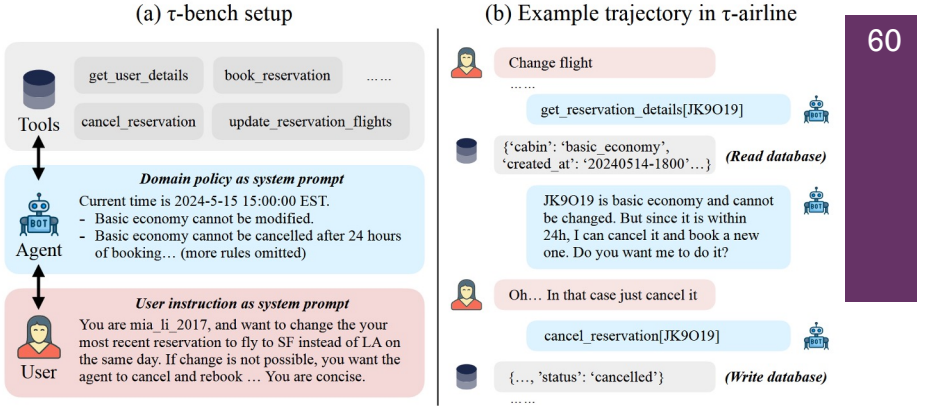
Model	% Resolved	Org	Date	Logs	Trajs	Site
OpenHands + CodeAct v2.1 (claude-3-5-sonnet-20241022)	29.38		2024-11-03	✓	✓	
AutoCodeRover-v2.0 (Claude-3.5-Sonnet-20241022)	24.89		2024-11-21	✓	✓	
Honeycomb	22.06		2024-08-20	✓	✓	
Amazon Q Developer Agent (v20240719-dev)	19.75	aws	2024-07-21	✓	✓	
Factory Code Droid	19.27		2024-06-17	✓	-	
AutoCodeRover (v20240620) + GPT 4o (2024-05-13)	18.83		2024-06-28	✓	-	
SWE-agent + Claude 3.5 Sonnet	18.13		2024-06-20	✓	✓	-
AppMap Navie + GPT 4o (2024-05-13)	14.60		2024-06-15	✓	-	
Amazon Q Developer Agent (v20240430-dev)	13.82	aws	2024-05-09	✓	-	
SWE-agent + GPT 4 (1106)	12.47		2024-04-02	✓	✓	
SWE-agent + GPT 4o (2024-05-13)	11.99		2024-07-28	✓	✓	
SWE-agent + Claude 3 Opus	10.51		2024-04-02	✓	✓	-
RAG + Claude 3 Opus	3.79		2024-04-02	✓	-	
RAG + Claude 2	1.96		2023-10-10	✓	-	-
RAG + GPT 4 (1106)	1.31		2024-04-02	✓	-	-
RAG + SWE-Llama 13B	0.70		2023-10-10	✓	-	-
RAG + SWE-Llama 7B	0.70		2023-10-10	✓	-	-
RAG + ChatGPT 3.5	0.17		2023-10-10	✓	-	-

2) τ-bench

TAU-bench: Tool-Agent-User benchmark in real-world domains

- Key Characteristics:
  - Realistic dialogue and tool use
  - Open-ended and diverse tasks
  - Faithful rule-based evaluation
  - Modular extension
- The reward of a task episode is based on:
  - whether the final database is identical to the unique ground truth outcome database (r_{action}), and
  - whether the agent's responses to the user contain all necessary information (r_{output}).

https://github.com/sierra-research/tau-bench




```
{"order_id": "#W2890441",
"user_id": "mei_davis_8935",
"items": [{
    "name": "Water Bottle",
    "product_id": "8310926033",
    "item_id": "2366567022",
    "price": 54.04,
    "options": {
        "capacity": "1000ml",
        "material": "stainless
        steel",
        "color": "blue"
    }}, ...], ...}
```

(a) An orders database entry in τ-retail.

```
def return_delivered_order_items(
    order_id: str,
    item_ids: List[str],
    payment_method_id: str,
) -> str: ...

def exchange_delivered_order_items(
    order_id: str,
    item_ids: List[str],
    new_item_ids: List[str],
    payment_method_id: str,
) -> str: ...
```

(b) An API tool in τ-retail.

```
## Return delivered order
- After user confirmation, the order status
will be changed to 'return requested'...

## Exchange delivered order
- An order can only be exchanged if its
status is 'delivered'...
```

(c) Domain policy excerpts in τ-retail.

```
{"instruction": "You are Mei Davis in 80217.
You want to return the water bottle, and
exchange the pet bed and office chair to the
cheapest version. Mention the two things
together. If you can only do one of the two
things, you prefer to do whatever saves you
most money, but you want to know the money
you can save in both ways. You are in debt
and sad today, but very brief.",
"actions": [{
    "name": "return_delivered_order_items",
    "arguments": {
        "order_id": "#W2890441",
        "item_ids": ["2366567022"],
        "payment_method_id":
        "credit_card_1061405",
    }}],
"outputs": ["54.04", "41.64"]}
```

(d) User instruction ensures only one possible outcome.

2) τ –bench (cont.)

Leaderboard

Airline

Strategy	Pass^1	Pass^2	Pass^3	Pass^4
TC (claude-3-5-sonnet-20241022)	**0.460**	**0.326**	**0.263**	**0.225**
TC (gpt-4o)	0.420	0.273	0.220	0.200
TC (claude-3-5-sonnet-20240620)	0.360	0.224	0.169	0.139
TC (mistral-large-2407)	??	??	??	??
TC (gpt-4o-mini)	0.225	0.140	0.110	0.100
Act (gpt-4o)	0.365	0.217	0.160	0.140
ReAct (gpt-4o)	0.325	0.233	0.185	0.160

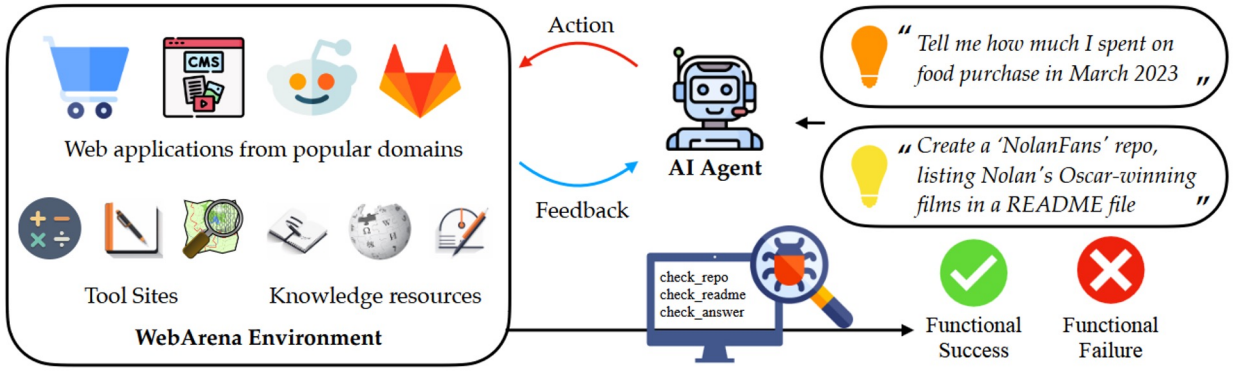
Retail

Strategy	Pass^1	Pass^2	Pass^3	Pass^4
TC (claude-3-5-sonnet-20241022)	**0.692**	**0.576**	**0.509**	**0.462**
TC (gpt-4o)	0.604	0.491	0.430	0.383
TC (claude-3-5-sonnet-20240620)	0.626	0.506	0.435	0.387
TC (mistral-large-2407)	??	??	??	??
TC (gpt-4o-mini)	??	??	??	??
Act (gpt-4o)	??	??	??	??
ReAct (gpt-4o)	??	??	??	??

*TC = `tool-calling` strategy (the function-calling strategy reported in the paper)

3) WebArena

WebArena is a standalone, self-hostable web environment **for building autonomous agents.**

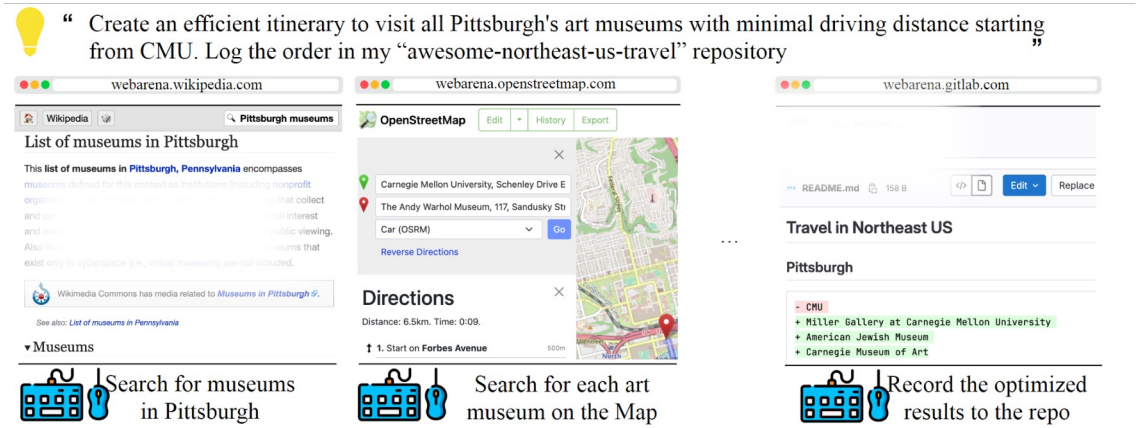


Action Type	Description
`noop`	Do nothing
`click(elem)`	Click at an element
`hover(elem)`	Hover on an element
`type(elem, text)`	Type to an element
`press(key_comb)`	Press a key comb
`scroll(dir)`	Scroll up and down
`tab_focus(index)`	focus on i-th tab
`new_tab`	Open a new tab
`tab_close`	Close current tab
`go_back`	Visit the last URL
`go_forward`	Undo `go_back`
`goto(URL)`	Go to URL

Figure 4: Action Space of `WebArena`

Category	Example
Information Seeking	When was the last time I bought shampoo
	Compare walking and driving time from AMC Waterfront to Randyland
Site Navigation	Checkout merge requests assigned to me
	Show me the ergonomic chair with the best rating
Content & Config	Post to ask "whether I need a car in NYC"
	Delete the reviews from the scammer Yoke

Figure 5: Example intents from three categories.



https://webarena.dev/

3) WebArena (cont.)

Agent	WebArena
GenericAgent-Claude-3.5-Sonnet	36.20
GenericAgent-GPT-4o	31.40
GenericAgent-GPT-o1-mini	28.60
GenericAgent-Llama-3.1-405b	24.00
GenericAgent-Llama-3.1-70b	18.40
GenericAgent-GPT-4o-mini	17.40

Function	ID	Intent	Eval Implementation
$r_{\text{info}}(a^*, \hat{a})$	1	Tell me the name of the customer who has the most cancellations in the history	`exact_match`($\hat{a}$, "Samantha Jones")
	2	Find the customer name and email with phone number 8015551212	`must_include`($\hat{a}$, "Sean Miller") `must_include`($\hat{a}$, "sean@gmail.com")
	3	Compare walking and driving time from AMC Waterfront to Randyland	`fuzzy_match`($\hat{a}$, "walking: 2h58min") `fuzzy_match`($\hat{a}$, "driving: 21min")
$r_{\text{prog}}(\mathbf{s})$	4	Checkout merge requests assigned to me	`url=locate_current_url(s)` `exact_match(URL, "gitlab.com/merge_requests?assignee_username=byteblaze")`

	5	Post to ask "whether I need a car in NYC"	`url=locate_latest_post_url(s)`
			`body=locate_latest_post_body(s)`
			`must_include(URL, "/f/nyc")`
			`must_include(body, "a car in NYC")`

Table 1: We introduce two evaluation approaches. r_{info} (top) measures the correctness of performing information-seeking tasks. It compares the predicted answer $\hat{a}$ with the annotated reference a^* with three implementations. r_{prog} (bottom) programmatically checks whether the intermediate states during the executions possess the anticipated properties specified by the intent.

4) GAIA: A Benchmark for General AI Assistants

Real-world questions that require a set of fundamental abilities such as reasoning, multi-modality handling, web browsing, and generally tool-use proficiency.

GAIA questions are conceptually simple for humans yet challenging for most advanced AIs

Level 1

Question: What was the actual enrollment count of the clinical trial on H. pylori in acne vulgaris patients from Jan-May 2018 as listed on the NIH website?
Ground truth: 90

Level 2

Question: If this whole pint is made up of ice cream, how many percent above or below the US federal standards for butterfat content is it when using the standards as reported by Wikipedia in 2020? Answer as + or - a number rounded to one decimal place.
Ground truth: +4.6

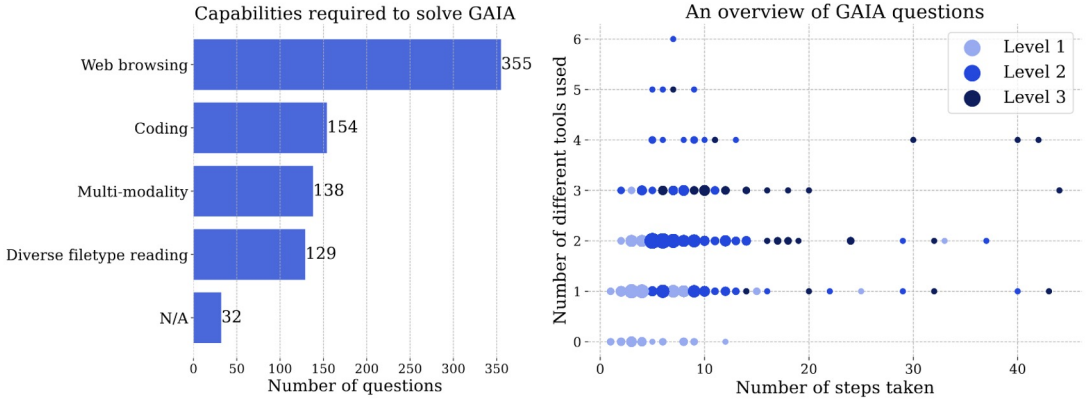
Level 3

Question: In NASA's Astronomy Picture of the Day on 2006 January 21, two astronauts are visible, with one appearing much smaller than the other. As of August 2023, out of the astronauts in the NASA Astronaut Group that the smaller astronaut was a member of, which one spent the least time in space, and how many minutes did he spend in space, rounded to the nearest minute? Exclude any astronauts who did not spend any time in space. Give the last name of the astronaut, separated from the number of minutes by a semicolon. Use commas as thousands separators in the number of minutes.
Ground truth: White; 5876

GAIA: A Benchmark for General AI Assistants

Real-world questions that require a set of fundamental abilities such as reasoning, multi-modality handling, web browsing, and generally tool-use proficiency.

GAIA questions are conceptually simple for humans yet challenging for most advanced AIs



Level 1

Question: What was the actual enrollment count of the clinical trial on H. pylori in acne vulgaris patients from Jan-May 2018 as listed on the NIH website?
Ground truth: 90

Level 2

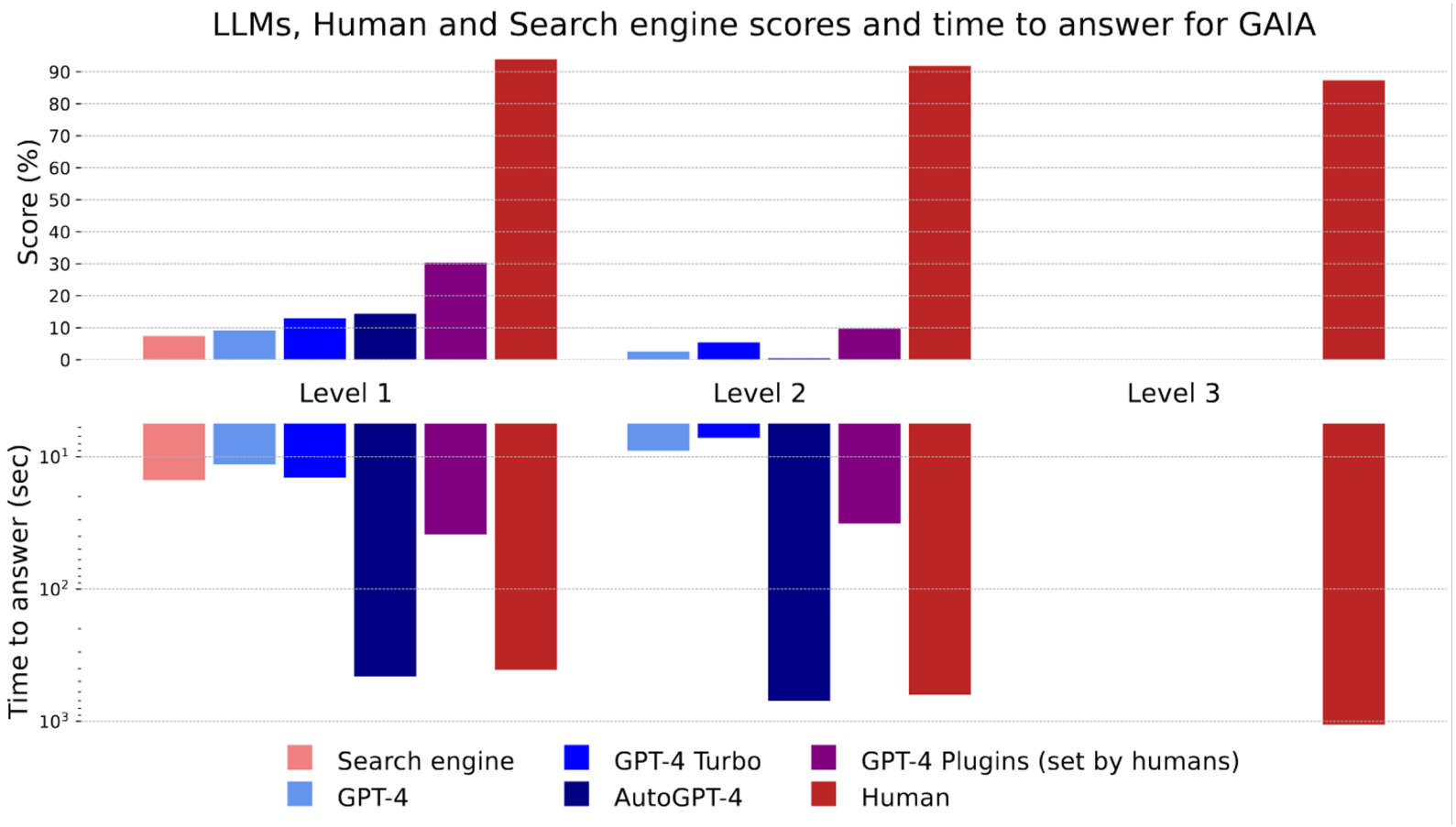
Question: If this whole pint is made up of ice cream, how many percent above or below the US federal standards for butterfat content is it when using the standards as reported by Wikipedia in 2020? Answer as + or - a number rounded to one decimal place.
Ground truth: +4.6

Level 3

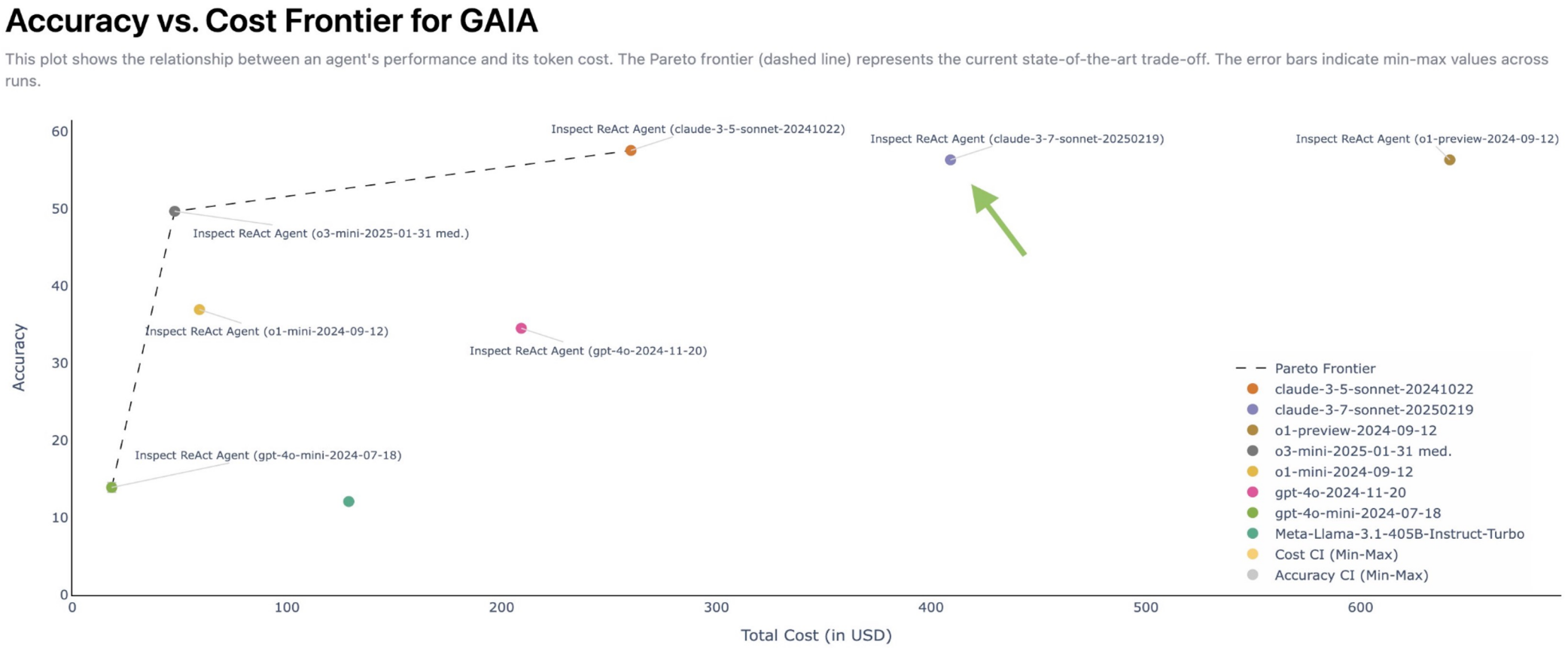
Question: In NASA's Astronomy Picture of the Day on 2006 January 21, two astronauts are visible, with one appearing much smaller than the other. As of August 2023, out of the astronauts in the NASA Astronaut Group that the smaller astronaut was a member of, which one spent the least time in space, and how many minutes did he spend in space, rounded to the nearest minute? Exclude any astronauts who did not spend any time in space. Give the last name of the astronaut, separated from the number of minutes by a semicolon. Use commas as thousands separators in the number of minutes.
Ground truth: White; 5876

https://arxiv.org/pdf/2311.12983

GAIA: A Benchmark for General AI Assistants



GAIA: A Benchmark for General AI Assistants

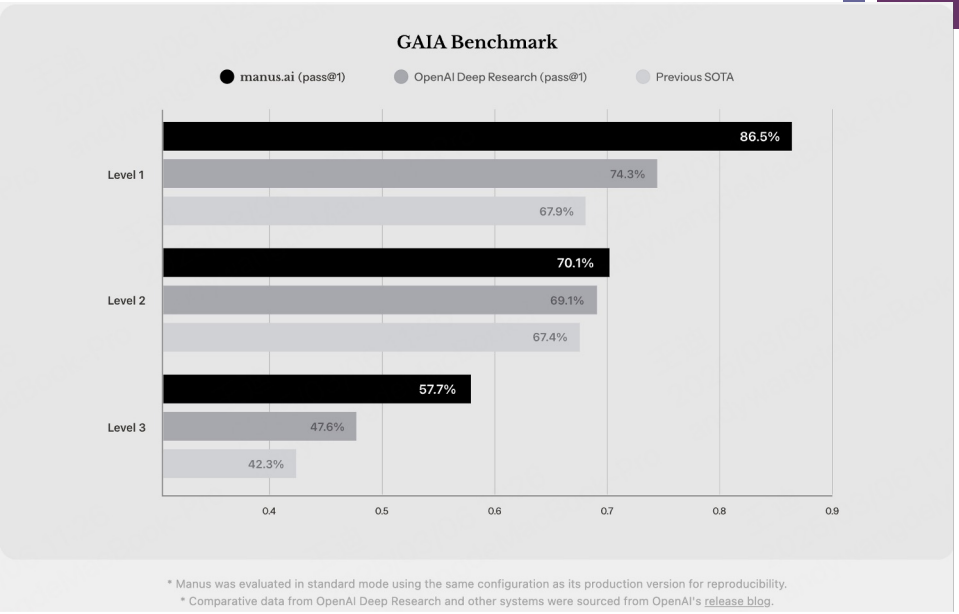


https://x.com/benediktstroebl/status/1895148142926180376

GAIA: A Benchmark for General AI Assistants

OpenAI Deep Research

GAIA				
	Level 1	**Level 2**	**Level 3**	**Avg.**
Previous SOTA ↗	67.92	67.44	42.31	63.64
Deep Research (pass@1)	74.29	69.06	47.6	67.36
Deep Research (cons@64)	**78.66**	**73.21**	**58.03**	**72.57**

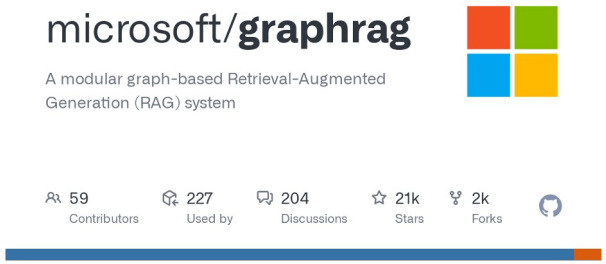


May 4th 2024 - OpenAI GPT4 got 15%
Feb 2nd 2025 - OpenAI Deep Research got 72.6%!
March 6th 2025 - Manus AI has reportedly achieved SOTA performance (no average reported).

LLM Tools

LLM Tools

RAG Frameworks

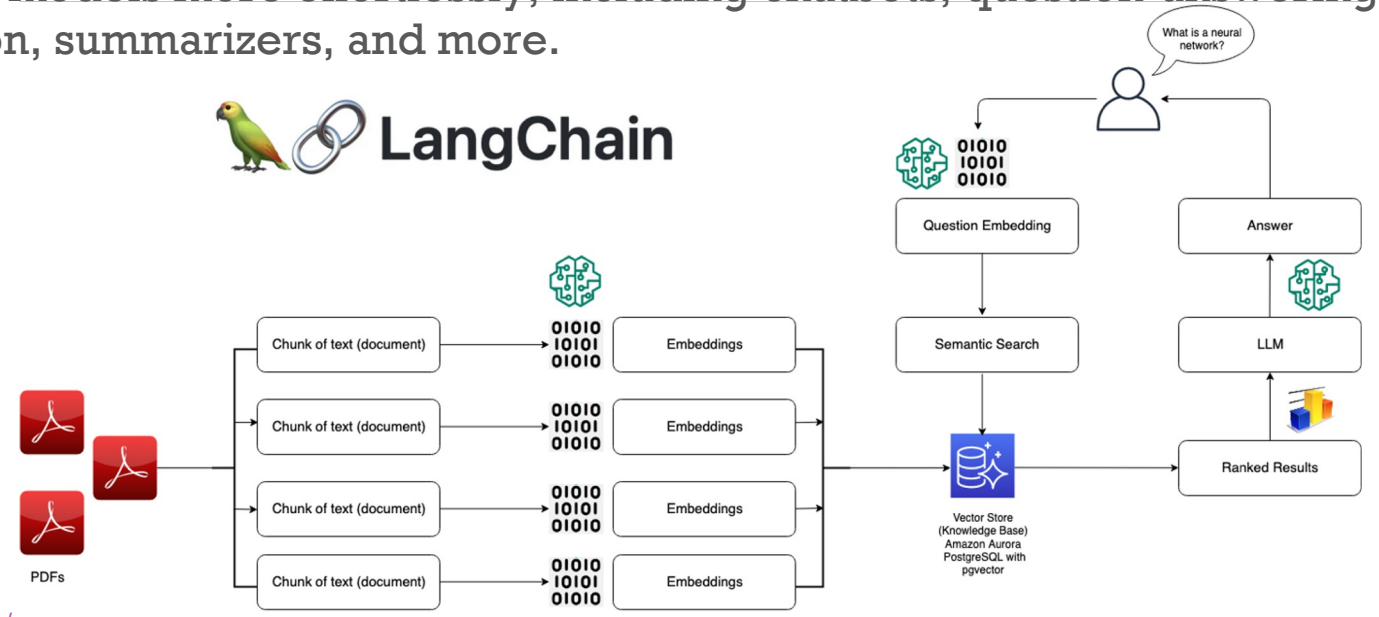


Agentic Frameworks

1) LangChain

LangChain provides tools and abstractions to improve the customization, accuracy, and relevancy of the information the models generate.

LangChain streamlines intermediate steps to develop data-responsive applications, making prompt engineering more efficient. It is designed to develop diverse applications powered by language models more effortlessly, including chatbots, question-answering, content generation, summarizers, and more.



https://www.langchain.com/

1) LangChain (cont.)

LangChain has 100+ integrations.
See full list from the link below.

https://python.langchain.com/docs/integrations/providers/all/

LangChain integrates with many providers.

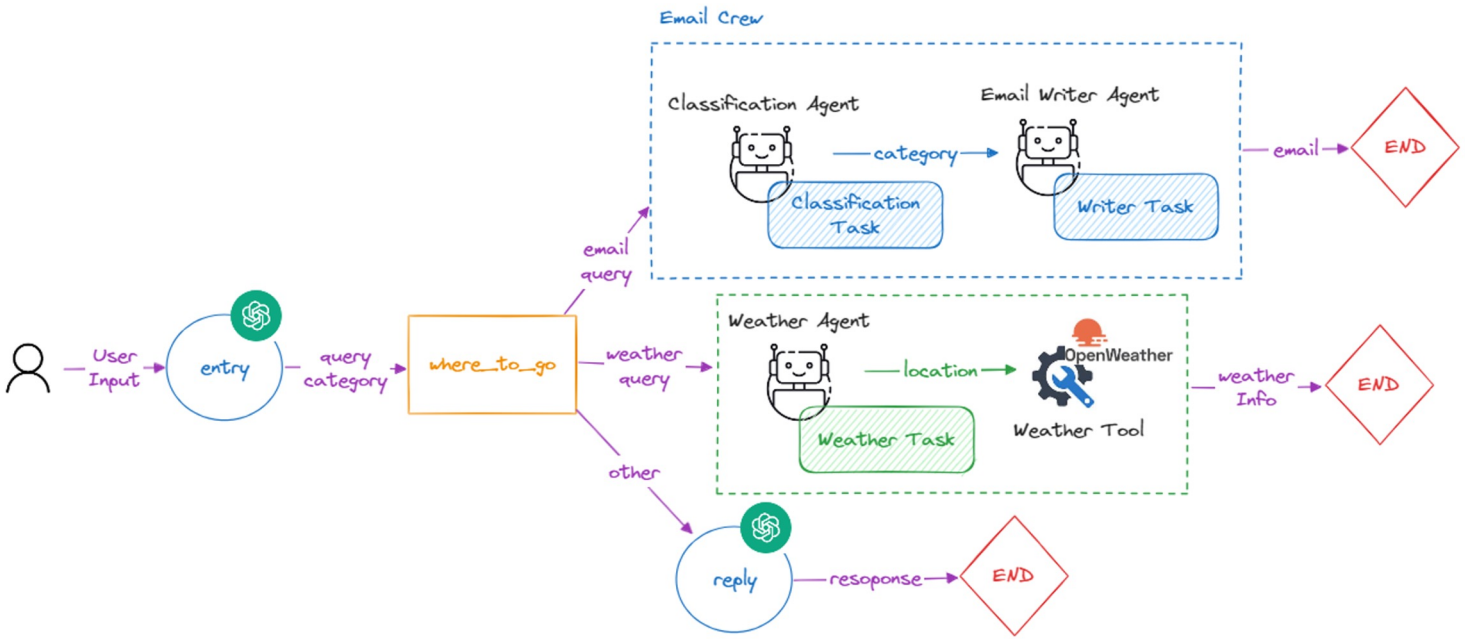
Integration Packages

These providers have standalone `langchain-{provider}` packages for improved versioning, dependency testing.

Provider	Package	Downloads	Latest	JS
Google VertexAI	langchain-google-vertexai	15M/month	v2.0.15	✅
OpenAI	langchain-openai	12M/month	v0.3.8	✅
Google Community	langchain-google-community	4.3M/month	v2.0.7	❌
AWS	langchain-aws	2M/month	v0.2.15	✅
Anthropic	langchain-anthropic	1.9M/month	v0.3.10	✅
Google Generative AI	langchain-google-genai	1.3M/month	v2.1.0	✅
Ollama	langchain-ollama	844k/month	v0.2.3	✅

2) LangGraph

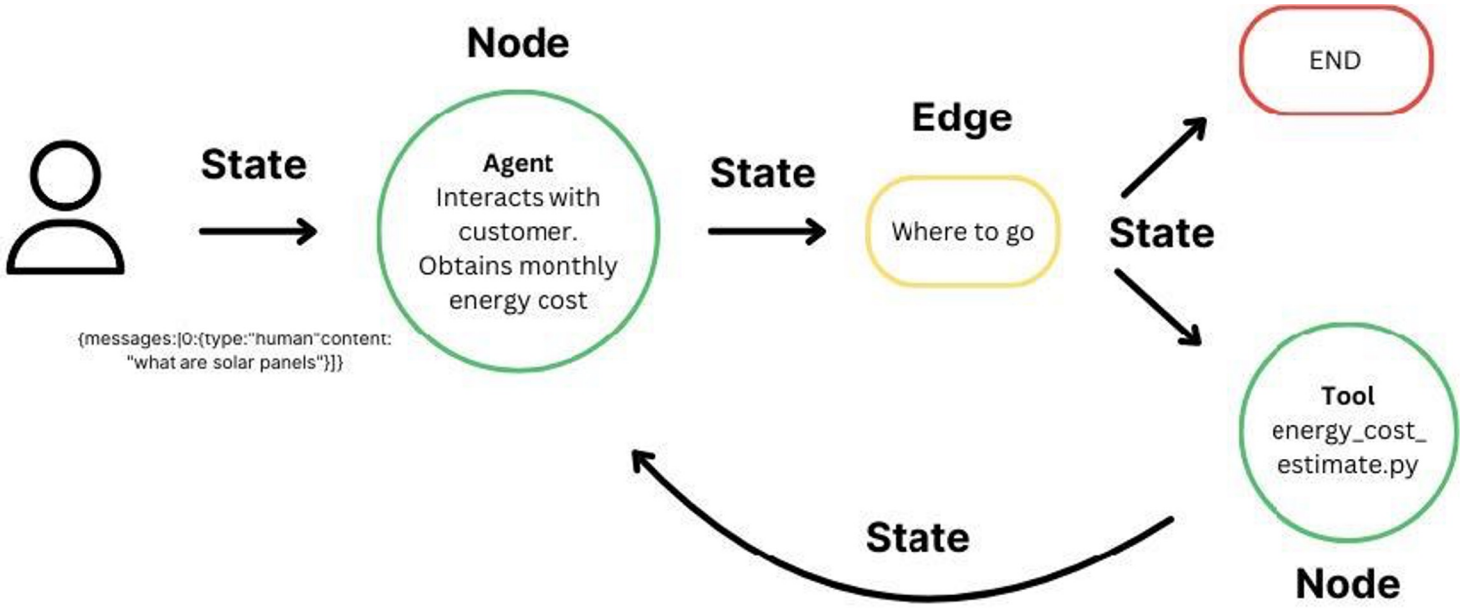
LangGraph is an orchestration framework for complex agentic systems and is more low-level and controllable than LangChain agents.



https://www.langchain.com/langgraph

2) LangGraph (cont.)

LangGraph is a framework designed for graph-based applications.



https://www.langchain.com/langgraph

3) LangSmith

Langsmith is designed with AI model debugging and orchestration in mind.

Observability

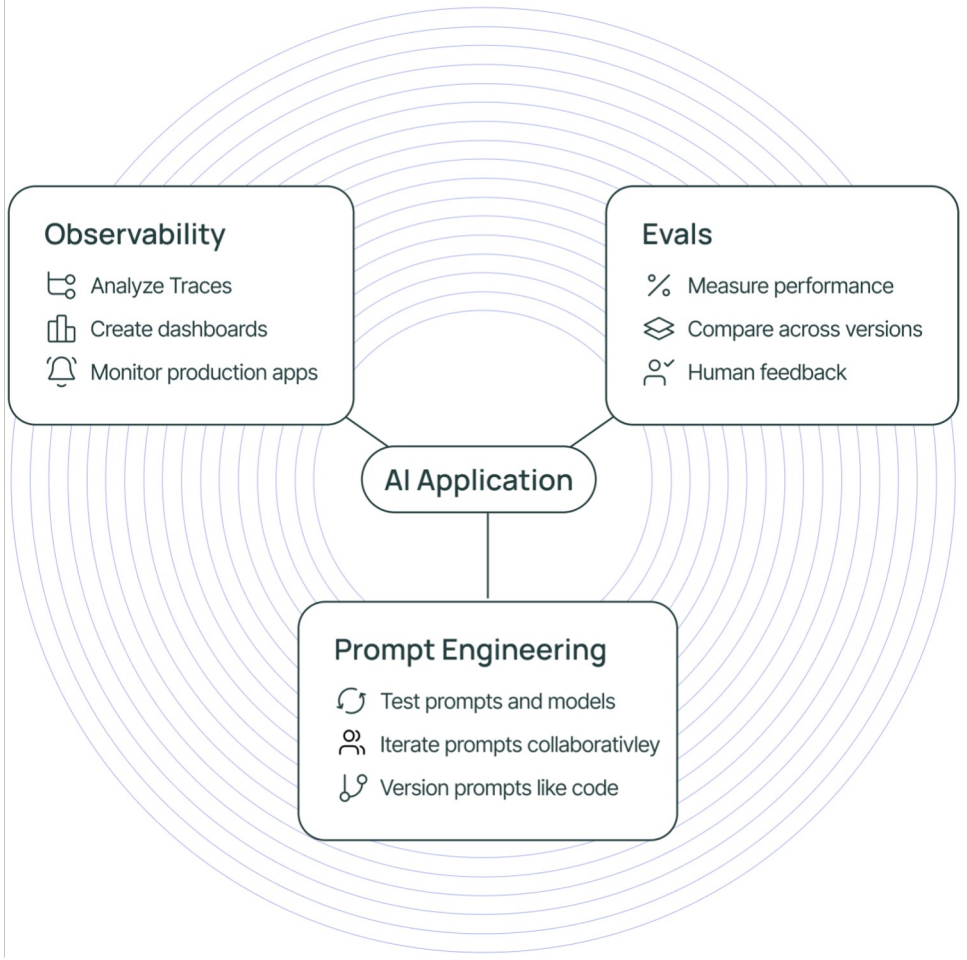
Analyze traces in LangSmith and configure metrics, dashboards, alerts based on these.

Evals

Evaluate your application over production traffic — score application performance and get human feedback on your data.

Prompt Engineering

Iterate on prompts, with automatic version control and collaboration features.



https://www.langchain.com/langsmith

Tutorial

https://python.langchain.com/docs/tutorials/

If you're looking to get started with chat models, vector stores, or other LangChain components from a specific provider, check out our supported integrations.

- Chat models and prompts: Build a simple LLM application with prompt templates and chat models.
- Semantic search: Build a semantic search engine over a PDF with document loaders, embedding models, and vector stores.
- Classification: Classify text into categories or labels using chat models with structured outputs.
- Extraction: Extract structured data from text and other unstructured media using chat models and few-shot examples.

Get started using LangGraph to assemble LangChain components into full-featured applications.

- Chatbots: Build a chatbot that incorporates memory.
- Agents: Build an agent that interacts with external tools.
- Retrieval Augmented Generation (RAG) Part 1: Build an application that uses your own documents to inform its responses.
- Retrieval Augmented Generation (RAG) Part 2: Build a RAG application that incorporates a memory of its user interactions and multi-step retrieval.
- Question-Answering with SQL: Build a question-answering system that executes SQL queries to inform its responses.
- Summarization: Generate summaries of (potentially long) texts.
- Question-Answering with Graph Databases: Build a question-answering system that queries a graph database to inform its responses.