



# Improving Generative LLMs: Cognitive Architectures and RAG

DL4DS – Spring 2024

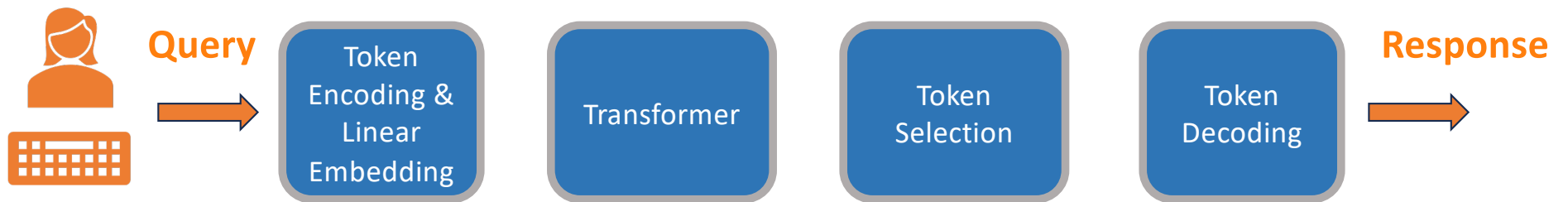
# Topics

- Generative LLM flow and how to evaluate
- Improve LLM performance by prompting strategies
- Improving with retrieval augmentation
- Building more complex systems with LLMs: "Cognitive Architectures"

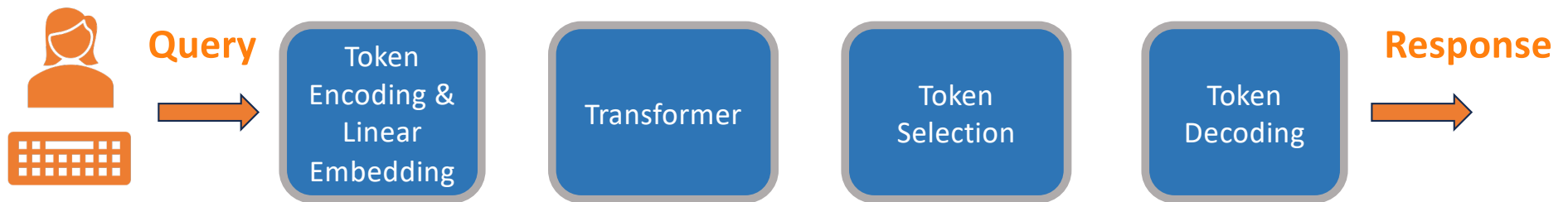
# Topics

- Generative LLM flow and how to evaluate
- Improve LLM performance by prompting strategies
- Improving with retrieval augmentation
- Building more complex systems with LLMs: "Cognitive Architectures"

# LLM Generative Flow



# LLM Generative Flow



- How do we evaluate the response?
- How do we improve the response?

# Generative LLM Evaluations

Evaluate for

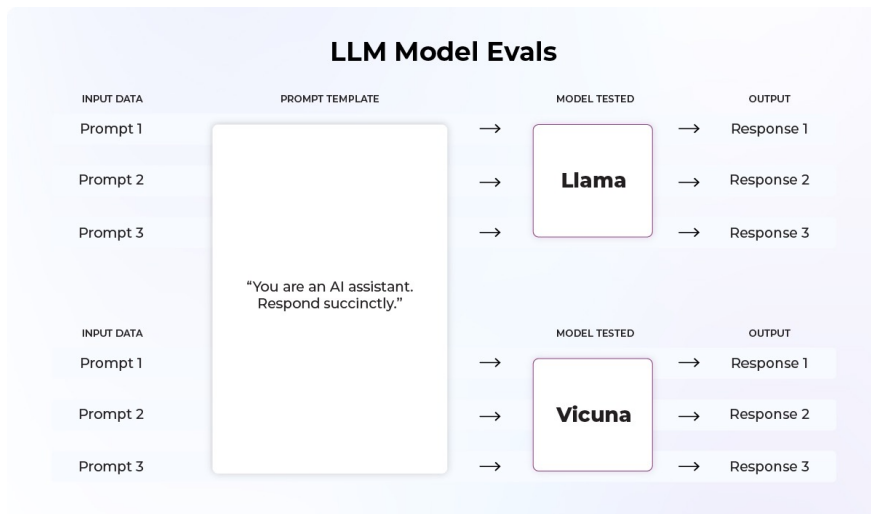
- Accuracy (is it factual or hallucinated?)
- Relevance (is it answering the question?)
- Bias, Toxicity (Is it fair? Or even worse is it racist or toxic?)
- Diversity of Response (does it always give same response? or equally useful diverse responses?)

# Ways to Evaluate

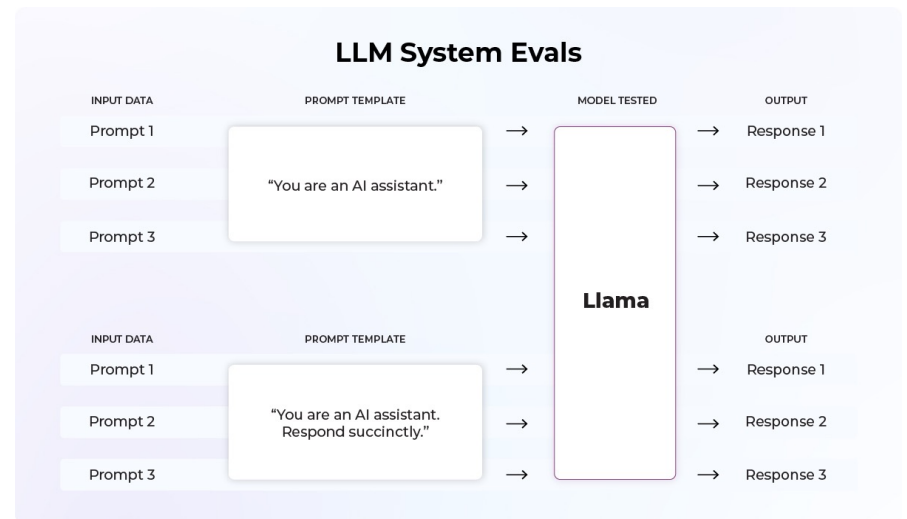
- Find a benchmark that matches your task
  - [HellaSwag](#) (*which evaluates how well an LLM can complete a sentence*),
  - [TruthfulQA](#) (*measuring truthfulness of model responses*), and
  - [MMLU](#) (*which measures how well the LLM can multitask*),
  - [WinoGrande](#) (*commonsense reasoning*),
  - [GSM8K](#), (*arithmetic reasoning*), etc.
- Evaluate with a metric, e.g. [BLEU](#), [METEOR](#), [ROUGE](#), [CIDEr](#), [SPICE](#), etc.
  - Pros and Cons of each metric
- Create your own evaluation prompt/response pairs –
  - need thousands!
- Build an LLM to evaluate your LLM!

See: <https://arize.com/blog-course/llm-evaluation-the-definitive-guide/> for a nice overview

# Model vs System Evals



Useful for choosing a model or deciding when to switch.



Useful for prompt tuning and monitoring over time.

See: <https://arize.com/blog-course/llm-evaluation-the-definitive-guide/> for a nice overview



# Open LLM Leaderboard

## 😊 Open LLM Leaderboard

LLM Benchmark Metrics through time About ! FAQ Submit

Search for your model (separate multiple queries with `;` and press ENTER...)

Select columns to show

- Average  ARC  HellaSwag  MMLU  TruthfulQA
- Winogrande  GSM8K  Type  Architecture  Precision
- Merged  Hub License  #Params (B)  Hub  Model sha

Hide models

- Private or deleted  Contains a merge/moerge  Flagged  MoE

Model types

- pretrained  continuously pretrained
- fine-tuned on domain-specific datasets  chat models (RLHF, DPO, IFT, ...)
- base merges and moerges  ?

Precision

- float16  bfloat16  8bit  4bit  GPTQ  ?

Model sizes (in billions of parameters)

- ?  ~1.5  ~3  ~7  ~13  ~35  ~60  70+

T	Model	Average	ARC	HellaSwag	MMLU	TruthfulQA
◆	SF-Foundation/Ein-72B-v0.11	80.81	76.79	89.02	77.2	79.02
◆	SF-Foundation/Ein-72B-v0.13	80.79	76.19	89.44	77.07	77.82
◆	SF-Foundation/Ein-72B-v0.12	80.72	76.19	89.46	77.17	77.78
◆	abacusai/Smaug-72B-v0.1	80.48	76.02	89.27	77.15	76.67
◆	ibivibiv/alpaca-dragon-72b-v1	79.3	73.89	88.16	77.4	72.69
🗨	moxeh/MoMo-72B-lora-1.8.7-DPO	78.55	70.82	85.96	77.13	74.71
◆	cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_DPO_f16	77.91	74.06	86.74	76.65	72.24
◆	saitlux/luxia-21.4b-alignment-v1.0	77.74	77.47	91.88	68.1	79.17
◆	cloudyu/TomGrc_FusionNet_34Bx2_MoE_v0.1_full_linear_DPO	77.52	74.06	86.67	76.69	71.32
◆	zhengx/MixTAO-7Bx2-MoE-v8.1	77.5	73.81	89.22	64.92	78.57
🗨	yunconglong/Truthful_DPO_TomGrc_FusionNet_7Bx2_MoE_13B	77.44	74.91	89.3	64.67	78.02
◆	JaeveonKang/CCK Asura v1	77.43	73.89	89.07	75.44	71.75

[https://huggingface.co/spaces/HuggingFaceH4/open\\_llm\\_leaderboard](https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard)

# Crowd-Sourcing Evaluations User Feedback

## Chatbot Arena: Benchmarking LLMs in the Wild with Elo Ratings

by: Lianmin Zheng\*, Ying Sheng\*, Wei-Lin Chiang, Hao Zhang, Joseph E. Gonzalez, Ion Stoica, May 03, 2023

Expand to see the descriptions of 30 models

Model A

What's the best way to evaluate LLM Q&A systems?

The best way to evaluate Legal Language Model (LLM) Q&A systems is by using a combination of quantitative and qualitative methods. Here are some key evaluation metrics and approaches:

1. **Accuracy:** Measure the correctness of the LLM's answers to specific legal questions. You can use a pre-defined dataset of legal questions and answers or create your own dataset. Compare the LLM's answers to the correct answers and calculate the accuracy.
2. **Precision, Recall, and F1-score:** Assess the LLM's ability to provide relevant information and filter out irrelevant information. Precision measures the percentage of relevant answers among all the answers provided, recall measures the percentage of relevant answers identified, and the F1-score is the harmonic mean of precision and recall.
3. **Answer Confidence:** Evaluate the LLM's confidence in its answers by analyzing the model's probability scores. This can help identify questions where the model may need further training or the answers may need additional verification.
4. **Response Time:** Measure how quickly the LLM can generate answers

Model B

What's the best way to evaluate LLM Q&A systems?

There are a few key considerations when evaluating large language model (LLM) question-answering (Q&A) systems:

1. **Accuracy:** The most important metric is how accurately the system can answer factual questions. This can be tested using benchmark datasets like SQuAD, TriviaQA, or Natural Questions. Evaluating accuracy on a diverse range of question types is important.
2. **Robustness:** Assess how the system handles ambiguous, open-ended, or adversarial questions. Look at things like sensitivity to small wording changes, ability to handle multi-part questions, and resistance to attempts to "trick" the system.
3. **Coherence and Fluency:** Evaluate the quality and naturalness of the system's language output. Does it provide responses that are grammatically correct, logically coherent, and easy for a human to understand?
4. **Knowledge Coverage:** Examine the breadth and depth of the system's knowledge. Can it answer questions across a wide range of topics, or is its knowledge more narrow or specialized?
5. **Reasoning Ability:** Assess the system's capacity for logical reasoning,

A is better    B is better    Tie    Both are bad

Enter your prompt and press ENTER    Send

### LMSYS Chatbot Arena Leaderboard

[Vote](#) | [Blog](#) | [GitHub](#) | [Paper](#) | [Dataset](#) | [Twitter](#) | [Discord](#)

LMSYS Chatbot Arena is a crowdsourced open platform for LLM evals. We've collected over 400,000 human preference votes to rank LLMs with the Elo ranking system.

Arena Elo    Full Leaderboard

Total #models: 73. Total #votes: 408144. Last updated: March 13, 2024.

Contribute your vote at [chat.lmsys.org](#). Find more analysis in the [notebook](#).

Rank	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	<a href="#">GPT-4-1106-preview</a>	1251	+5/-4	48226	OpenAI	Proprietary	2023/4
1	<a href="#">GPT-4-0125-preview</a>	1249	+5/-6	22282	OpenAI	Proprietary	2023/12
1	<a href="#">Claude-3-Opus</a>	1247	+6/-6	14854	Anthropic	Proprietary	2023/8
4	<a href="#">Bard-(Gemini-Pro)</a>	1202	+6/-7	12623	Google	Proprietary	Online
4	<a href="#">Claude-3-Sonnet</a>	1190	+6/-6	14845	Anthropic	Proprietary	2023/8
5	<a href="#">GPT-4-0314</a>	1185	+4/-6	27245	OpenAI	Proprietary	2021/9
7	<a href="#">GPT-4-0613</a>	1159	+4/-5	43783	OpenAI	Proprietary	2021/9
7	<a href="#">Mistral-Large-2402</a>	1155	+5/-6	18959	Mistral	Proprietary	Unknown
8	<a href="#">Qwen1.5-72B-Chat</a>	1146	+4/-5	16729	Alibaba	Qianwen LICENSE	2024/2
8	<a href="#">Claude-1</a>	1145	+5/-6	21929	Anthropic	Proprietary	Unknown
8	<a href="#">Mistral-Medium</a>	1145	+5/-4	23931	Mistral	Proprietary	Unknown
12	<a href="#">Claude-2.0</a>	1126	+7/-5	13679	Anthropic	Proprietary	Unknown
12	<a href="#">Mistral-Next</a>	1123	+5/-5	11875	Mistral	Proprietary	Unknown
12	<a href="#">Gemini-Pro-(Dev-API)</a>	1118	+5/-6	12146	Google	Proprietary	2023/4
13	<a href="#">Claude-2.1</a>	1115	+4/-6	32431	Anthropic	Proprietary	Unknown

<https://lmsys.org/>

<https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard>

# Topics

- Generative LLM flow and how to evaluate
- **Improve LLM performance by prompting strategies**
- Improving with retrieval augmentation
- Building more complex systems with LLMs: "Cognitive Architectures"

# Ask for reflection

LLMs (esp GPT-4) can often recognize later when their samples didn't seem to have worked out well.

can you write a poem that does not rhyme? think carefully about the assignment

In the vast expanse of a star-strewn sky,  
A lone wanderer's heart begins to sigh,  
Drifting through the cosmos, bound by no line,  
A celestial dance, a journey, divine.

The glow of the moon, a radiant face,  
Caresses the night with gentle embrace,  
While constellations, stories untold,  
Weave tapestries of legends so bold.

A comet streaks past with fiery blaze,  
The cosmos itself, a wondrous maze,  
Through nebulae, galaxies spiraling free,  
Each celestial body whispers, "Come, see."

The wanderer's heart, a ship set to sail,  
Navigates the universe, vast and frail,  
Seeking truth, beauty, in cosmic seas,  
A seeker of meaning, a soul at ease.

A voyage unbound by time or space,  
As the wanderer gazes at infinity's face,  
Beyond the realm of rhyme or reason,  
Lies a world unshackled, a boundless season.

did the poem meet the assignment?

Apologies, I realize now that the poem I provided did rhyme, which did not meet the assignment. Here's a non-rhyming poem for you:

In the heart of the forest, whispers linger,  
Tall trees sway, guided by a subtle breeze,  
A canopy of verdant green, nature's shelter,  
Unfolding layers of life, a living tapestry.

Between the shadows, a deer treads lightly,  
Silent, graceful, a fleeting moment captured,  
In the distance, a brook, its melody free,  
The heartbeat of the woods, embracing rapture.

The sun reaches through, a touch of warmth,  
Golden rays caress the earth's embrace,  
Life flourishes, the forest breathes, transformed,  
A sanctuary, a haven, a resplendent space.

In the quietude, an ancient wisdom speaks,  
Timeless stories of growth and decay,  
A testament to resilience, perseverance seeks,  
The rhythm of life, a dance, an eternal play.

<https://karpathy.ai/stateofgpt.pdf>

[evjang.com/2023/03/26/self-reflection.html]

# Condition on good performance

LLMs don't want to succeed. They want to imitate training sets with a spectrum of performance qualities. You want to succeed, and you should ask for it.

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	APE	Let's work this out in a step by step way to be sure we have the right answer.	<b>82.0</b>
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

## Related examples:

"You are a leading expert on this topic"

"Pretend you have IQ 120"

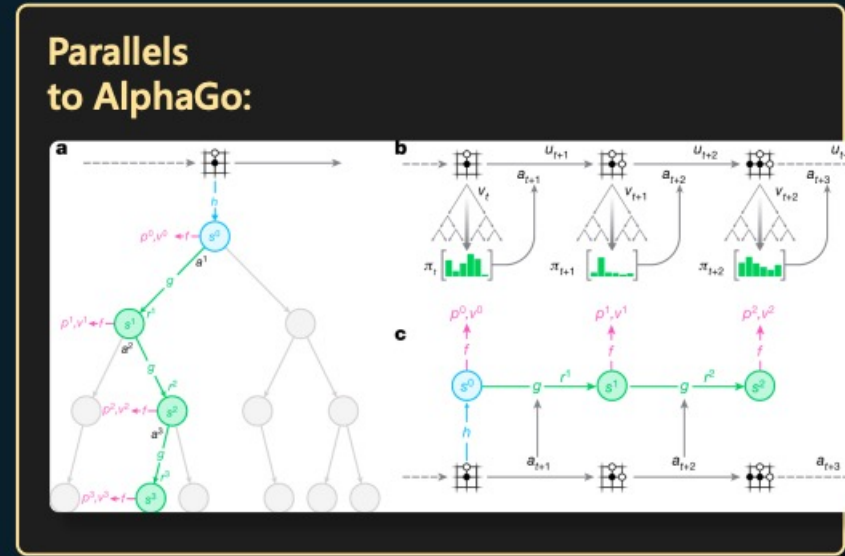
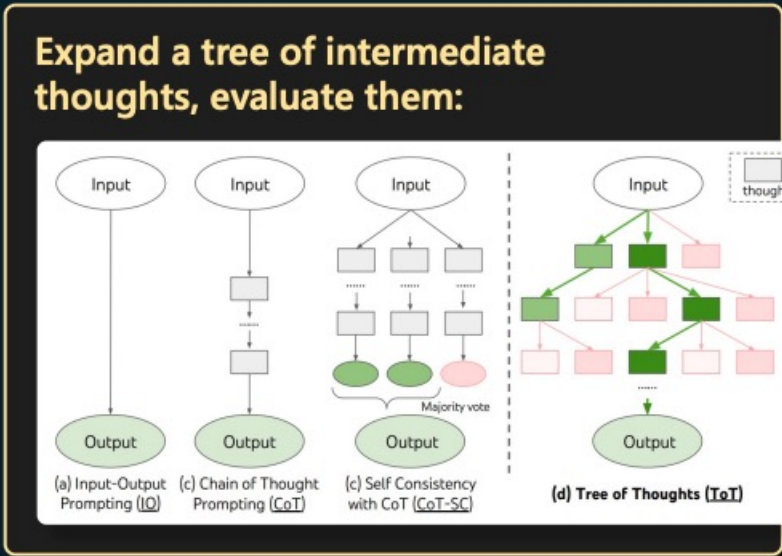
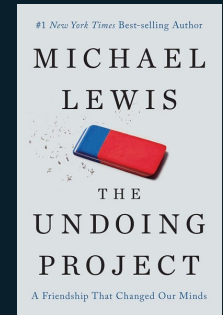
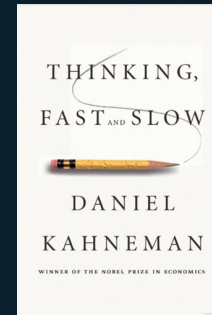
...

[Large Language Models Are Human-Level Prompt Engineers, 2023]

<https://karpathy.ai/stateofgpt.pdf>

# Recreate our 'System 2'

Parallels to System 1 (fast, automatic) vs. System 2 (slow, deliberate) modes of thinking



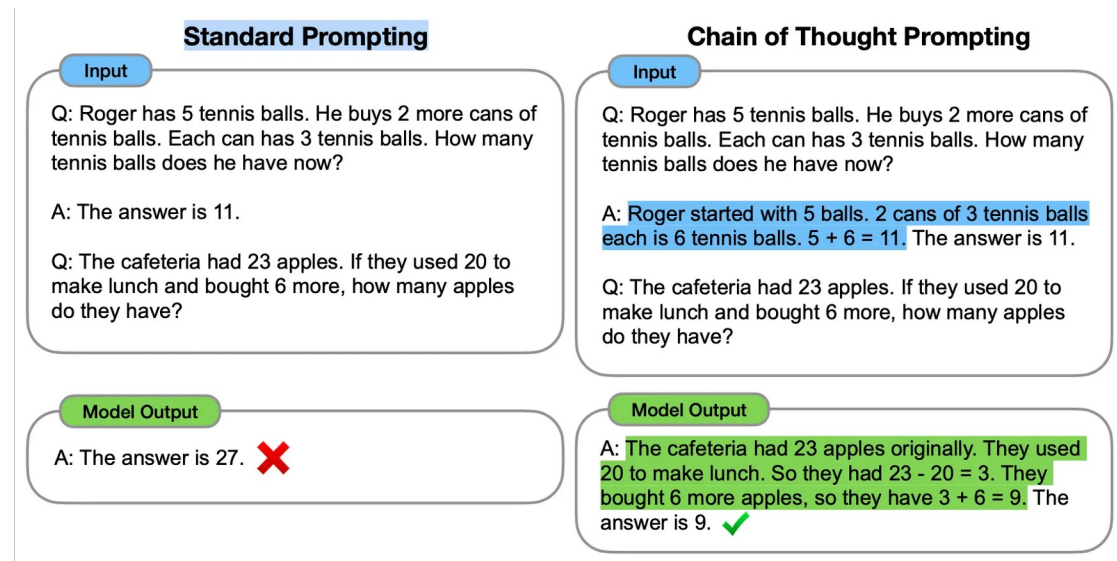
[Mastering the game of go without human knowledge, Silver et al. 2017 (AlphaGo)]  
 [Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao et al. 2013]

<https://karpathy.ai/stateofgpt.pdf>

# Chain of Thought Prompting (few shot)

CoT enhances the reasoning abilities of LLMs by generating intermediate reasoning steps before arriving at a final answer

- Enables models to break down complex, multi-step problems into clearer, intermediate steps
- Offers a clear view into the model's thought process, highlighting areas for debugging and improvement.
- Suitable for a broad range of reasoning tasks including math, commonsense, and symbolic reasoning
- Easily activated in LLMs with simple prompts, leveraging existing capabilities without customization.



J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." arXiv, Jan. 10, 2023. doi: [10.48550/arXiv.2201.11903](https://doi.org/10.48550/arXiv.2201.11903).

# Chain of Thought Prompting (zero shot)

- LLMs are decent zero-shot reasoners by simply adding “Let’s think step by step” before each answer

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let’s think step by step.**

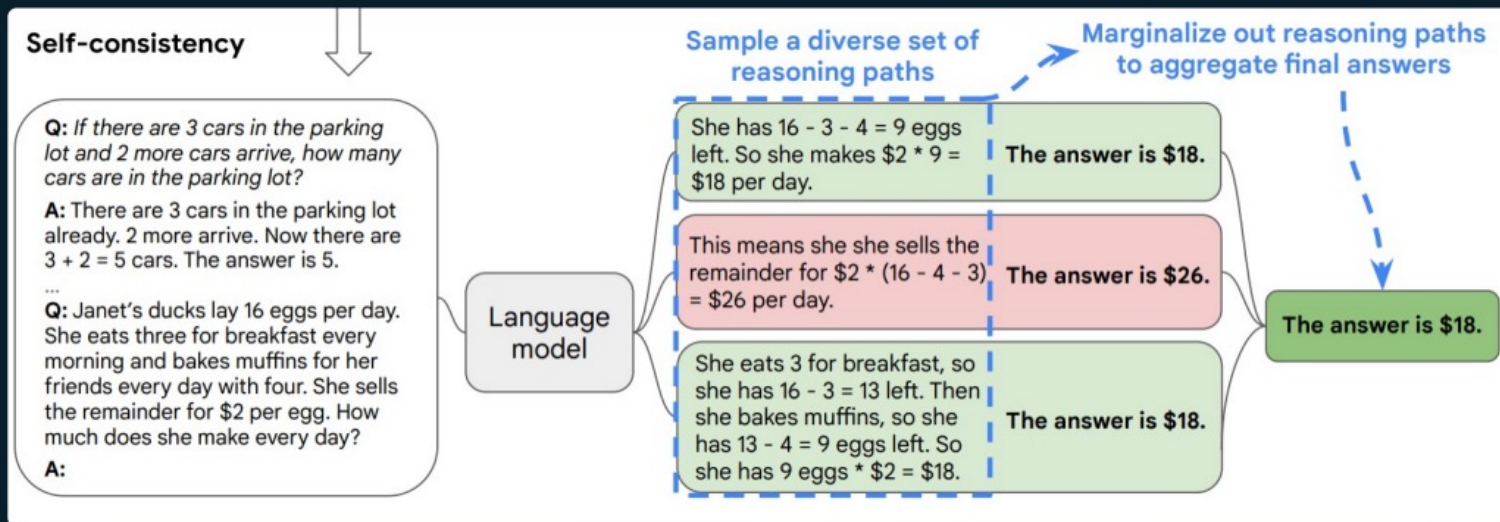
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

T. Kojima, S. (Shane) Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large Language Models are Zero-Shot Reasoners,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 22199–22213, Dec. 2022. ([link](#))



# Ensemble multiple attempts

LLMs can get "unlucky" and sample a bad thought. Once they do they are "stuck with it". Make a few attempts.

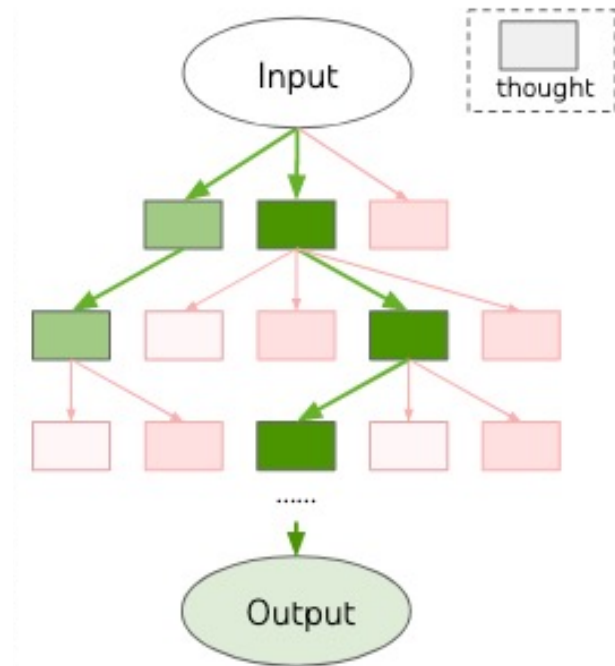


[Self-Consistency Improves Chain of Thought Reasoning in Language Models, Wang et al. 2023]

<https://karpathy.ai/stateofgpt.pdf>

# Tree of Thought

- ToT allows LMs to perform deliberate decision making by considering multiple different reasoning paths and self-evaluating choices to decide the next course of action, as well as looking ahead or backtracking when necessary to make global choices.
- Our experiments show that ToT significantly enhances language models' problem-solving abilities on three novel tasks requiring non-trivial planning or search



These techniques can improve LLM response generation, but how do we tailor to specific knowledge bases?

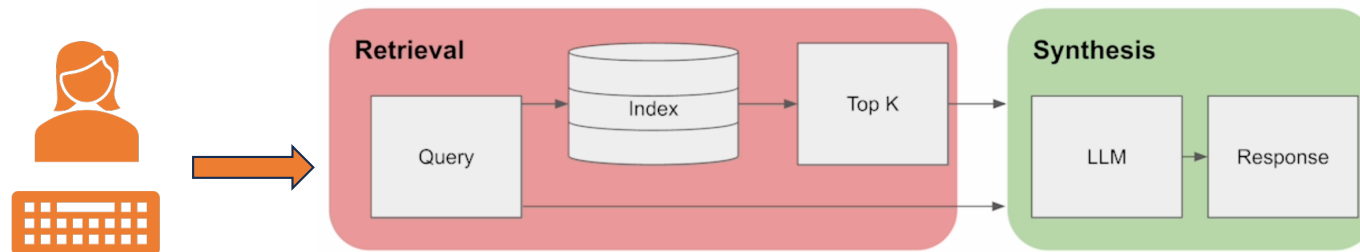
# Topics

- Generative LLM flow and how to evaluate
- Improve LLM performance by prompting strategies
- **Improving with retrieval augmentation**
- Building more complex systems with LLMs: "Cognitive Architectures"

# Retrieval-Augmented Generation (RAG)

RAG enhances LLMs by referencing external knowledge to generate relevant responses.

- Integrates external data into LLM text generation.
- Reduces hallucination, improves response relevance.
- Works with
  - Unstructured data (e.g. documents)
  - Structured data (e.g. SQL data)
  - Code (e.g. python)



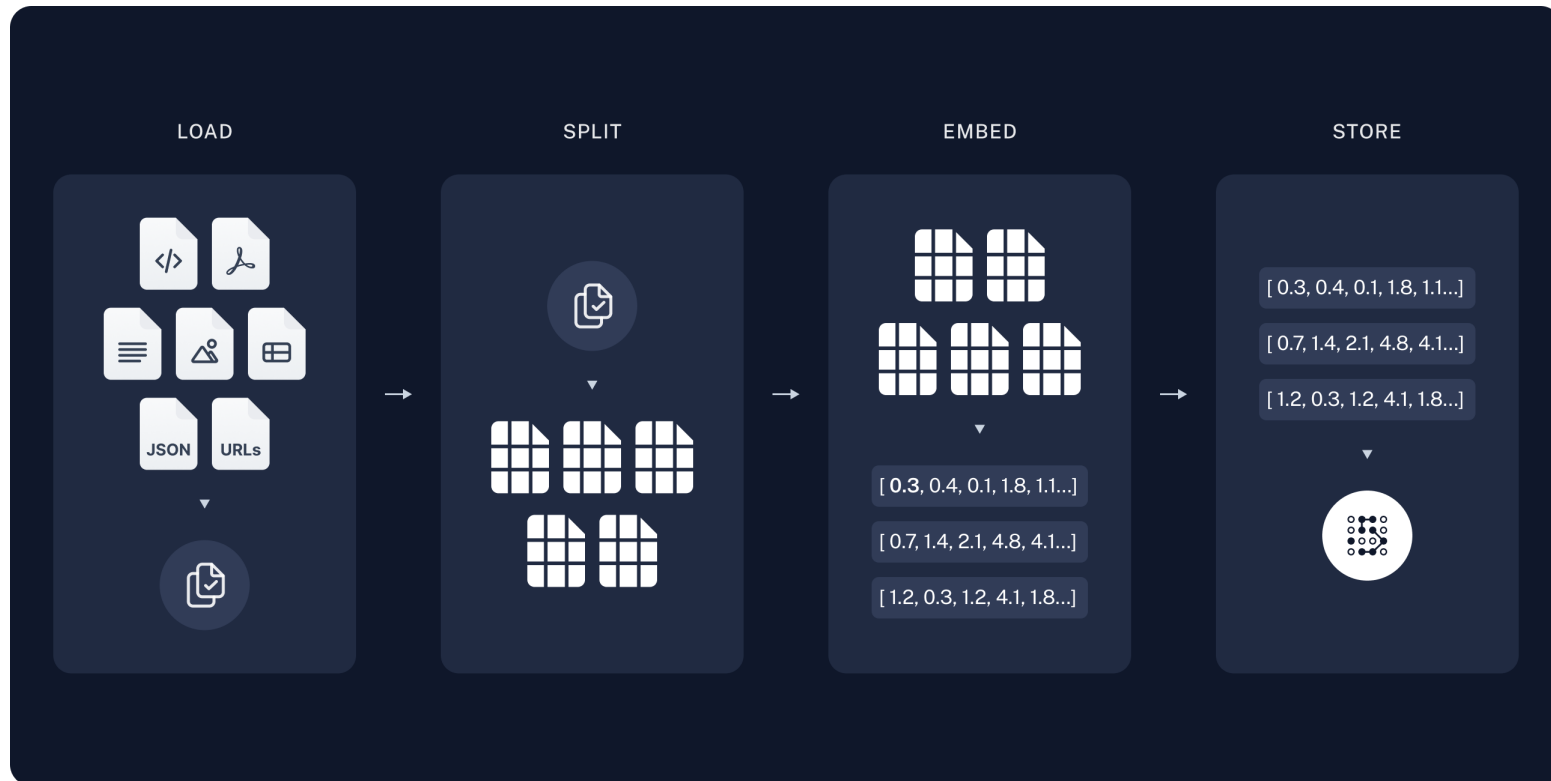
# RAG Architecture

Typical RAG application has two main components:

- Loading and Indexing:
  - A pipeline for ingesting data from a source and indexing it
  - Usually happens offline
- Retrieval and Generation:
  - Takes user query at run time and retrieves relevant data from the index and passes it to the model

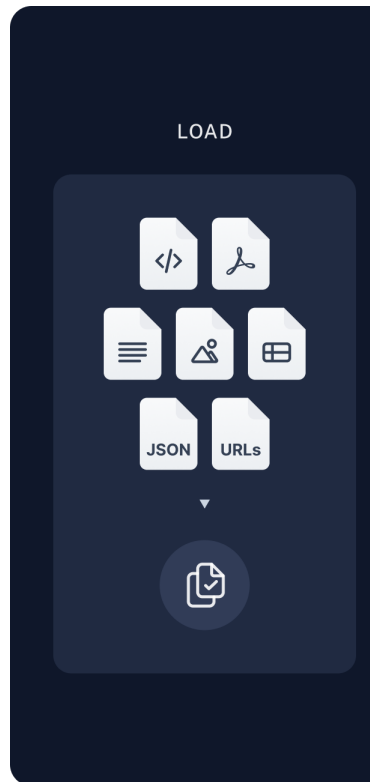
[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)

# RAG – Loading and Indexing



[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)

# RAG – Load



Load the data, e.g.

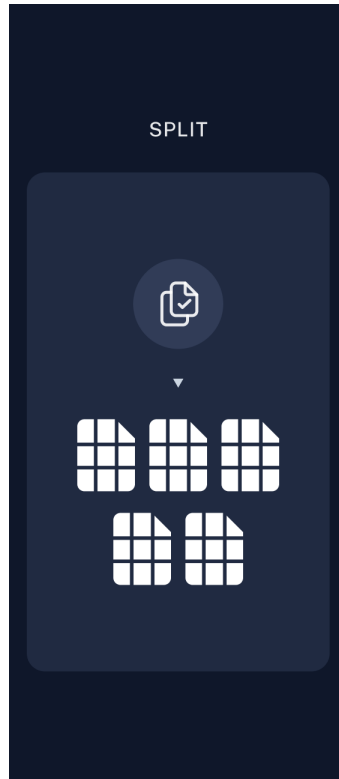
- PDFs
- HTML
- Plain text
- Images, video, audio
- Structured data (SQL, CSV/TSV, ...)
- JSON
- URLs
- ...

See for example: [https://python.langchain.com/docs/modules/data\\_connection/document\\_loaders/](https://python.langchain.com/docs/modules/data_connection/document_loaders/)

[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)



# RAG – Split



Break large documents into smaller chunks.

Easier to:

- index
- pass to model
- search
- fit into model's context window

See for example: [https://python.langchain.com/docs/modules/data\\_connection/document\\_transformers/](https://python.langchain.com/docs/modules/data_connection/document_transformers/)

[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)

# RAG – Embed

- **Encode** (e.g. with Byte Pair Encoding) and
- **Transform** to embedding vectors with the learned embedding model.



See for example: [https://python.langchain.com/docs/modules/data\\_connection/text\\_embedding/](https://python.langchain.com/docs/modules/data_connection/text_embedding/)

[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)

# RAG – Store

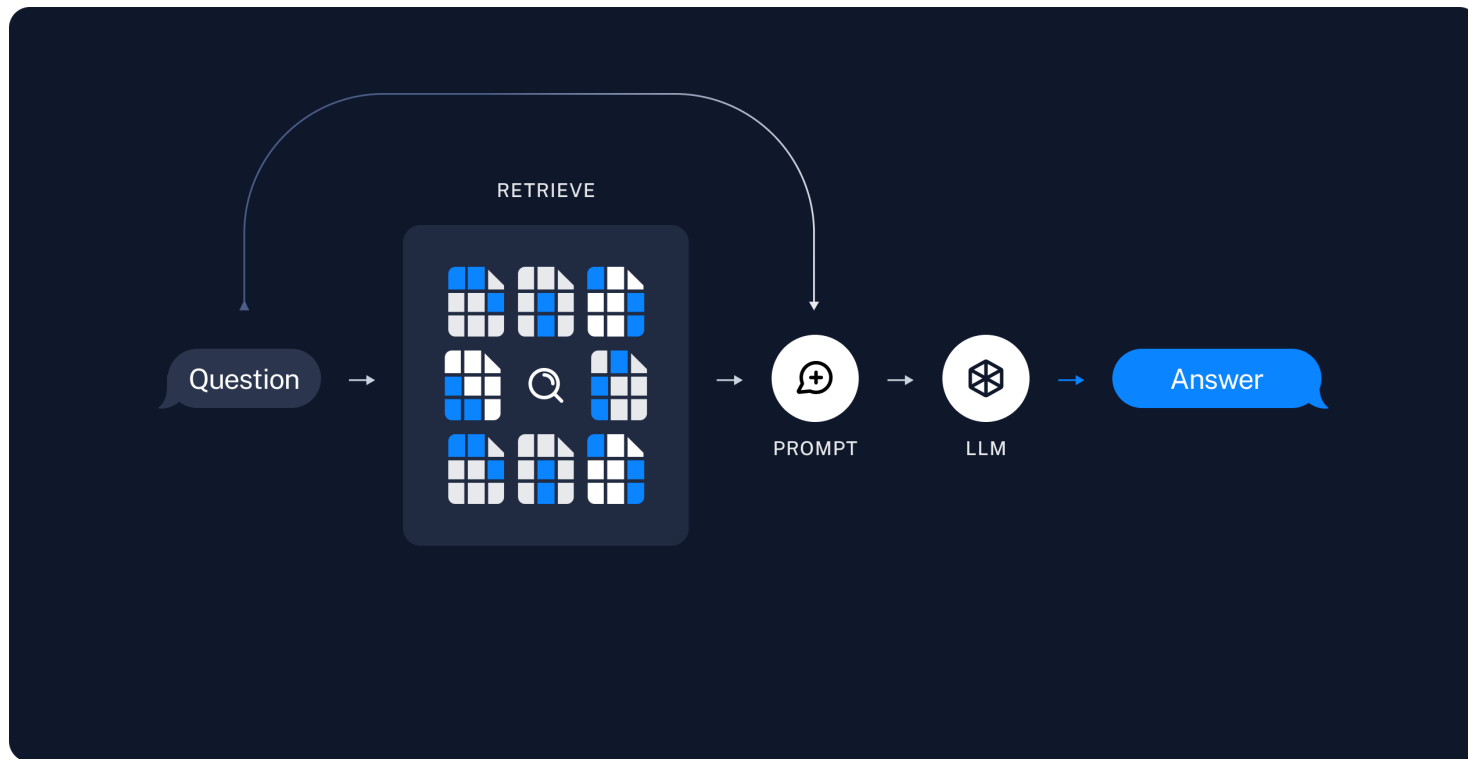
- Store the data in some kind of Vector Store
- e.g. Chroma, FAISS, Lance, Pinecone, etc...



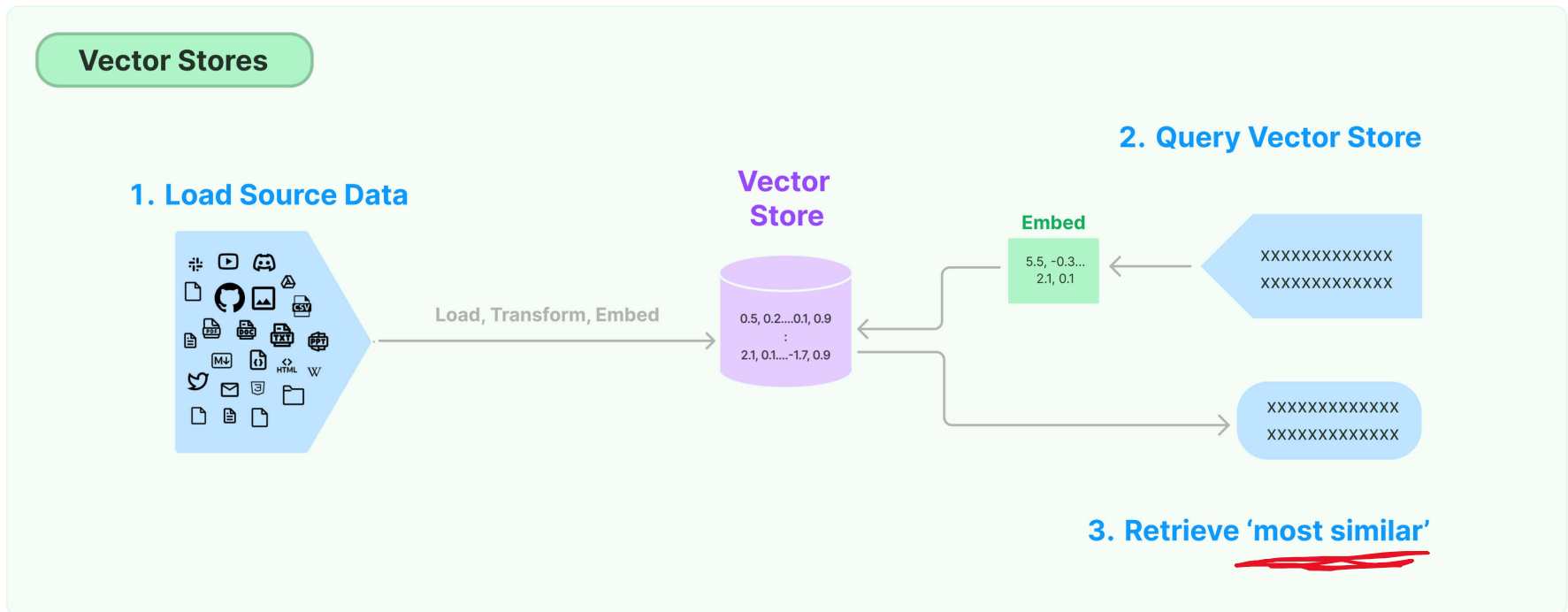
See for example: [https://python.langchain.com/docs/modules/data\\_connection/vectorstores/](https://python.langchain.com/docs/modules/data_connection/vectorstores/)

[https://python.langchain.com/docs/use\\_cases/question\\_answering/](https://python.langchain.com/docs/use_cases/question_answering/)

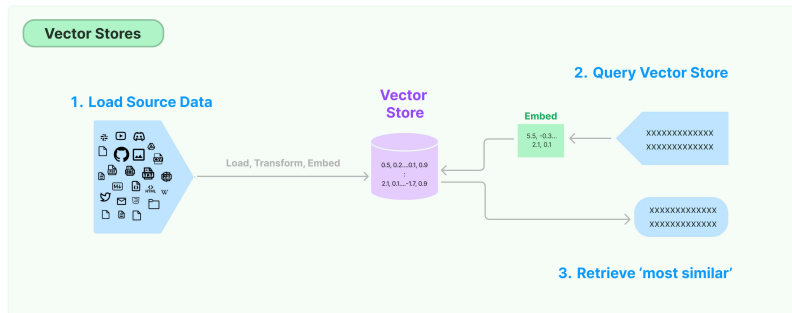
# RAG – Retrieval and Generation



# RAG – Retrieval



# RAG – Retrieval Similarity Measure



$$\text{L2 Norm}^*: d = \sum_i (A_i - B_i)^2$$

$$\text{Inner Product: } d = 1 - \sum_i (A_i \times B_i)$$

$$\text{Cosine Similarity: } 1 - \frac{\sum_i (A_i \times B_i)}{\sqrt{\sum_i (A_i^2)} \sqrt{\sum_i (B_i^2)}}$$

\* Default on Chroma Vector Database

<https://docs.trychroma.com/usage-guide#changing-the-distance-function>

Is simple similarity measure  
between query and document  
the best approach?

# RAG – Other Query-Document Matching Approaches

## 1. BERT and Variants for Query-Document Matching

### **BERT:**

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. *This foundational paper introduces BERT and its methodology for language understanding, which has been widely applied to information retrieval tasks.*

### **Application in Information Retrieval:**

Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv:1901.04085. *This work explores how BERT can be used for re-ranking search results, demonstrating its effectiveness in improving information retrieval systems.* <https://arxiv.org/abs/1901.04085>

## 2. Fine-tuning for Specific Tasks

### **Fine-Tuning BERT for Search:**

MacAvaney, S., Cohan, A., & Goharian, N. (2019). CEDR: Contextualized Embeddings for Document Ranking. SIGIR. *This paper discusses fine-tuning BERT with contextual embeddings specifically for document ranking, providing insights into adapting Transformer models for search tasks.* <https://dl.acm.org/doi/abs/10.1145/3331184.3331317>

## 3. Dual-encoder and Cross-encoder Architectures

### **Dual-Encoders for Efficient Retrieval:**

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP. *This paper introduces a method using dense vector representations for passages and questions to improve open-domain question answering.* <https://arxiv.org/abs/2004.04906>

### **Cross-Encoders for Detailed Similarity Scoring:**

Humeau, S., Shuster, K., Lachaux, M. A., & Weston, J. (2019). Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. arXiv:1905.01969. *The poly-encoder architecture introduced here incorporates aspects of both dual and cross-encoders, offering a balance between speed and accuracy for matching tasks.* <https://arxiv.org/abs/1905.01969>

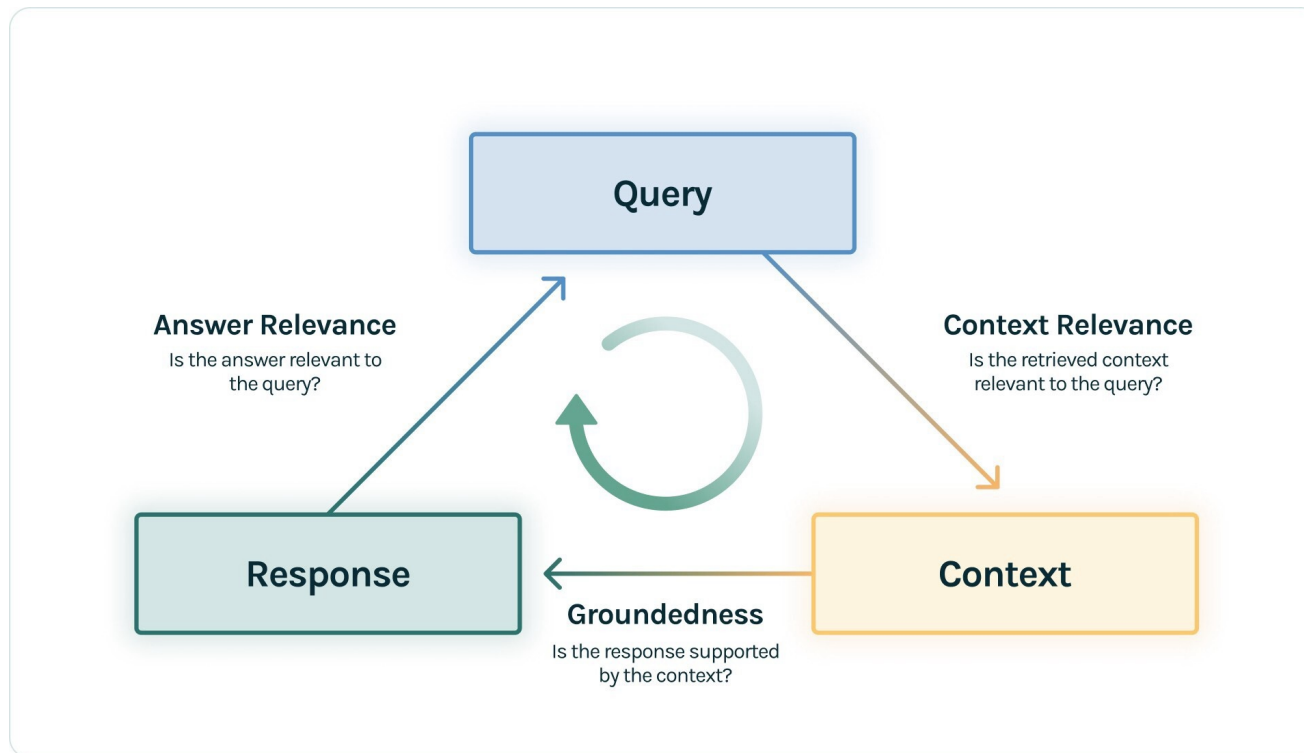
## 4. Semantic Search Systems

### **Semantic Search with Transformers:**

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., & Cheng, X. (2020). A Deep Look into Neural Ranking Models for Information Retrieval. Information Processing & Management. *This review covers deep learning approaches to information retrieval, including the use of Transformer models for understanding query intent and document relevance in a semantic search context.* <https://www.sciencedirect.com/science/article/pii/S0306457319302390>



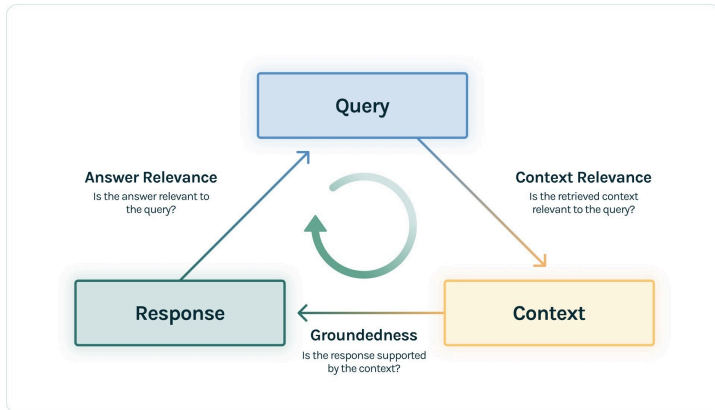
# Evaluating RAG-based LLMs



[https://www.trulens.org/trulens\\_eval/getting\\_started/core\\_concepts/rag\\_triad/](https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/)

# Evaluating RAG: Context Relevance

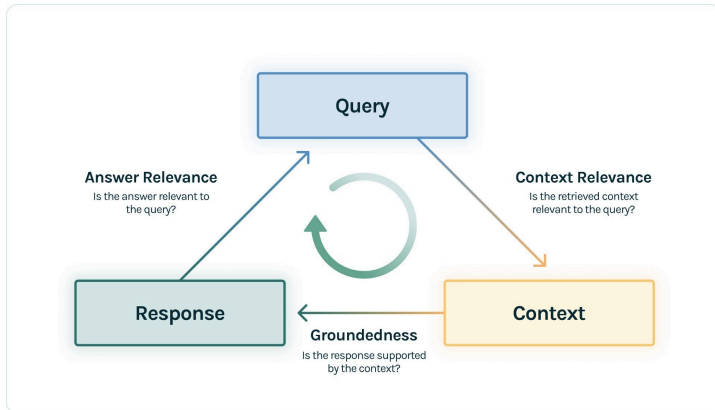
- Is the content retrieved from the vector database relevant to the query?
- Irrelevant information will be likely integrated into the response, contributing to hallucinations



[https://www.trulens.org/trulens\\_eval/getting\\_started/core\\_concepts/rag\\_triad/](https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/)

# Evaluating RAG: Groundedness

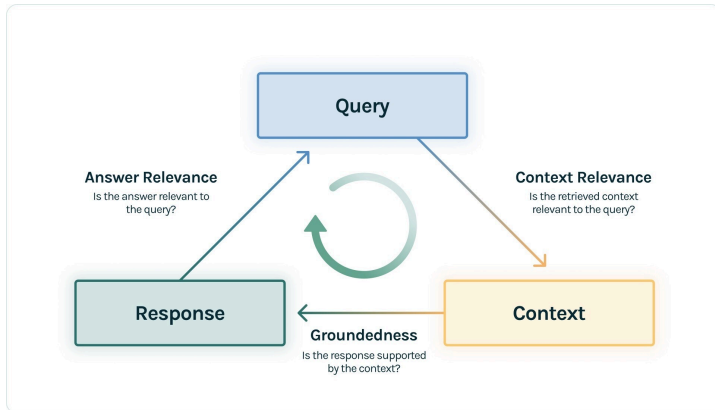
- The context was provided to the LLM as part of the prompt
- Did the LLM response incorporate the context appropriately?
- Can we support each claim in the response from the context?



[https://www.trulens.org/trulens\\_eval/getting\\_started/core\\_concepts/rag\\_triad/](https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/)

# Evaluating RAG: Answer Relevance

- Is the answer relevant to the original question?
- Prompt is augmented with context.
- Did the context cause the LLM to stray away from the question?



[https://www.trulens.org/trulens\\_eval/getting\\_started/core\\_concepts/rag\\_triad/](https://www.trulens.org/trulens_eval/getting_started/core_concepts/rag_triad/)

# Growing ecosystem of tools to do evaluation

```
# in a notebook  
tru.get_leaderboard(app_ids=[])
```

app_id	Groundedness	Answer Relevance	Context Relevance	latency	total_cost
Automerging Query Engine	1.00000	0.940	0.4350	2.25	0.000799
Sentence Window Query Engine	0.87800	0.925	0.3675	2.25	0.000868
Direct Query Engine	0.80125	0.930	0.2550	2.20	0.002911

```
# launches on http://localhost:8501/  
tru.run_dashboard()
```

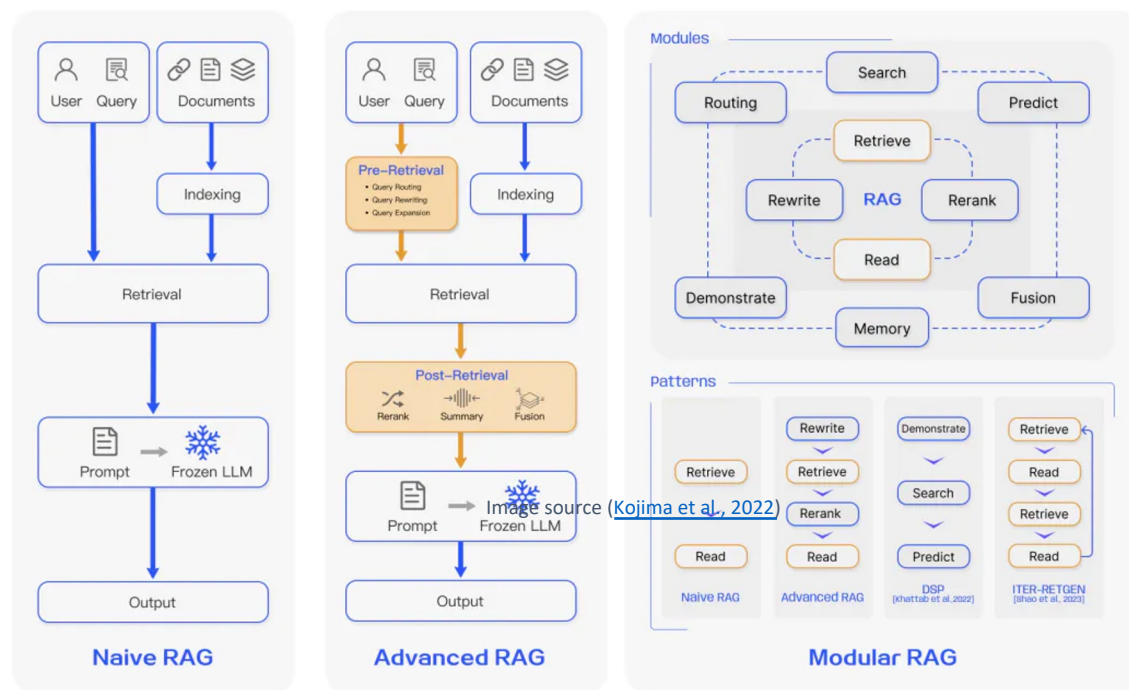


## Evaluate and Track LLM Applications

Evaluate, iterate faster, and select your best LLM app with TruLens.

# Retrieval-Augmented Generation (RAG)

RAG systems have evolved from Naive RAG to Advanced RAG and Modular RAG. This evolution has occurred to address certain limitations around performance, cost, and efficiency.



## Pre-Retrieval Improvements

- Enhance indexed data quality, optimize chunk size and overlap.
- Rewrite user queries for better match in vector database.
- Use metadata and pronoun replacement to maintain context in chunks.

## Retrieval Enhancements

- Explore alternative search methods (e.g., full-text, graph-based).
- Experiment with different embedding models for task suitability.
- Implement hierarchical and recursive search for precision.

## Post-Retrieval Optimization

- Re-rank or score chunks for relevance; compress information from multiple chunks.
- Employ smaller, faster models for specific steps to reduce latency.
- Parallelize intermediate steps and use caching for common queries.

## Balancing Quality and Latency

- Opt for parallel processing, smaller models, and caching strategies.
- Tailor RAG approach based on the complexity of user queries and the nature of tasks.

<https://www.promptingguide.ai/research/rag>

# Topics

- Generative LLM flow and how to evaluate
- Improve LLM performance by prompting strategies
- Improving with retrieval augmentation
- Building more complex systems with LLMs: "Cognitive Architectures"

# Cognitive Architecture\*

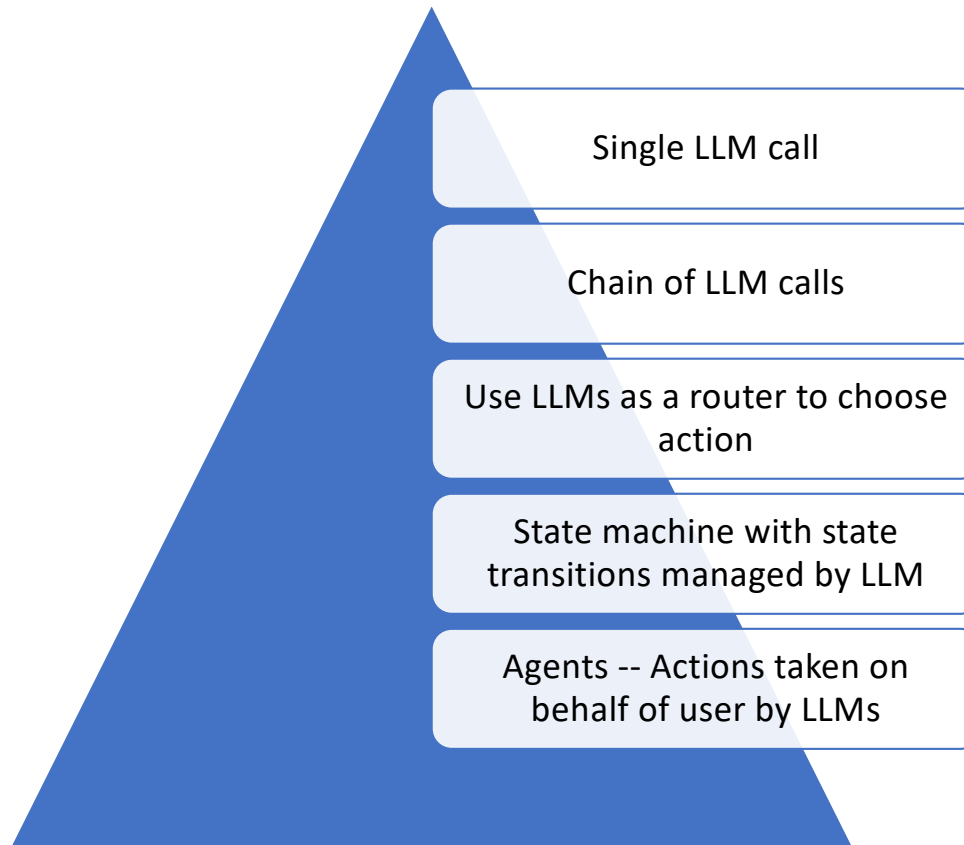
- Orchestration of components of an LLM application
- There are two main components here:
  - (1) how is context provided to the application,
  - (2) how does the application “reason”. Both of these components make up the cognitive architecture of an application.

\*Perhaps coined by [Flo Crivello](#), creator of [Lindy](#) (no code customized AI assistants).

<https://blog.langchain.dev/openais-bet-on-a-cognitive-architecture/>

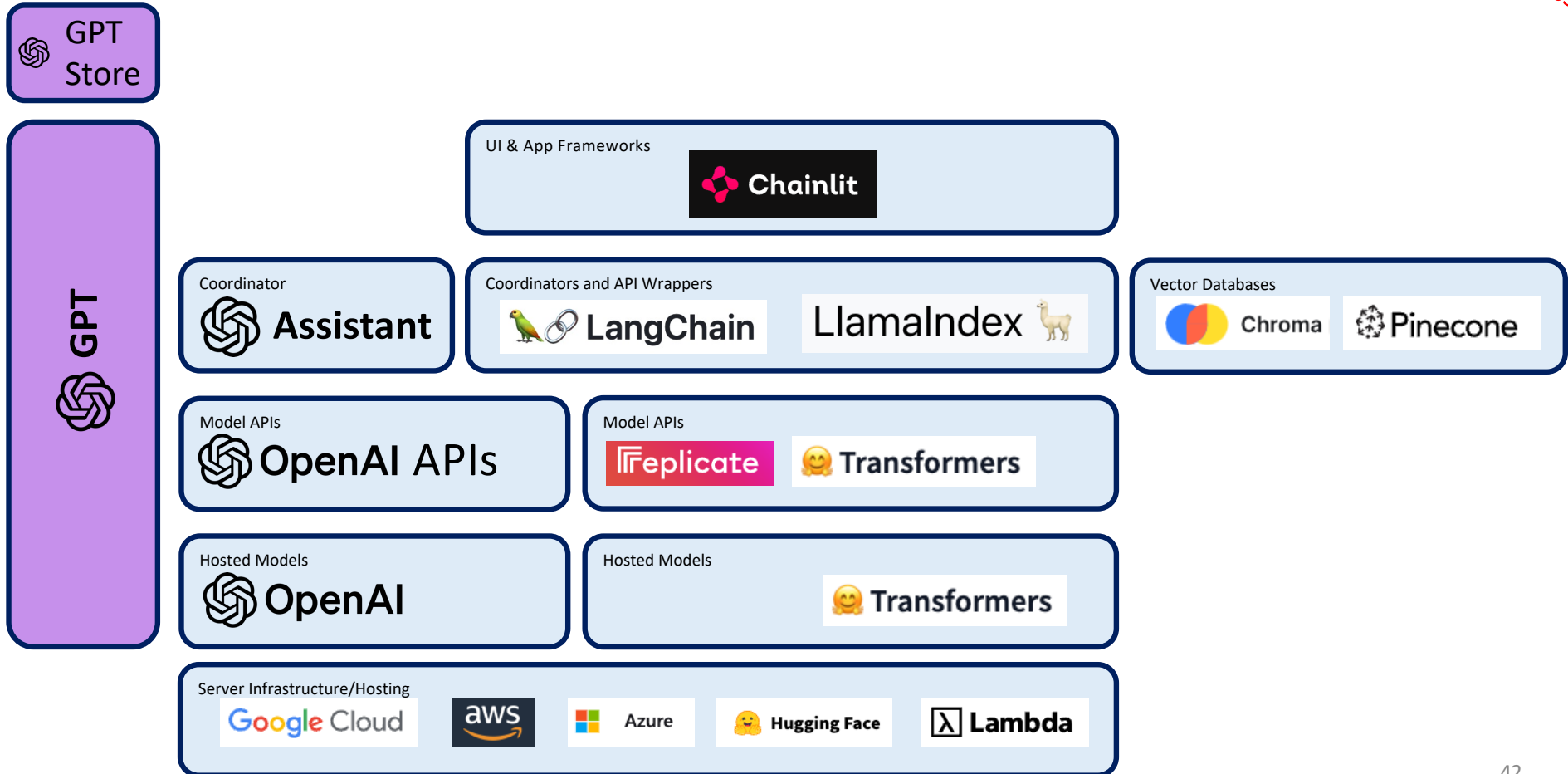


# Hierarchy of Cognitive Architectures



# CogArch SW Stack & Ecosystem

*Work in Progress*



# Tool use / Plugins

Offload tasks that LLMs are not good at  
Importantly: they don't "know" they are not good

## Intersperse text with special tokens that call external APIs

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

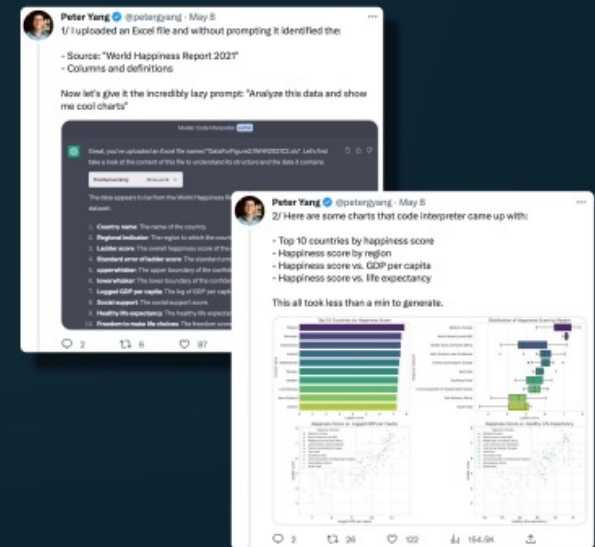
Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

[Toolformer, Schick et al. 2023]

## ChatGPT plugins

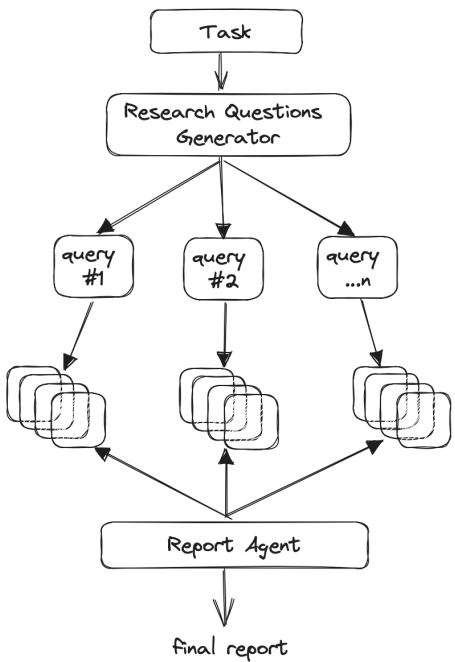


## ChatGPT Code Interpreter



<https://karpathy.ai/stateofgpt.pdf>

# LLMs Chains & Agents



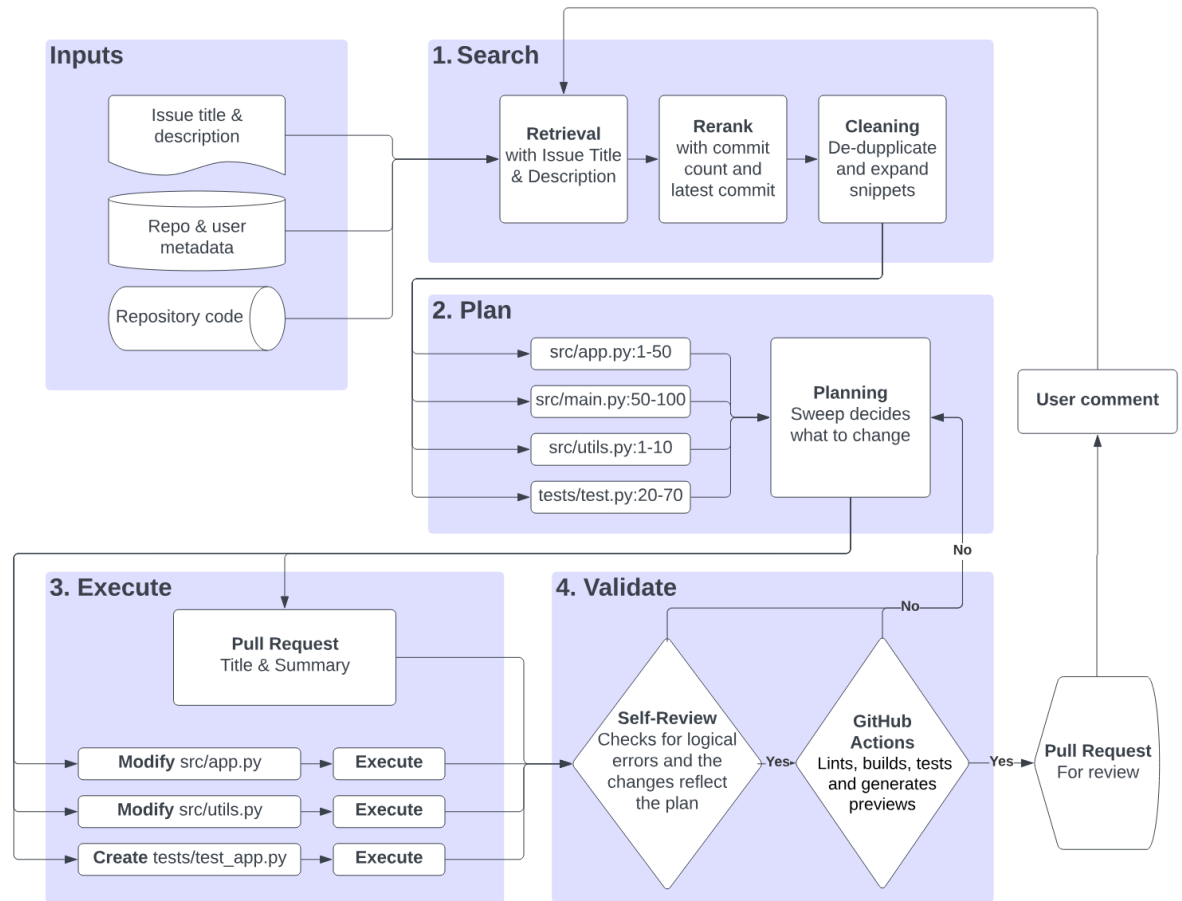
### [AutoGPT]

### [ReAct: Synergizing Reasoning and Acting in Language Models, Yao et al. 2022]

<https://github.com/assafelovic/gpt-researcher>

<https://karpathy.ai/stateofgpt.pdf>

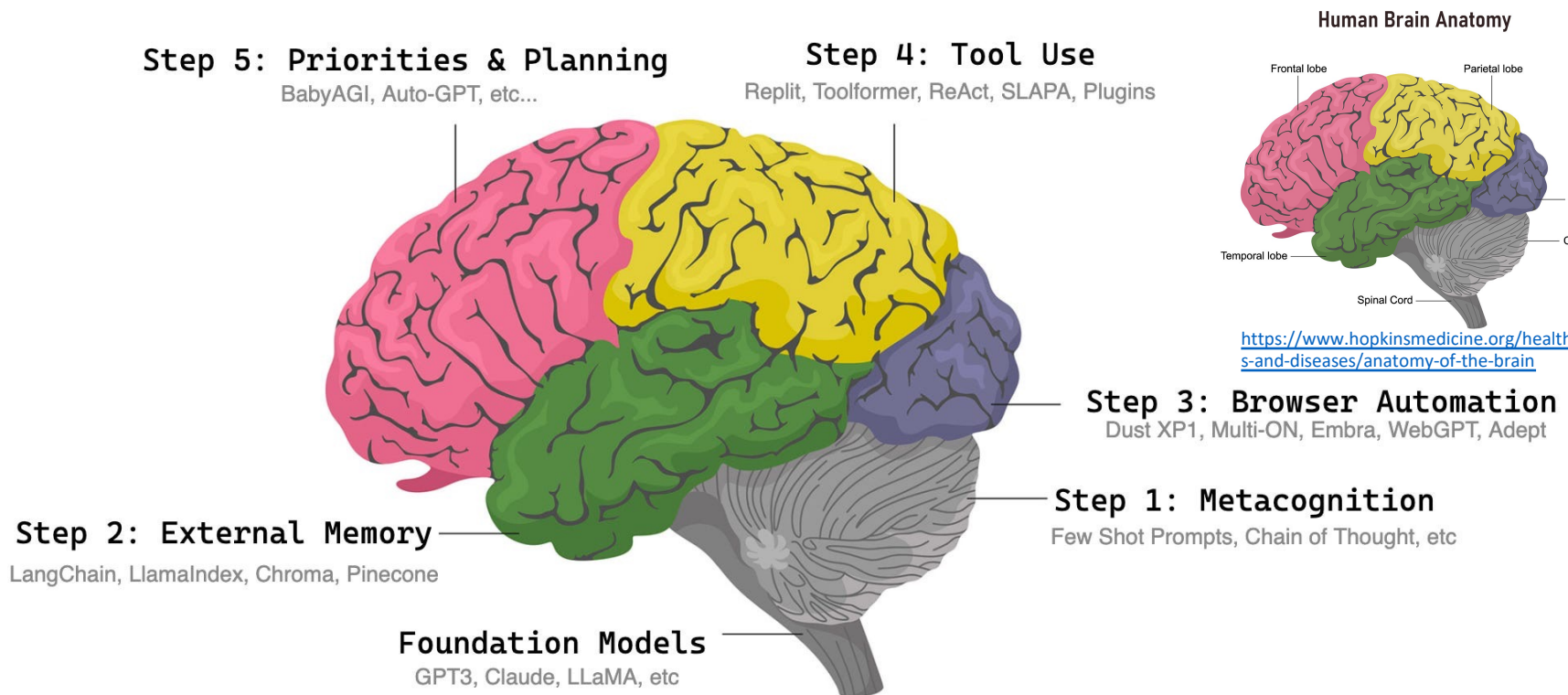
# sweep.dev cognitive architecture



<https://docs.sweep.dev/blogs/sweeps-core-algo>

An ecosystem view

# The Anatomy of Autonomy

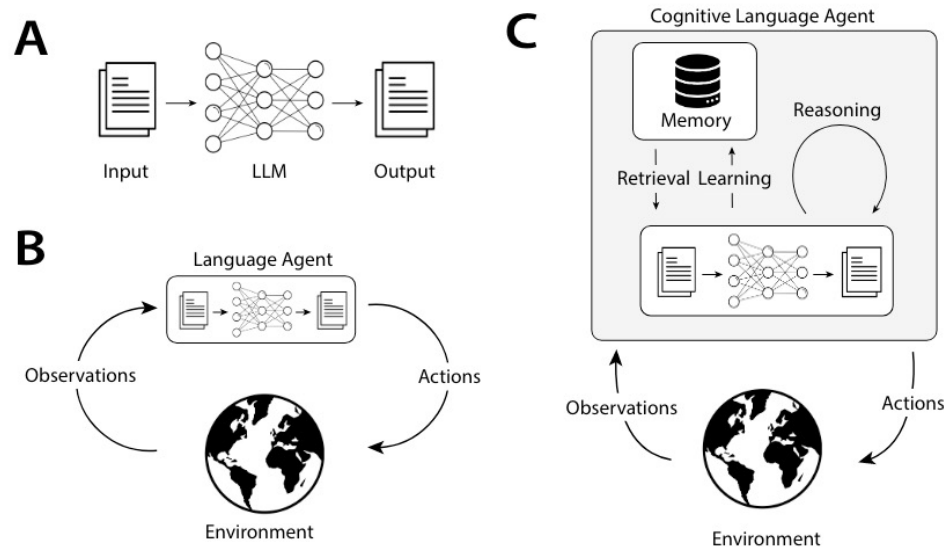


<https://www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-of-the-brain>

<https://www.latent.space/p/agents>

# Cognitive Architectures for Language Agents

- “draw on rich history of cognitive science and symbolic artificial intelligence...”
- CoALA: Cognitive Architectures for Language Agents

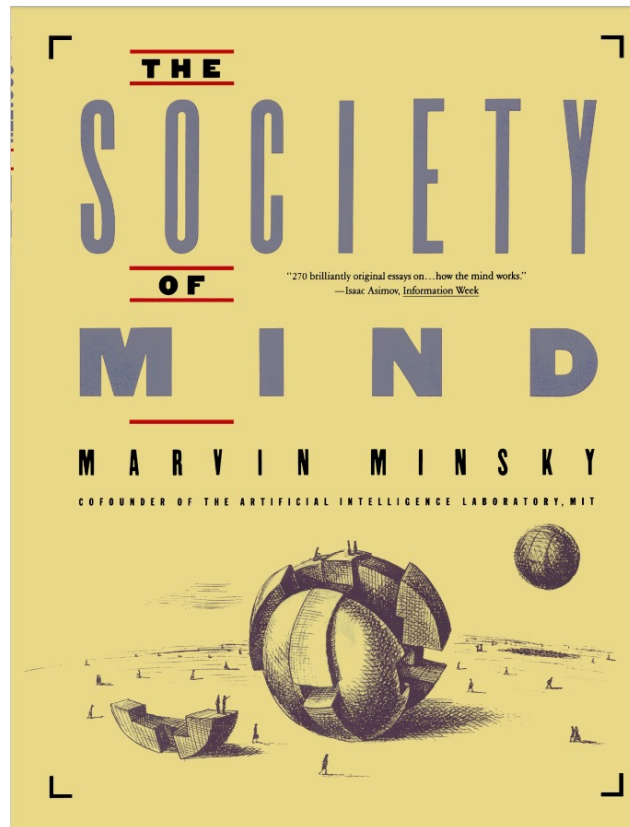


A: In natural language processing (NLP), an LLM takes text as input and outputs text.

B: Language agents place the LLM in a direct feedback loop with the external environment by transforming observations into text and using the LLM to choose actions.

C: Cognitive language agents additionally use the LLM to manage the agent's internal state via processes such as learning and reasoning.

# Theories of Intelligence



*The Society of Mind* is both the title of a 1986 book and the name of a theory of natural [intelligence](#) as written and developed by [Marvin Minsky](#).<sup>[1]</sup>

In his book of the same name, Minsky constructs a model of human intelligence step by step, built up from the interactions of simple parts called [agents](#), which are themselves mindless. He describes the postulated [interactions](#) as constituting a "society of mind", hence the title.<sup>[2]</sup>

[https://en.wikipedia.org/wiki/Society\\_of\\_Mind](https://en.wikipedia.org/wiki/Society_of_Mind)



<https://web.media.mit.edu/~minsky/papers/ConfocalMemoir.html>  
<https://patents.google.com/patent/US3013467A/en>



# Next Time

- LLM Finetuning
- back to book sequence on
  - unsupervised learning
  - GANs
  - VAEs
  - Diffusion Models

[Feedback](#)

