Introduction to LLM for Communication Engineering Students

Cheng-Shang Chang (張正尚)

July 25, 2024

國立清華大學
智慧感知聯網研究中心
Internet of Senses Research Center

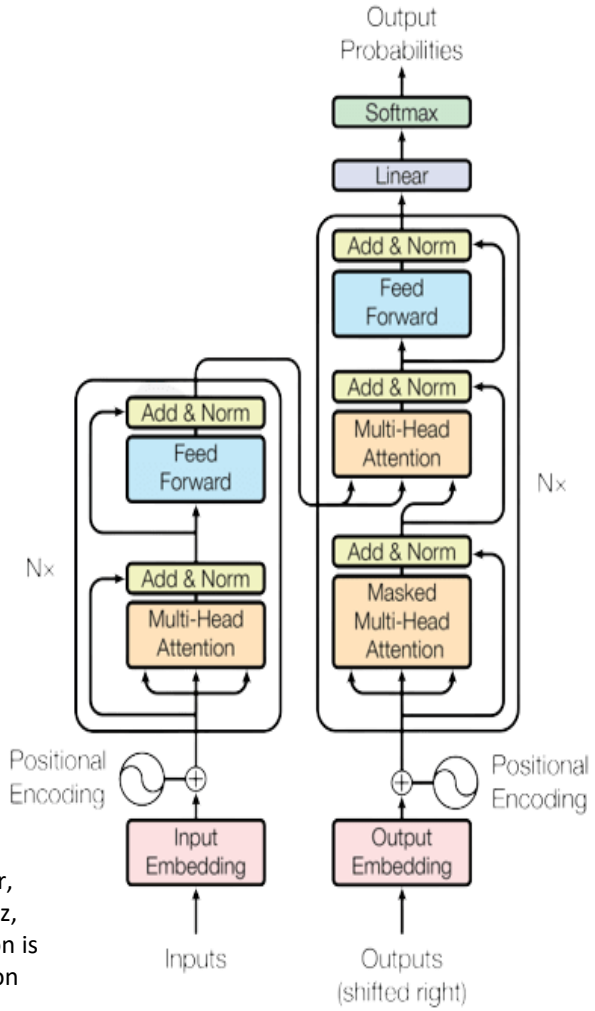
Institute of Communications Engineering
National Tsing Hua University

Graduate students

- 黃屏倫
- 劉奕緯
- 陳品仲
- 吳彥杰

Large Language Models (LLM)

- LLMs take a sequence of tokens as their input (prompts) and generate another sequence of tokens as their output (answers).
- Denote by I (resp. O) be the input (resp. output) token space.
- An LLM can be represented by a random function: $\phi: \mathcal{I}^N \mapsto \mathcal{O}^N$.

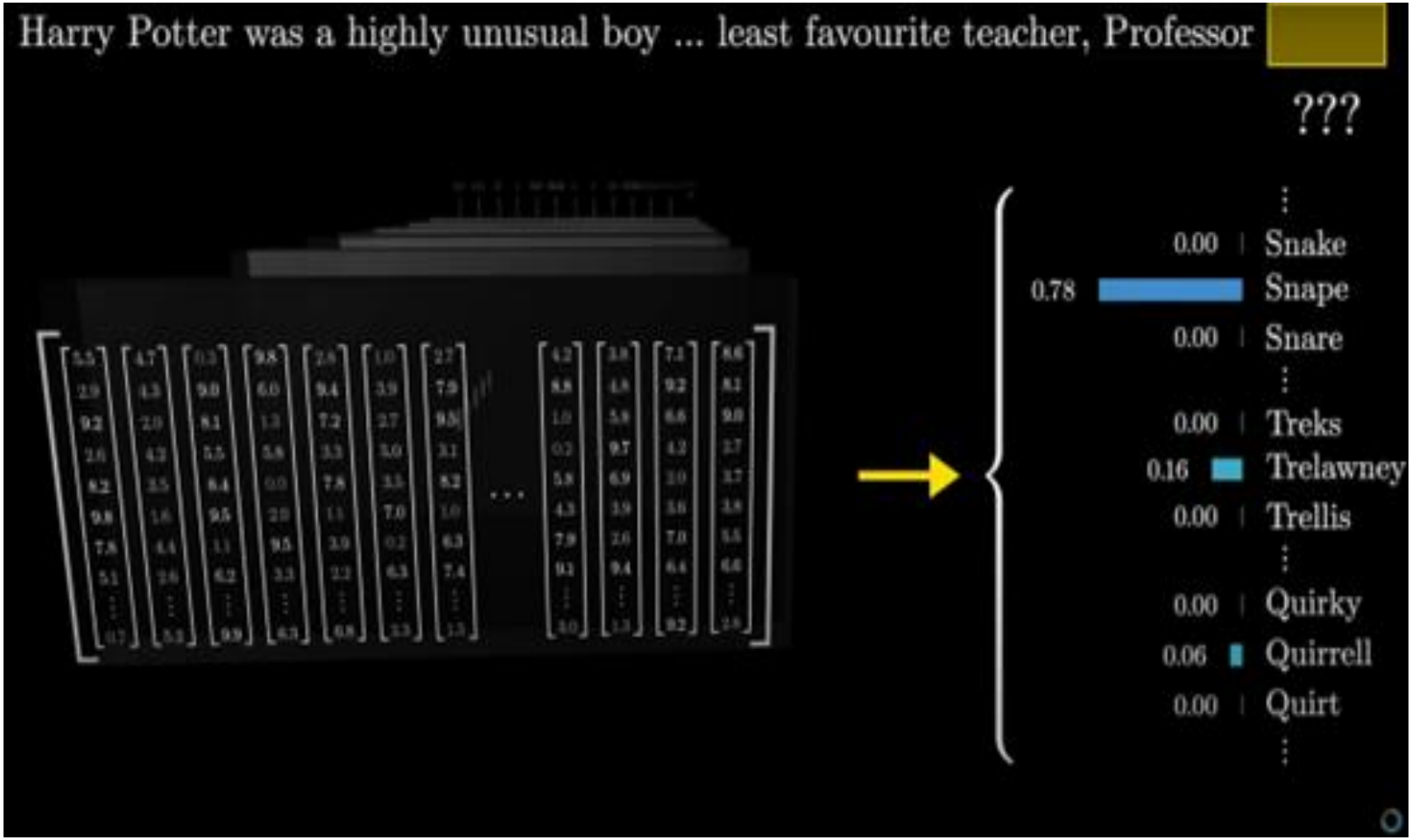


Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Autoregressive LLM

- Next token prediction
- Denote the input of N tokens by $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N)$
- Denote the output of N tokens by $\mathrm{y} = (\mathrm{y}_1, \mathrm{y}_2, \ldots, y_N)$
- $y_{<t} = (y_1, y_2, \ldots, y_t)$
- An LLM is said to be autoregressive if y_{t+1} is a random function of x and $y_{<t}$.
- A Transformer-based autoregressive LLM generates a set of candidate tokens (e.g., the top-p sampling) and use a selection method (e.g., the Boltzmann selection) to select one token from the candidate set.

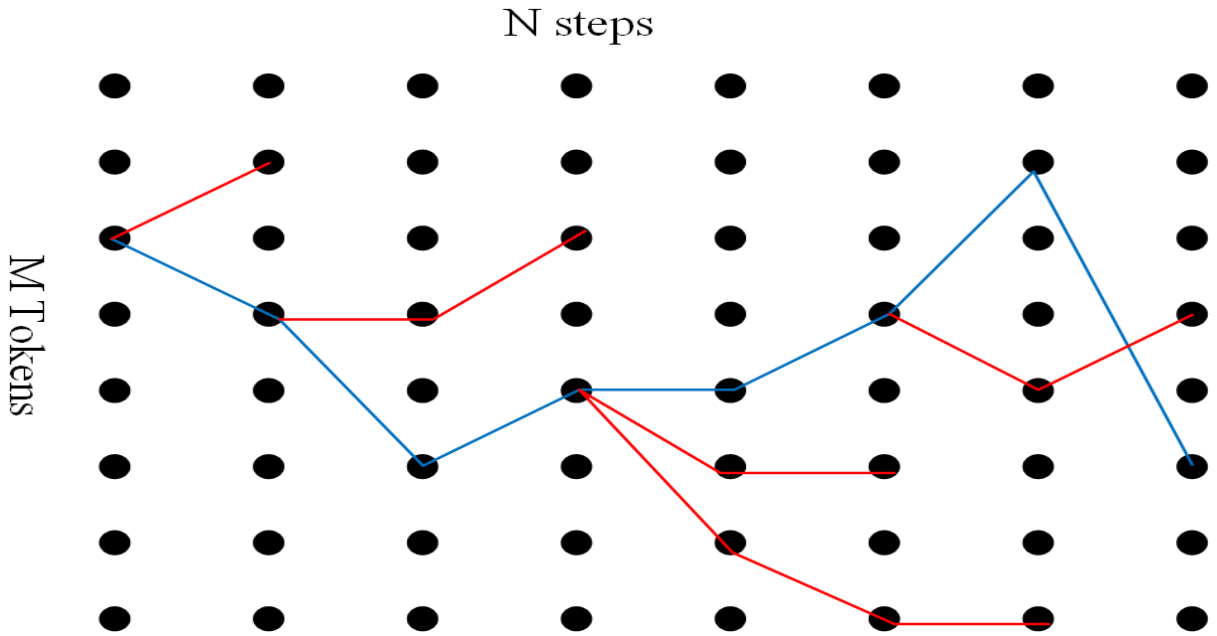
Next token prediction



3Blue 1Brown (https://www.youtube.com/watch?v=wjZofJX0v4M&t=123s)

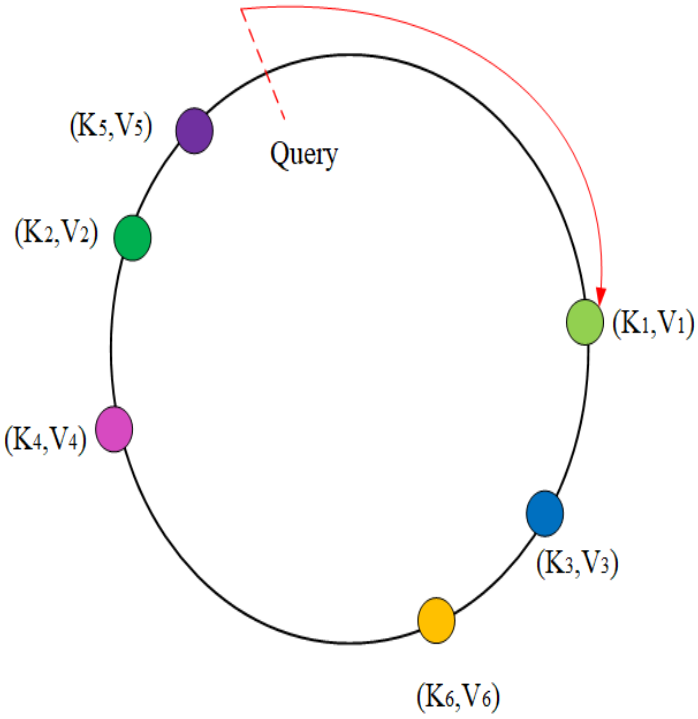
Next token prediction

- GPT-4, M=100K tokens (CL 100K tokenizer)
- N=8192 tokens (context length)



LLM as an Associative Memory

- Consistent hashing is widely used in modern data centers and peer-to-peer networks
- The (Q,K,V)-architecture with a hash function *h*
- Every data point containing a key (K) and a value (V) is hashed into a ring.
- By assigning the points in the ring to the closest hashed data point in the clockwise direction, there is an interval associated with each data point.
- A query (Q) that falls in the interval associated with a specific data point retrieves the value of that data point.



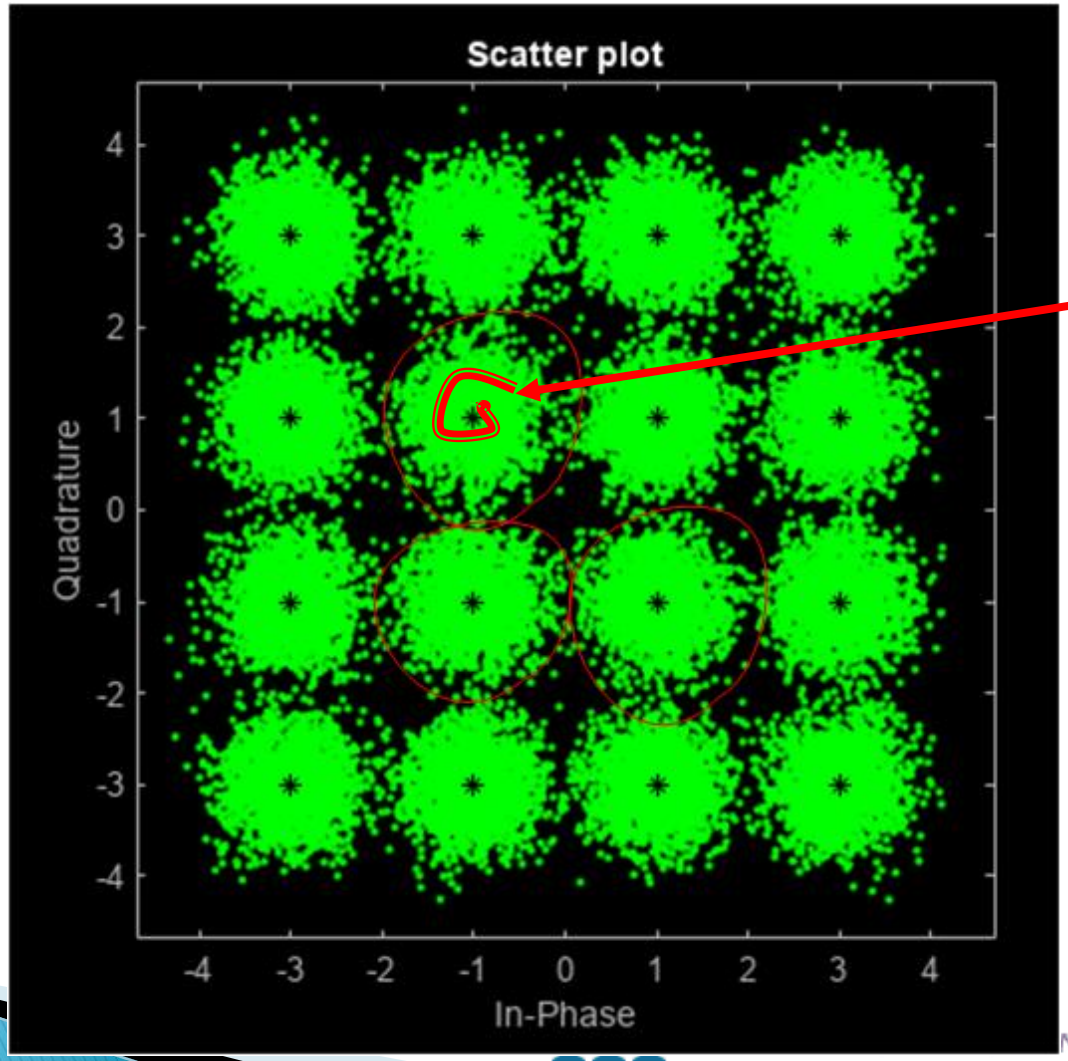
LLM as an Associative Memory

- Transformer-based LLMs utilize the (Q,K,V)-architecture
- A key *K* consists of a sequence of tokens (representing a text).
- The value (*V*) associated with the key is an embedding vector in a high-dimensional Euclidean space.
- By using a large number of tokens for training, one can train a transformer with a large number of parameters to obtain a hash function *h* that maps a key *K* to an embedding vector *V*.

LLM as an Associative Memory

- A Transformer retrieves the value for a query Q by finding the "closest" key K to the query Q.
- Given the large number of keys, checking every key in the associated memory to identify the "closest" key is not feasible.
- Transformers are also known to be Hopfield networks.
- Every key in a Hopfield network is associated with a ``region of convergence'' in the Euclidean space, akin to the associated intervals in consistent hashing.

Region of Convergence like 16-QAM



Query

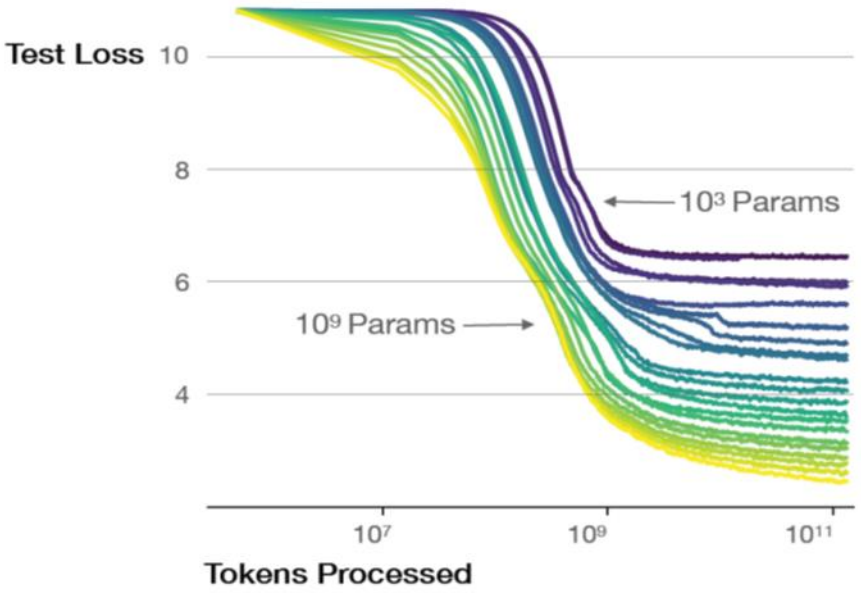
16-QAM from MathWorks

LLM as an Associative Memory

- Given a query *Q* represented by a sequence of tokens, the query is first mapped to a sequence of vectors by a tokenizer.
- The sequence of vectors is repeatedly updated by the attention mechanism in each layer depicted in the transformer architecture.
- Each update reduces the energy in the corresponding Hopfield network, and eventually, the sequence of vectors converges to the "closest" key.
- The embedding vector of that key is then retrieved as the value of the query *Q*.
- A query *Q* that falls in the region of convergence of a key retrieves the value of that key.
- Such an iterative update approach is similar to how humans retrieve patterns in their brains.

LLM as an Associative Memory

- The capacity of the associative memory should be large number to "store" the patterns learned from the dataset
- Need a large number of parameters to store the patterns for the knowledge collected from the Internet



Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361* (2020).

LLM as an Associative Memory

- Hallucination (decoding errors) as a byproduct of lossy compression of the associative memory
- During training, LLMs like GPT-4 process vast amounts of text data from the internet. The training involves creating compressed representations of this data, capturing essential patterns and relationships while discarding less critical details.
- Due to the lossy nature of their training, they sometimes rely on patterns that seem plausible but are not grounded in the actual data, leading to the generation of false or misleading information, known as hallucinations.

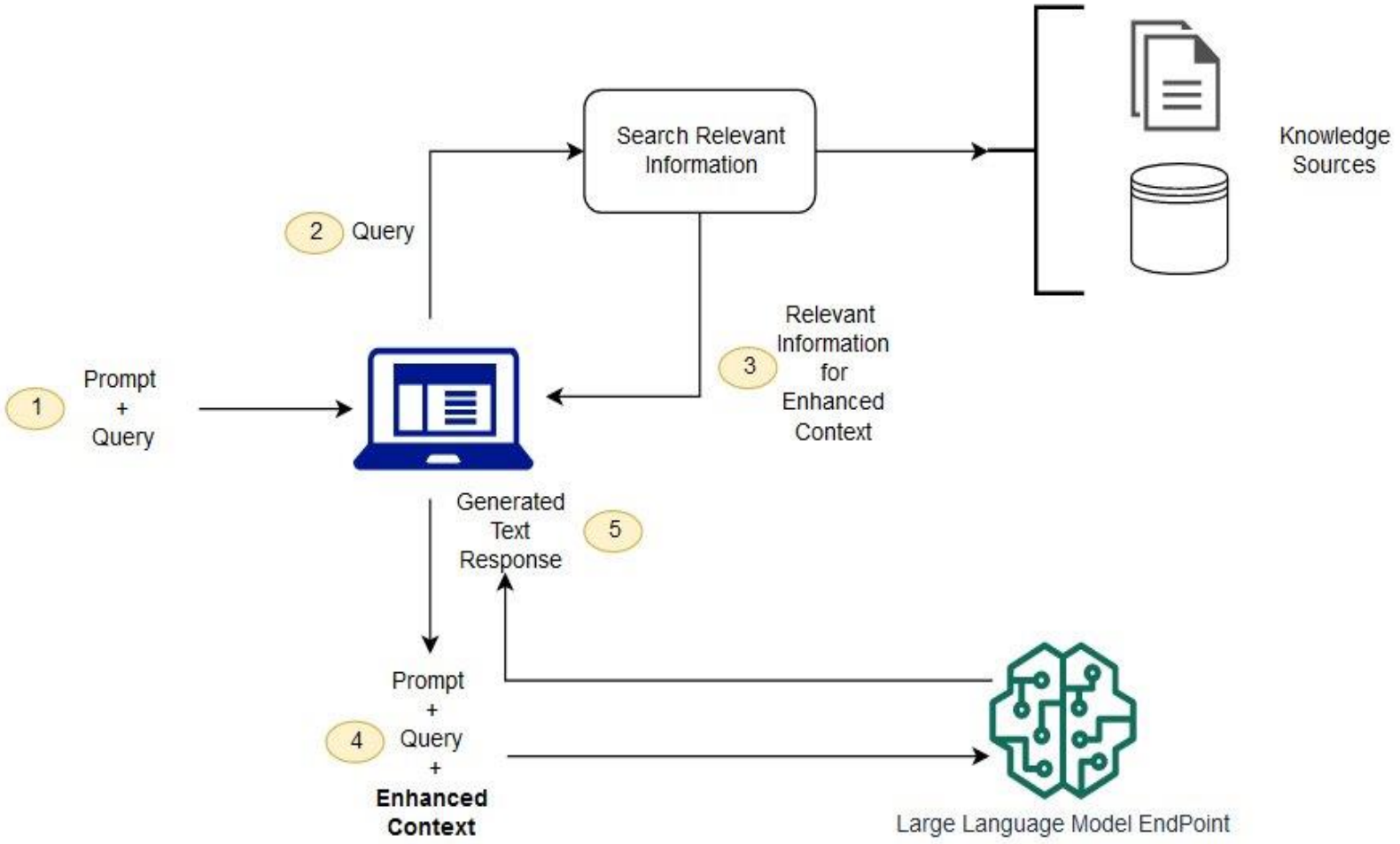
Mitigating Hallucination

- Treating LLM as a black box
- Improvement from the input:
  - Prompt engineering
  - A common approach is known as Retrieval-Augmented Generation (RAG)
- Improvement from the output:
  - Meta-generation decoding algorithms that call sub-generators
- Cognitive architectures

Retrieval-Augmented Generation (RAG)

- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks(2020 Patrick Lewis et al.)
- LLMs' knowledge is limited to the public data up to a specific point in time that they were trained on.
- Need to augment the knowledge of the model with the specific information it needs.
- The process of bringing the appropriate information and inserting it into the model prompt is known as Retrieval Augmented Generation (RAG).

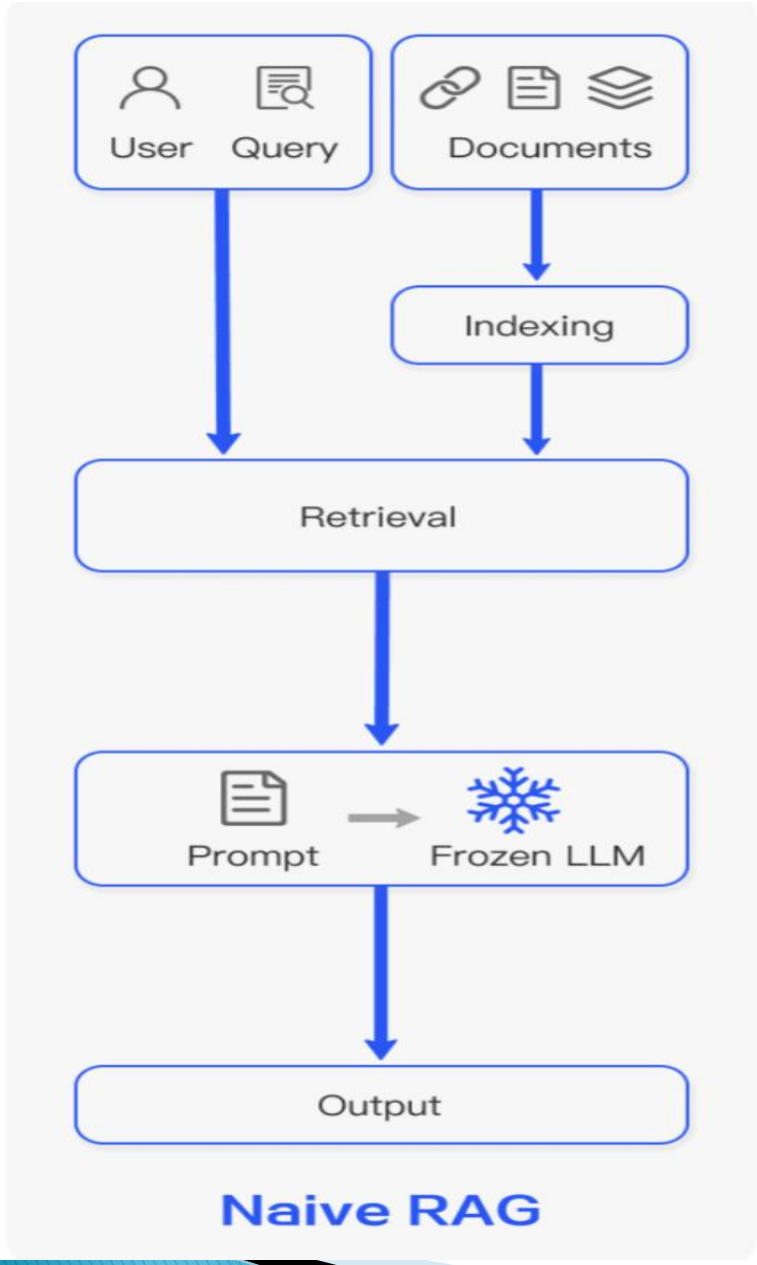
Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems* 33 (2020): 9459-9474.

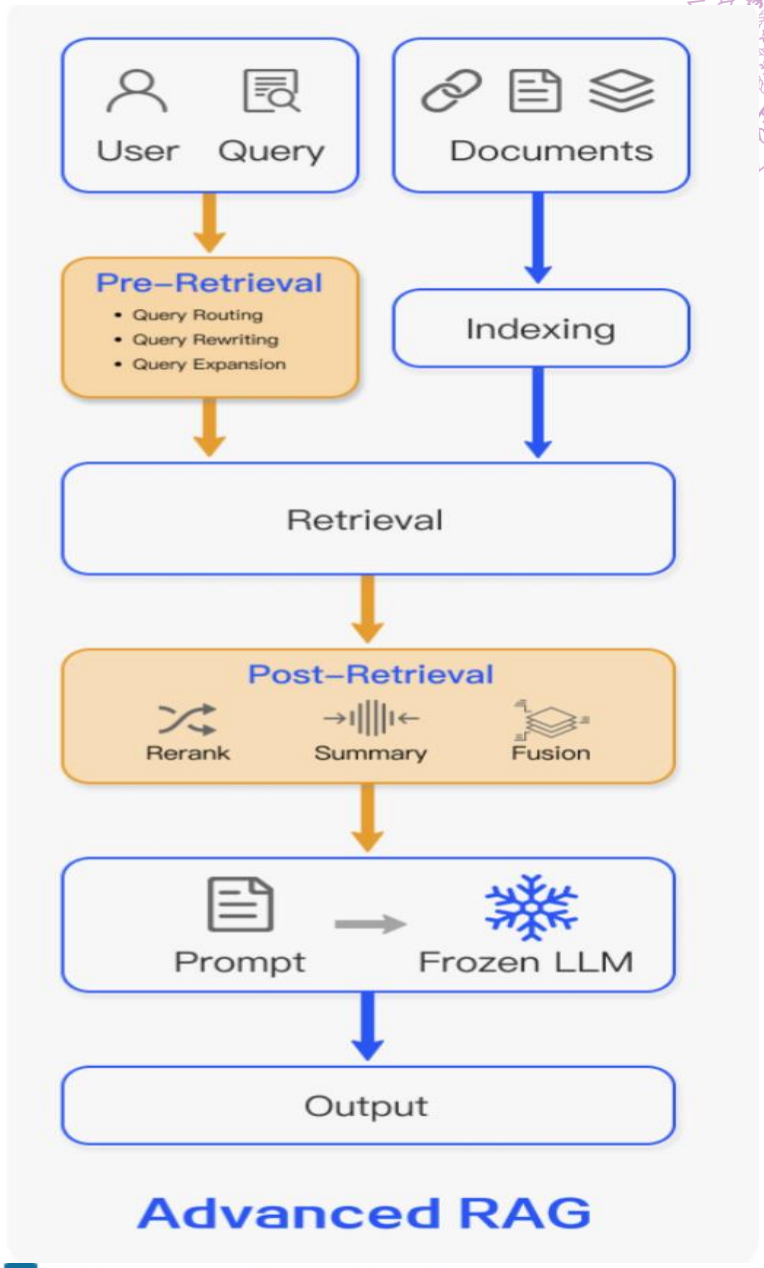


https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-customize-rag.html

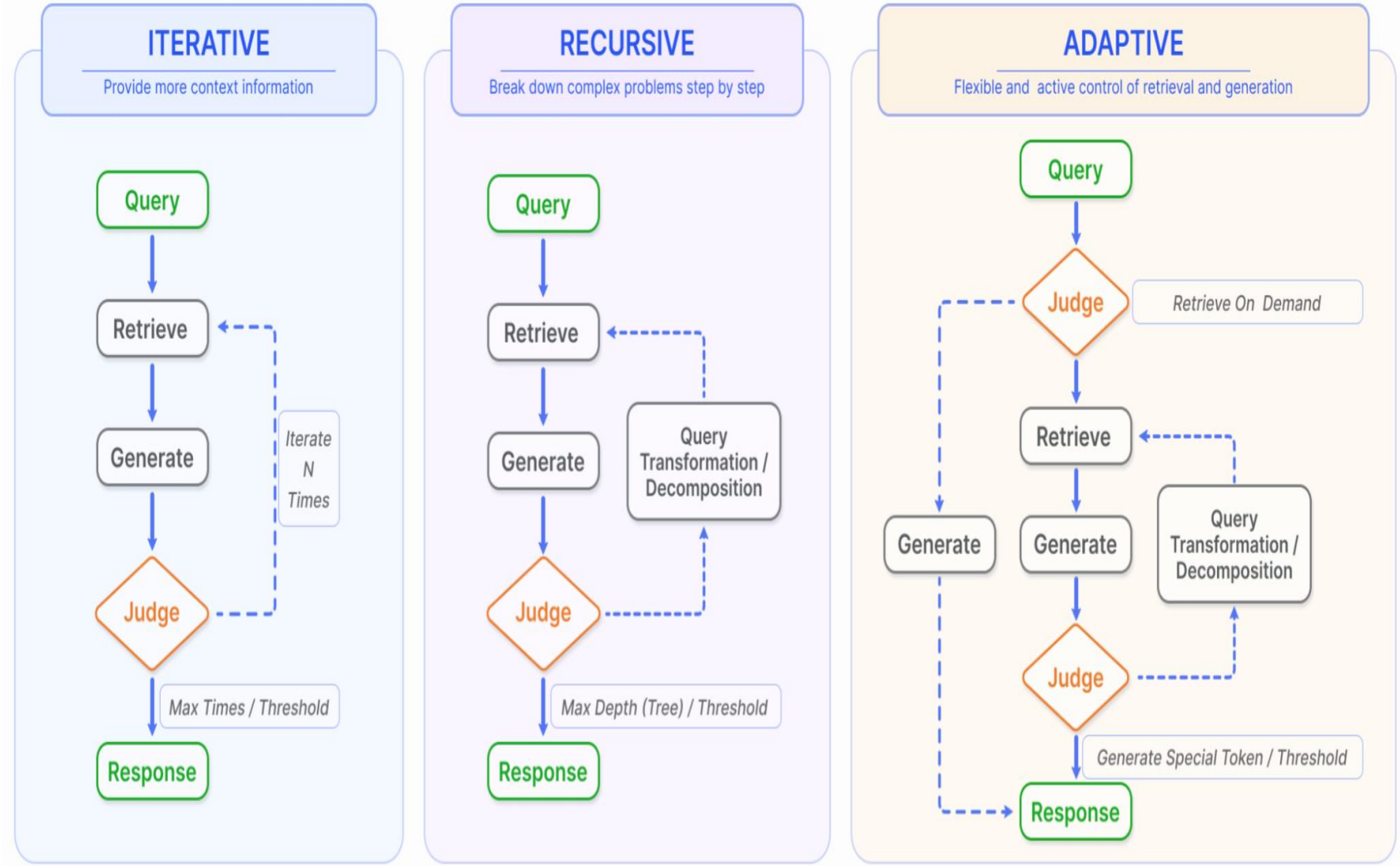
Naive RAG encounters notable drawbacks:

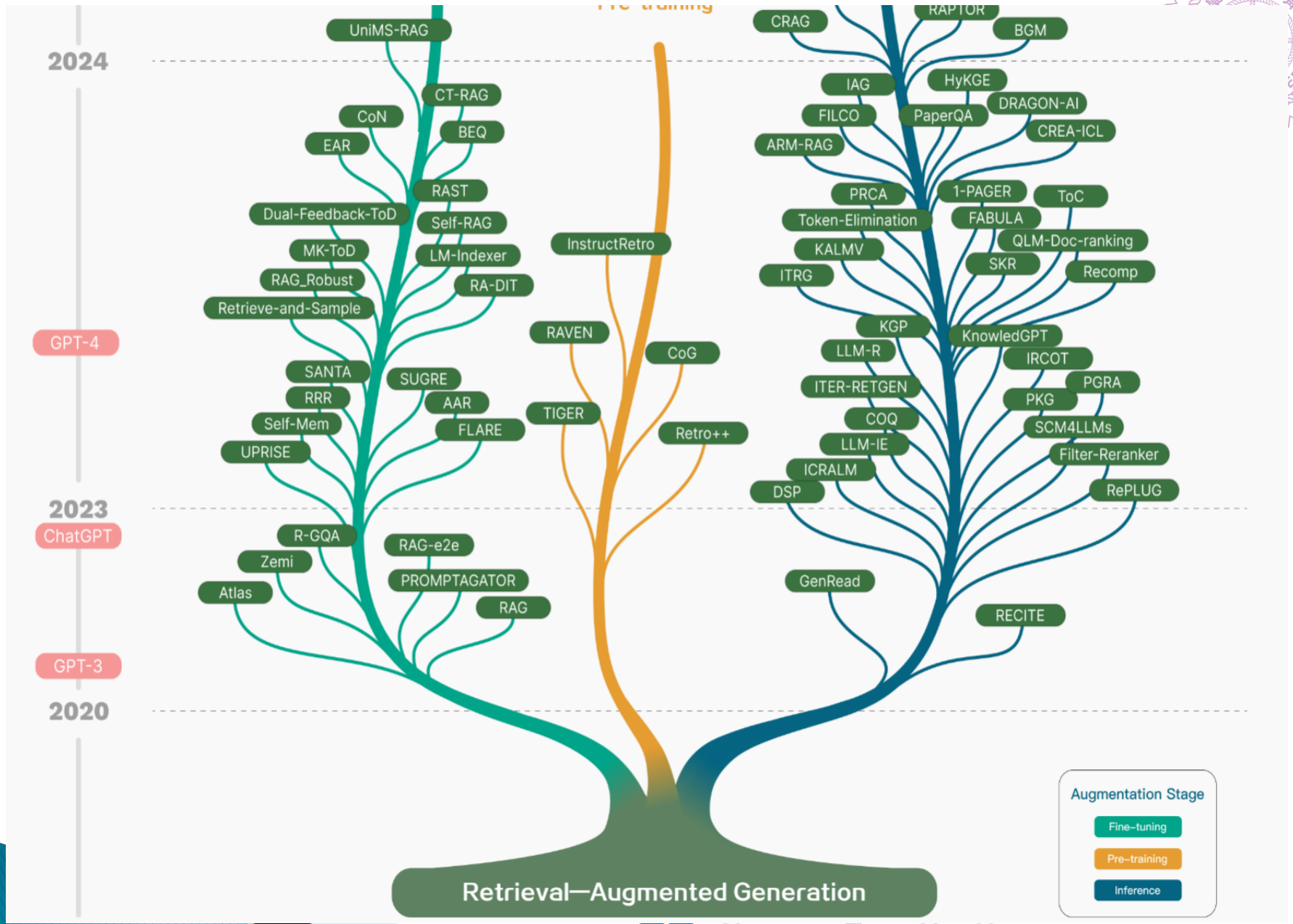
- Retrieval Challenges: The retrieval phase often struggles with precision and recall, leading to the selection of misaligned or irrelevant chunks, and the missing of crucial information.
- Generation Difficulties: In generating responses, the LLM model may face the issue of hallucination


User
Query
Documents
Indexing
Retrieval
Prompt
Frozen LLM
Output
Naive RAG


User
Query
Documents
Pre-Retrieval
Query Routing
Query Rewriting
Query Expansion
Indexing
Retrieval
Post-Retrieval
Rerank
Summary
Fusion
Prompt
Frozen LLM
Output
Advanced RAG

AUGMENTATION PROCESS IN RAG




2024
GPT-4
2023
ChatGPT
GPT-3
2020
Pre-training
UniMS-RAG
CT-RAG
CoN
BEQ
EAR
RAST
Dual-Feedback-ToD
Self-RAG
MK-ToD
LM-Indexer
RAG_Robust
RA-DIT
Retrieve-and-Sample
SANTA
SUGRE
RRR
AAR
Self-Mem
FLARE
UPRISE
R-GQA
RAG-e2e
Zemi
PROMPTAGATOR
Atlas
RAG
InstructRetro
RAVEN
CoG
TIGER
Retro++
CRAG
RAPTOR
BGM
IAG
HyKGE
FILCO
PaperQA
DRAGON-AI
ARM-RAG
CREA-ICL
PRCA
1-PAGER
ToC
Token-Elimination
FABULA
KALMV
QLM-Doc-ranking
ITRG
SKR
Recomp
KGP
KnowledGPT
LLM-R
IRCOT
ITER-RETGEN
PGRA
PKG
COQ
SCM4LLMs
LLM-IE
Filter-Reranker
ICRALM
DSP
RePLUG
GenRead
RECITE
Retrieval—Augmented Generation
Augmentation Stage
Fine-tuning
Pre-training
Inference

Meta-Generation

- Token-level generation algorithms
- Meta-generation describes algorithms that call sub-generators
  - Chained meta-generators
  - Parallel meta-generators
  - Step-level meta-generators
  - Refinement-based meta-generators

Welleck, Sean, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. "From Decoding to Meta-Generation: Inference-time Algorithms for Large Language Models." arXiv preprint arXiv:2406.16838 (2024).

Token-level generation algorithms

- Assume to have access to a language model's logits $s_\theta(y_{<t}, x)$ and next-token distribution $p_\theta(y|x)$
- Maximum a-posteriori (MAP) decoding algorithms: select the most likely sequence

$$\text{argmax}_{y \in Y} \max p_\theta(y|x)$$

- Greedy decoding: recursively selecting the highest probability token from the next-token distribution

$$\begin{aligned}\arg\max_{y \in \mathcal{Y}} p_\theta(y|x) &= \arg\max_{(y_1,\ldots,y_T) \in \mathcal{Y}} \prod_{t=1}^{T} p_\theta(y_t|y_{<t}, x) \\ &\approx \left(\hat{y}_1 = \arg\max_{y_1 \in \mathcal{V}} p_\theta(y_1|x), \cdots, \hat{y}_T = \arg\max_{y_T \in \mathcal{V}} p_\theta(y_T|\hat{y}_{<T}, x) \right).\end{aligned}$$

- Pitfall: degenerating behavior, i,e., favor shorter sequences

Decoding by sampling

- Ancestral sampling: $p_\theta(y|x)$
- Other methods: top-p sampling, top-k sampling….
- Sampling adapter:
  - Example (Temperature sampling as an adapter)

$$\exp(s_\theta(y_{<t}, x)/T)$$

  - Uniform sampling when the temperature *T* is large
  - Reduce to MAP when the temperature *T* is small

Controlled sampling

- Modulate by a sequence level criterion *c(y)*

$$p_\theta(y|x)c(y)$$

- Classifier: $p_\theta(y|x)p(a|x,y)^\beta$
- Indicator: $p_\theta(y|x)\mathbf{I}_{y\in Y_x^*}$
- Reward: the reward function could be learned by reinforcement learning from human feedback

$$p_\theta(y|x)\exp(\frac{1}{T}r(x,y))$$

Meta-generation algorithms

- **Definition:** Higher-level strategies that operate on sequences rather than individual tokens.
- Chained meta-generators: a paper outline, sections, and a revision to satisfy the page limit
- Self-Ask (Press et al., 2023) alternates between prompting a language model to generate a sub-question and calling a search engine to find an answer to the question.
- System 2 Attention (Weston & Sukhbaatar, 2023) uses multi-step generation to help the model refrain from attending to irrelevant information.

Meta-generation algorithms

- **Parallel meta-generators:**
  - **Reranking algorithms**: order an N-best list with a reranking, then selects the top-k ranked sequences.
  - **best-of-N** refers to generating an N-best list and picking the best sequence according to a scoring function
  - **Best-of-N in reasoning:** In some settings the goal is to generate correct sequences, such as a correct solution to a mathematical problem or a program that passes test cases. A common approach in these cases is to learn a verifier $v(y) \rightarrow [0, 1]$ that predicts the probability that an output y is correct, and use it within Best-of-N.

Meta-generation algorithms

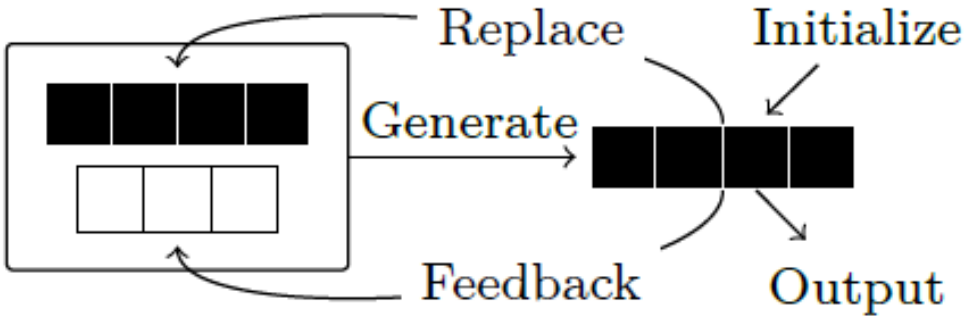
- **Parallel meta-generators:**
  - **Transformation algorithms**: transform the list into a new sequence which might not be part of the N-best list itself.
  - **Majority voting**: First, this algorithm processes an N-best list $(y^{(1)}, \ldots, y^{(N)})$ and counts how each of the candidates $y^{(i)}$ votes towards a different set of outputs $(a_1, \ldots, a_K)$. Second, it selects the output that received the largest number of votes.
  - Other methods: Weighted majority voting, Self-consistency, Minimum Bayes Risk decoding…

Meta-generation algorithms

- **Step-level search algorithms**
- Token level beam search (list decoder): a breadth-first search algorithm that maintains a fixed number of the most promising candidate sequences at each step, called the "beam width."
- Partial sequence expansion
  - Tree-of-thoughts
  - Monte-carlo tree search
  - Graph-of-thoughts

Meta-generation algorithms

- **Refinement algorithms:**



(c) Refinement

- At present, refinement usually works best for tasks that either have rich environment feedback or can be reliably evaluated by current language models.

Cognitive Architectures for Language Agents (CoALA) Agentic RAG

- **Need memory**
- Language models are stateless: they **do not persist information** across calls.
- In contrast, language agents may store and maintain information internally for multi-step interaction with the world.
- **Short-term**: working memory
- **Long-term** memories: episodic, semantic, and procedural.

Sumers, Theodore R., Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. "Cognitive architectures for language agents." *arXiv preprint arXiv:2309.02427* (2023).

Cognitive Architectures for Language Agents (CoALA)

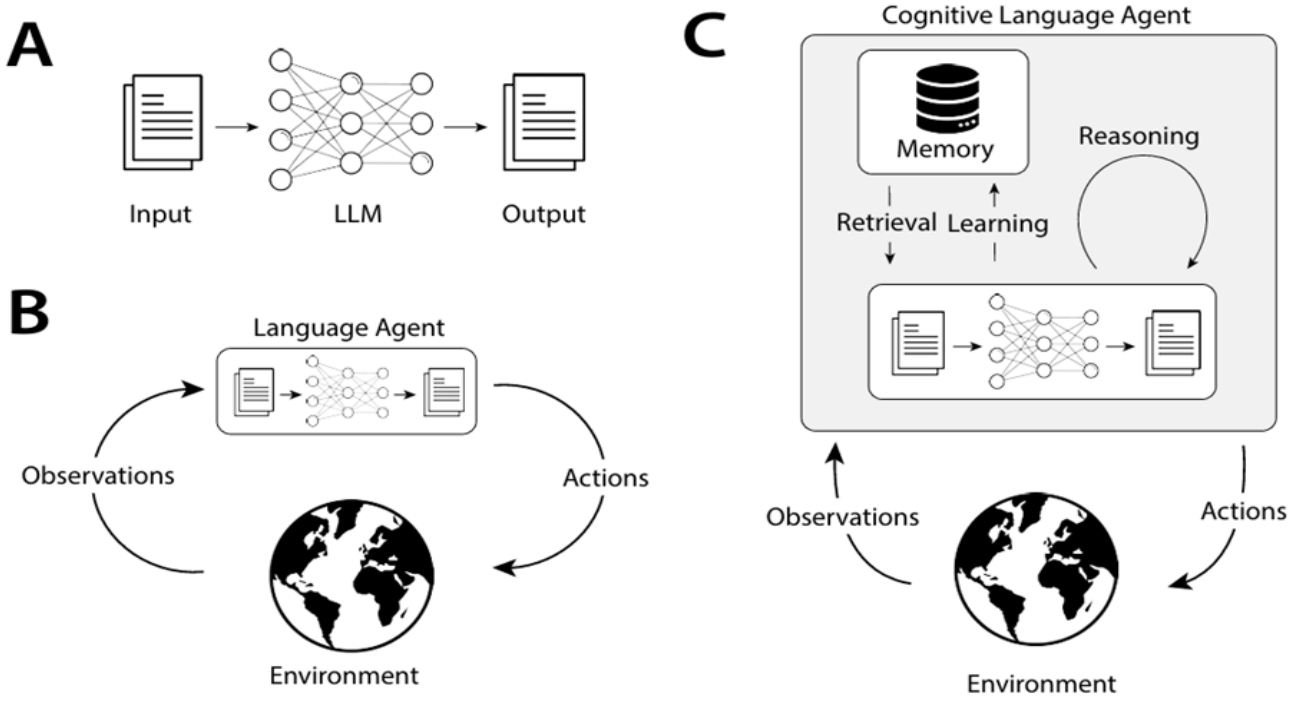


Figure 1: Different uses of large language models (LLMs). **A**: In natural language processing (NLP), an LLM takes text as input and outputs text. **B**: *Language agents* (Ahn et al., 2022; Huang et al., 2022c) place the LLM in a direct feedback loop with the external environment by transforming observations into text and using the LLM to choose actions. **C**: *Cognitive* language agents (Yao et al., 2022b; Shinn et al., 2023; Wang et al., 2023a) additionally use the LLM to manage the agent's internal state via processes such as learning and reasoning. In this work, we propose a blueprint to structure such agents.

Cognitive Architectures for Language Agents (CoALA)

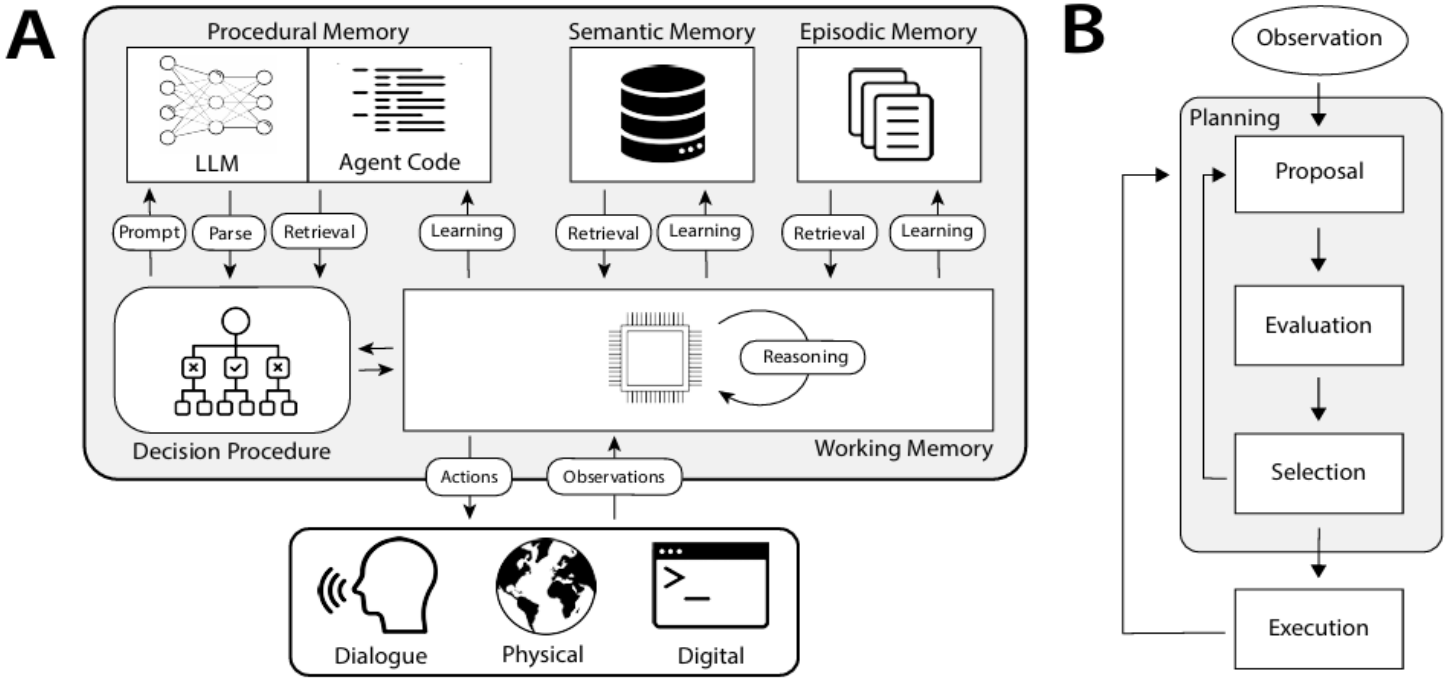
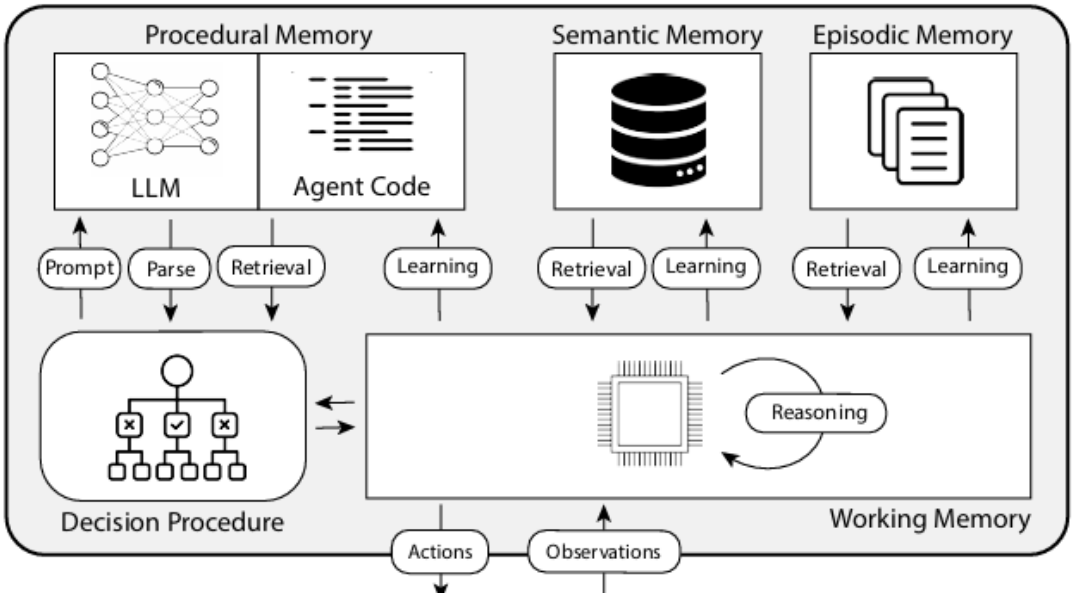


Figure 4: Cognitive architectures for language agents (CoALA). **A**: CoALA defines a set of interacting modules and processes. The **decision procedure** executes the agent's source code. This source code consists of procedures to interact with the LLM (prompt templates and parsers), internal memories (retrieval and learning), and the external environment (grounding). **B**: Temporally, the agent's decision procedure executes a **decision cycle** in a loop with the external environment. During each cycle, the agent uses **retrieval** and **reasoning** to plan by proposing and evaluating candidate **learning** or **grounding** actions. The best action is then selected and executed. An observation may be made, and the cycle begins again.

Cognitive Architectures for Language Agents (CoALA)

- **Working memory**
  - A data structure that persists across LLM calls.
  - Working memory also **interacts with long-term memories and grounding interfaces**.



Cognitive Architectures for Language Agents (CoALA)

- **Episodic memory**
  - Episodic memory stores experience
  - This can consist of history event flows, …, or other representations of the agent's experiences
- **Semantic memory**
  - Semantic memory stores an agent's knowledge
  - A form of learning to incrementally build up world knowledge from experience.
- **Procedural memory**
  - Language agents contain two forms of procedural memory: **implicit** knowledge stored in the **LLM weights**, and **explicit** knowledge written in the **agent's code**.

Conclusion

- LLM is an associative memory
- The capacity of an LLM should be large enough to store the world knowledge
- Hallucination is a byproduct of lossy compression of the associative memory
- Hallucination can be mitigated by
  - Retrieval-Augmented Generation (RAG)
  - Meta-generation decoding algorithms
  - Cognitive architectures