An Introduction to LLM Inference and Reasoning

Speaker: Zijian Zhou

Table of Content

Some Fundamentals

A large language model (aka LLM)

- Is Large in parameters size (usually more than 1B parameters)
- Operates on “language”

Some Fundamentals

What is ”language” for a model?

First, we need a way to chunk the language into units for the model to understand and predict

- Split by letter, word, sentence, paragraphs?

Letter: The vocabulary is simple - just 26 letters (at least for English), but each letter contains almost no semantic information!

Word: The vocabulary is close to human-level – around 10k, but what if there are rare words? (like mis-spelling or ”chillax”)

Some Fundamentals

What is ”language” for a model?

(Current) Solution: we split a language into “subwords”

Subwords contain both letters and words, or part of the words.

=> Can represent all kinds of “words”, and contain sufficient semantic meaning.

Some Fundamentals

Challenge: deciding on the vocabulary is difficult: What subwords should we include in the vocabulary?

Byte-Pair Encoding (BPE) is a method for balancing the vocab size and the semantic richness.

Rough idea:

- first split the language into the smallest units (e.g., letters)
- then merge the letters into most frequently-appeared subwords
- Do this until a desired vocabulary size is reached.

Some Fundamentals

E.g., to build a vocabulary for

"hug", "pug", "pun", "bun", "hugs”

First split them into letters

("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Merge the most frequent pair of letters

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)

Continue

("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

…

Reference: https://huggingface.co/learn/llm-course/en/chapter6/5

Table of Content

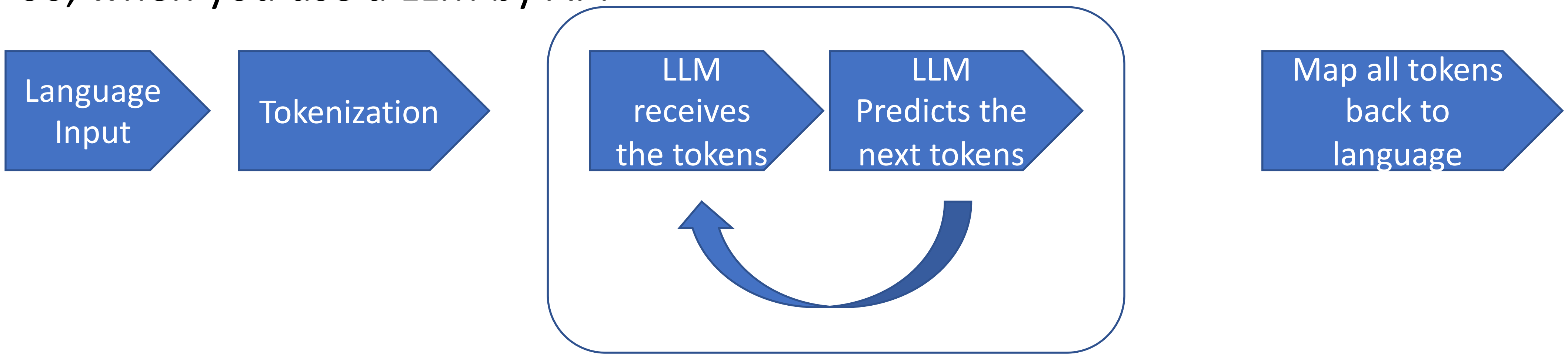
- Some Fundamentals
- LLM Inference
- Reasoning

LLM Inference

This process is called ”**tokenization**”.

LLMs model the “**token space**” by predicting **what should be the next token**, given a set of tokens.

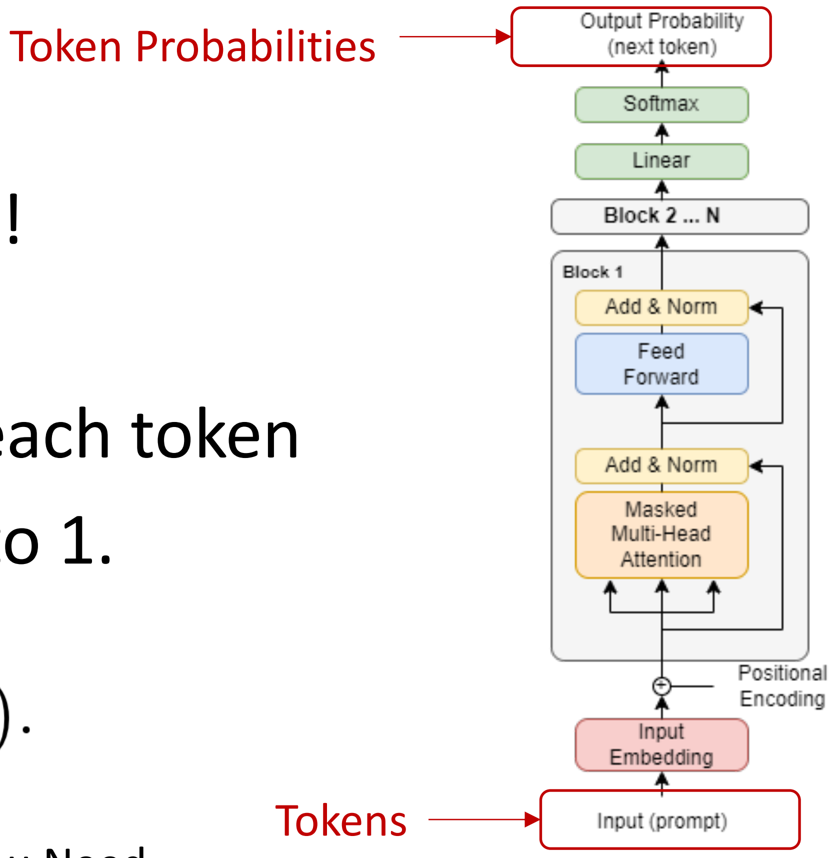
So, when you use a LLM by API



Recursively (or, “auto-regressively”)

LLM Inference

- **Caveat**: a language model DOES NOT predict just one single output token
- It predicts a DISTRIBUTION!



A probability is assigned to each token

In the vocabulary, summed to 1.

$$P(Y \mid X) = \prod_{i=1}^{N} P\big(y_i \mid y_{<i}, X\big).$$

Reference: Vaswani et. al., Attention Is All You Need

LLM Inference

How is the next token selected then?

It is sampled, with a few tricks to improve robustness.

`temperature` number or null Optional Defaults to 1

What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. We generally recommend altering this or `top_p` but not both.

`top_p` number or null Optional Defaults to 1

An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

We generally recommend altering this or `temperature` but not both.

Reference: https://platform.openai.com/docs/api-reference/responses/get

LLM Inference

Temperature (T):

$$P(y_i \mid y_{<i}, X) \; = \; \mathcal{F}\left(\text{softmax}\left(\frac{z_i}{T}\right)\right)$$

where z_i is the logit of y_i. Higher T means the logits are more even => the probability distribution is more spread out.

Reference: https://platform.openai.com/docs/api-reference/responses/get

LLM Inference

Top p: sort all probabilities in descending order. Select only from tokens with cumulative probabilities up to p (usually $p \geq 0.9$).

Top k: sort all probabilities in descending order. Select only from the top k tokens with highest probabilities. (Not so popular now as top p can usually handle the work better)

Reference: https://platform.openai.com/docs/api-reference/responses/get

LLM Inference

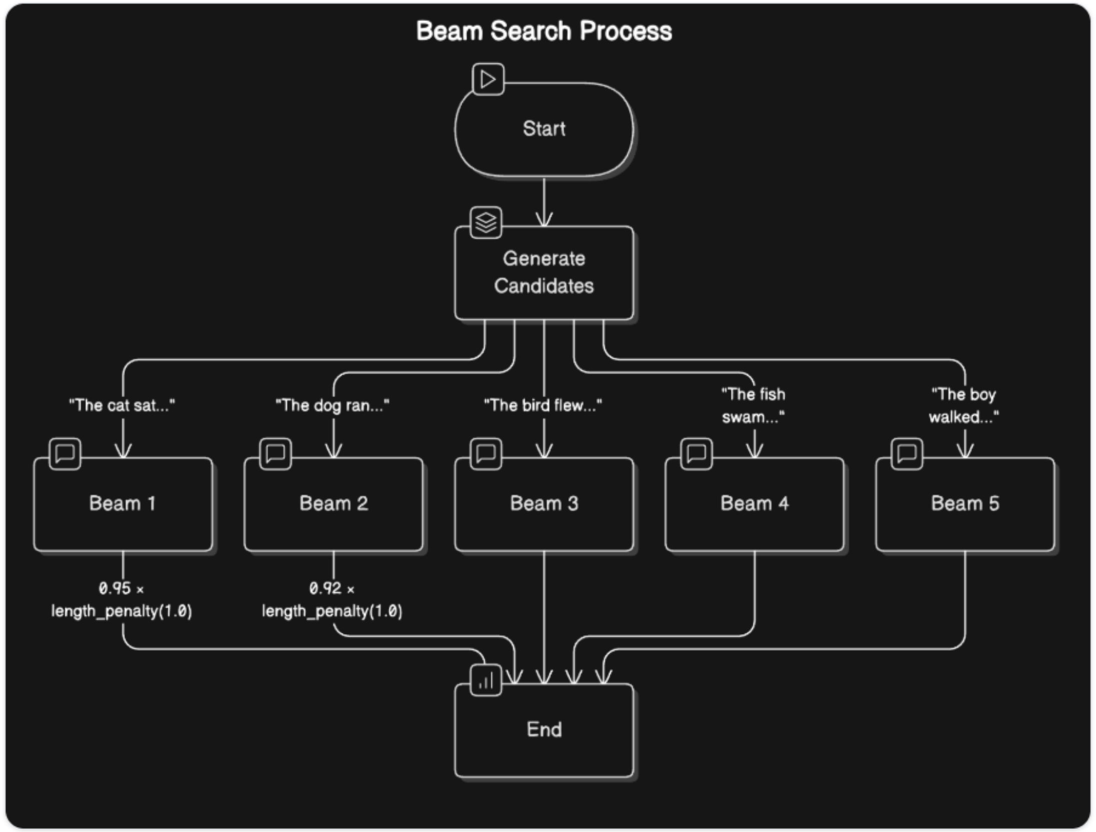
Note: sampling allows many tricks to be applied to LLMs for all kinds of interesting purposes.

Some examples

- Beam Search (next slide)
- Guided Decoding (in assignment)
- Horizontal Scaling of LLM Reasoning (discussed later)

LLM Inference

Beam search: Generating multiple candidate sequences at the same time and at each time, only keep the most possible next tokens among all candidates.



Reference: https://medium.com/@sulbha.jindal/llm-inferencing-strategies-review-of-greedy-search-and-beam-search-cfbdb96e021a

LLM Inference

A remaining question: how does a LLM know when to stop?

A specialized token (usually represented as <EOS>) is artificially added into the vocabulary to train the LLM.

The loop breaks when a <EOS> token is sampled. The special token is not existent in the language space.

LLM Inference

In the same vein, special tokens are also used to handle conversations.

Take Qwen series of models for example,

- A system message always starts with <|im_system|> and ends with <|im_end|>
- A user message starts with <|im_user|> and ends with <|im_end|>
- An assistant message starts with <|im_assistant|> and ends with <|im_end|>

Then, a conversation is just concatenating all message tokens

LLM Inference

Note: Special tokens are handy tools to make LLM perform various tasks (when **trained well**).

Common usage:

- `<BOS>`: start of a sequence
- `<EOS>`: stop of a sequence
- `<PAD>`: let the LLM skip this token (useful in training)

More advanced usage:

- `<summarize>`: make the LLM summarize the previous text
- `<switch>`: make the LLM perform subtasks during the generation
- …

Fast LLM Inference

Fast LLM Inference

KV-Cache

The heaviest part in LLM computation is to calculate “attention”, i.e., the relevance scores between different tokens in the input.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The attention scores $(QK^T/\sqrt{d_k})$, can be reused.

The attention weights $softmax(\frac{QK^T}{\sqrt{d_k}})$, can be easily computed with cached scores.

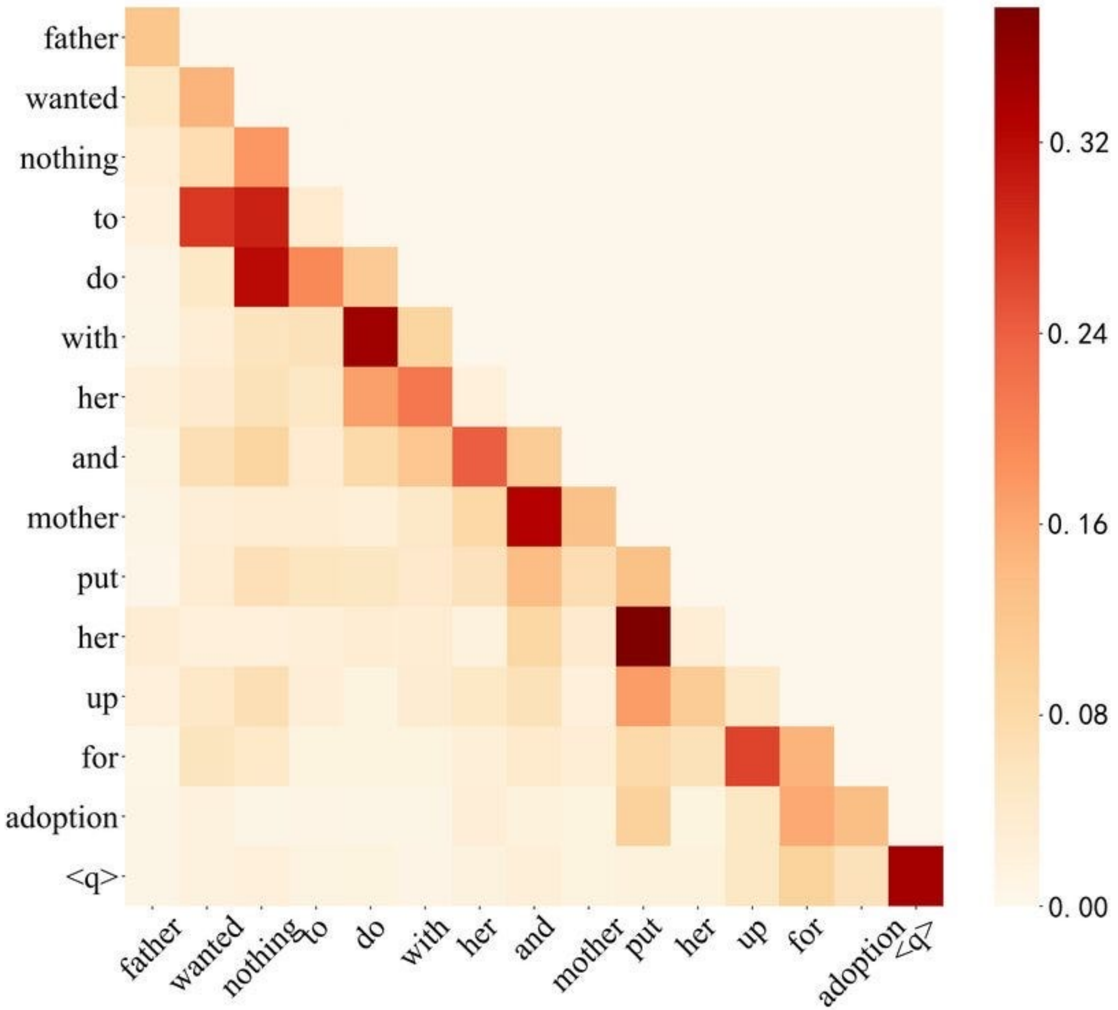
Reference: Vaswani et. al., Attention Is All You Need

Fast LLM Inference

Attention weights represented as a 2-D matrix

Note that only the lower triangular parts have values,

because only later tokens can attend to previous tokens.

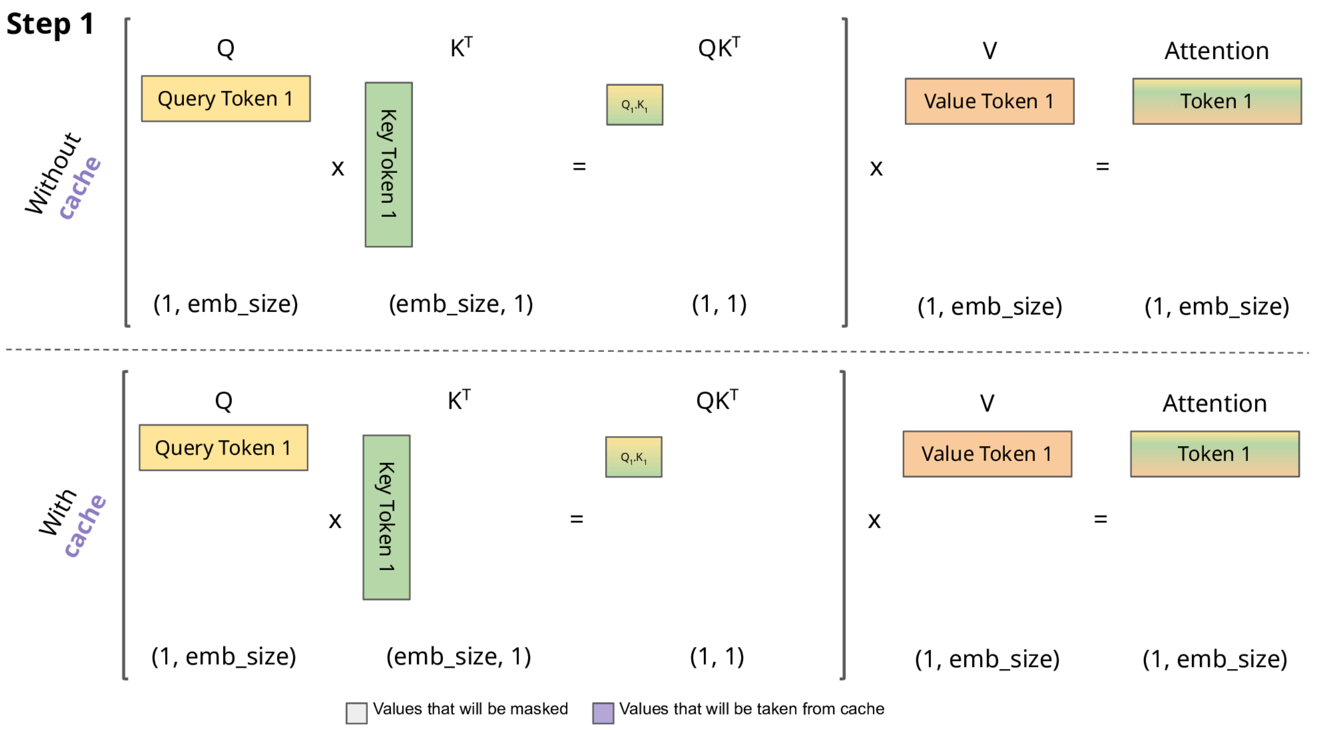


Reference: https://medium.com/@AIExplainedML/how-does-the-attention-mechanism-in-gpt-models-work-5f489a59346b

Fast LLM Inference

KV Cache

In auto-regressive (AR) generation, we compute the attention of the new token against existing tokens



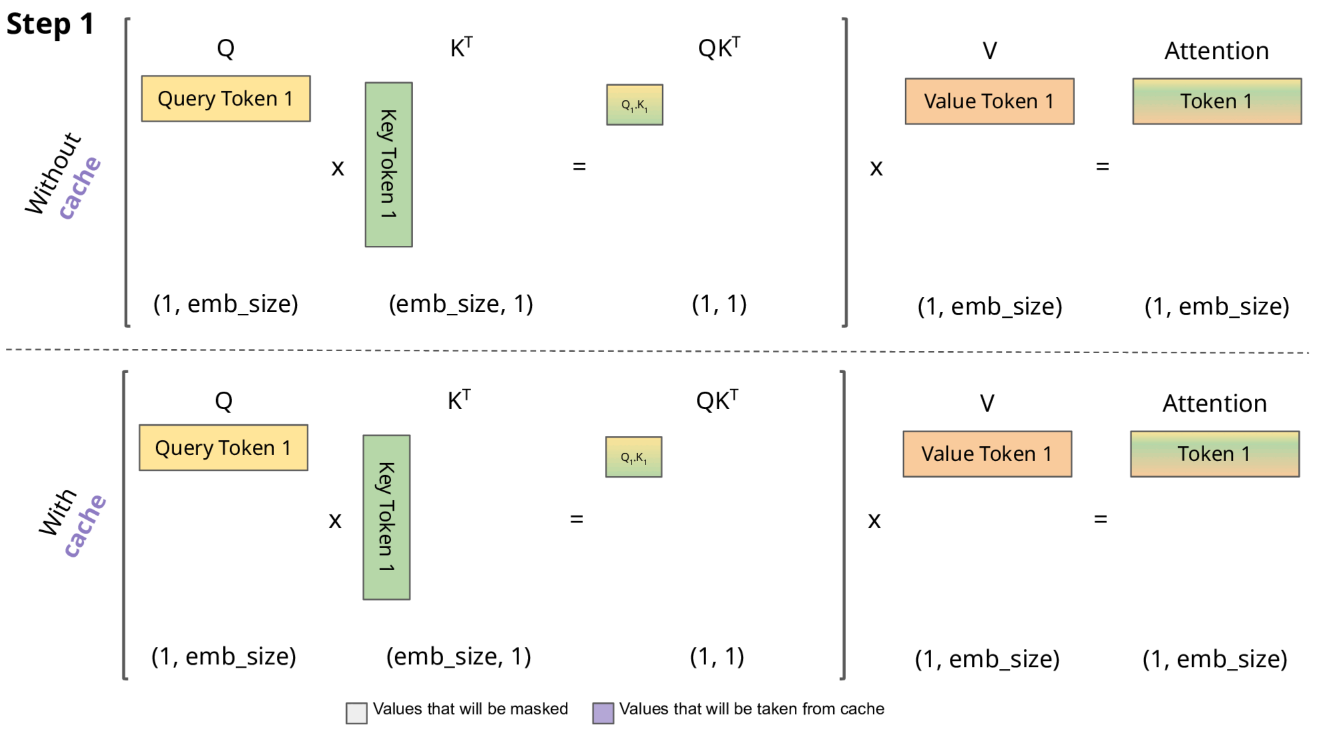
Previous attention scores can be reused

Reference: https://medium.com/@joaolages/kv-caching-explained-276520203249

Fast LLM Inference

KV Cache

Time Complexity reduced from $O(n^2)$ to $O(n)$,



where n refers to the number of tokens so far

Reference: https://medium.com/@joaolages/kv-caching-explained-276520203249

Fast LLM Inference

KV Cache

Provided the memory can store all the KV-cache, the time taken to generate a new token grows in $O(n)$.

Reference: https://medium.com/@joaolages/kv-caching-explained-276520203249

Fast LLM Inference

KV Cache

With KV Cache, the matrix computation is not very intensive to GPUs.

Problem: the next token can only be computed after the previous token is generated.

=> We say AR decoding is, in general, IO bound (i.e., time is mainly wasted in waiting).

Fast LLM Inference

Prefilling

When a LLM first receives a user's prompt, there is no cached attention weight (i.e., no KV cache).

In this case, intensive computation is required to compute the attention matrix for all prompt tokens.

Thankfully, with GPU, the matrix computation can be done in parallel.

We say the prefilling stage is, in general, compute bound (i.e., time is mainly used for heavy computation).

Fast LLM Inference

LLM Generation

Prefilling	AR Decoding
Applies to prompt tokens	Applies to output tokens
Compute Bound	IO Bound
Takes up lots of compute resource	Takes up relatively smaller compute resource
Faster	Slower

Fast LLM Inference

Can we trade compute for IO during AR?

Yes! We can use a smaller and faster model to quickly draft a few tokens.

And then let the LLM "prefill" these tokens all at once.

[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~.~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

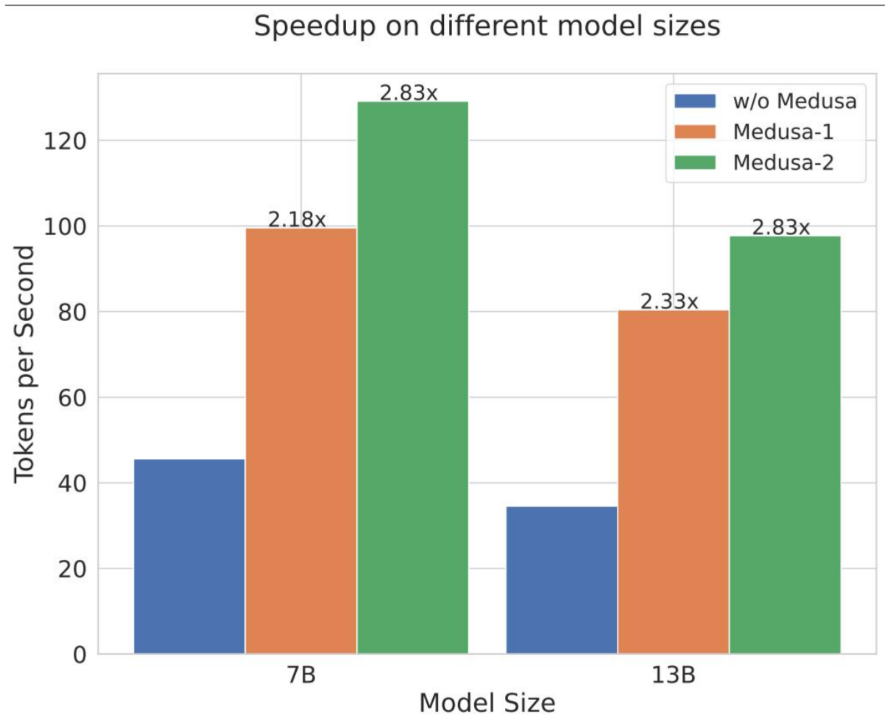
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or ~~0~~ 1

Reference: Leviathan et. al., Fast Inference from Transformers via Speculative Decoding

Fast LLM Inference

We can optimize the “draft model” to draft tokens more likely to be accepted. The technique is called speculative decoding (SD).

The draft model can also be integrated in the target model. DeepSeek implemented this in V3, branded as multi token prediction (MTP).



Reference:

Cai et. al., Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads

DeepSeek AI, DeepSeek-V3 Technical Report

Wu et. al., TETRIS: Optimal Draft Token Selection for Batch Speculative Decoding

Table of Content

- Some Fundamentals
- LLM Inference
- Reasoning

Reasoning

define reasoning ability

Reasoning ability is the mental capacity to:

1. **Process information logically** – understand facts or premises and see how they fit together.
2. **Draw valid inferences** – move from what is known to what must (or is likely to) be true.
3. **Solve problems and make decisions** – generate, evaluate, and choose among solutions.
4. **Adapt thinking to new contexts** – transfer what's been learned to unfamiliar situations, spot patterns, and revise conclusions when presented with new evidence.

In short, it's the cognitive toolkit that lets us move beyond raw data to coherent understanding, judgment, and action.

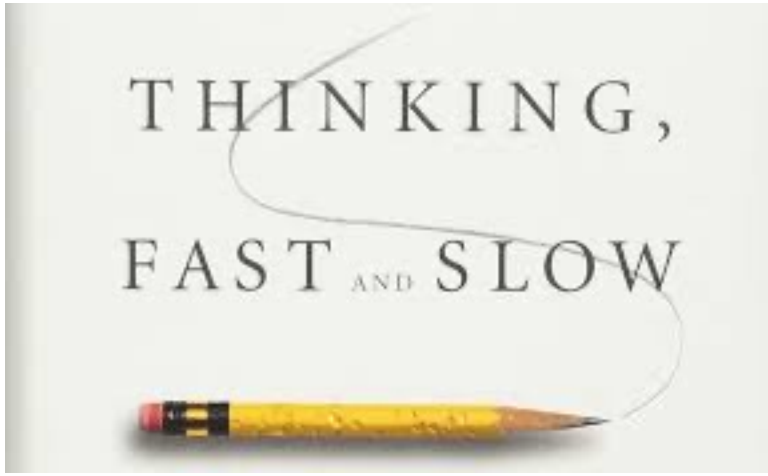
Reference: https://chat.openai.com/

Reasoning

Text is a perfect tool for formulating reasoning:

That is what most essays, books, and papers are written for!

Reasoning



Two systems of thinking:

System 1: fast, intuitive, effortless

- Good for informational, pattern recognition, etc..

System 2: deliberate, analytic, effortful

- Good for heavy thinking, arithmetic, planning, etc..

Reference: Daniel Kahneman, “Thinking, Fast and Slow”

Reasoning

The naïve, auto-regressive, next token generation is like System 1.

How to make LLM perform System 2 thinking?

Two Kinds of Reasoning

- Without training

- With training

Reasoning Without Training

LLMs have an "innate" ability to perform System 2 thinking.

Intuition:

logical reasoning can be mimicked.

Text already embodies logic.

The MacLaurin series:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots$$

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!} = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots$$

Substitute $z = ix$ in the last series:

$$e^{ix} = \sum_{n=0}^{\infty} \frac{(ix)^n}{n!} = 1 + ix + \frac{(ix)^2}{2!} + \frac{(ix)^3}{3!} + \cdots$$

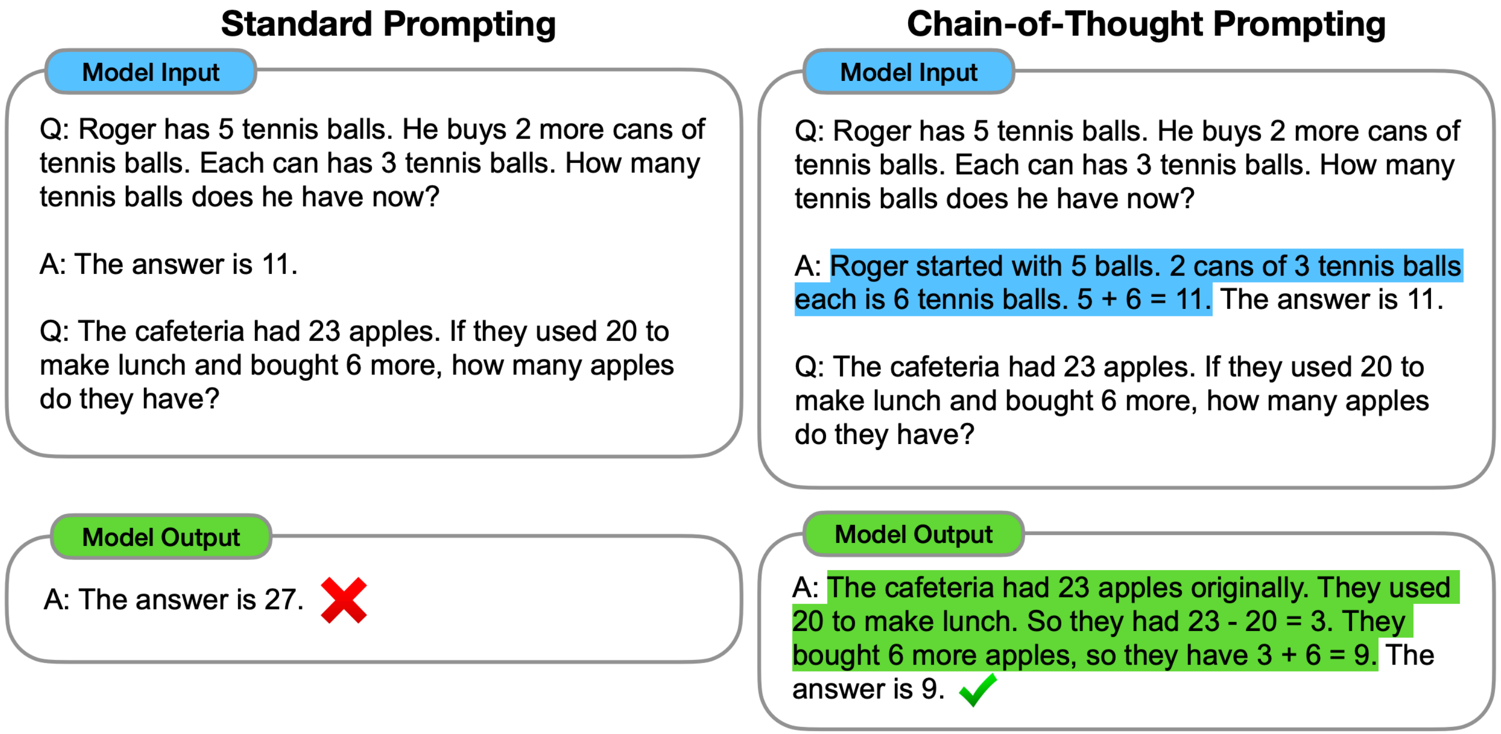
$$= 1 + ix - \frac{x^2}{2!} - i\frac{x^3}{3!} + \frac{x^4}{4!} + i\frac{x^5}{5!} - \cdots$$

$$= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + i\left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots\right)$$

$$= \cos x + i \sin x$$

Reasoning Without Training

Chain-of-Thought (CoT): give a few demonstrations of reasoning traces.



Reference: Wei et. al., Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Reasoning Without Training

Or even simpler: just tell the LLM to do it!

"Let's think step by step"

(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. ✓

(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. ✓

Reference: Kojima et. al., Large Language Models are Zero-Shot Reasoners

Reasoning Without Training

Or you can make the "step by step" explicit by calling the LLM multiple times.

What is the cube of square root of 16? Think step by step.

I know that the square root of 16 is 4.

Continue your thoughts. If you have the answer, start with Answer:

Now I know the square root of 16 is 4. The cube of 4 is 64.

Continue your thoughts. If you have the answer, start with Answer:

Answer: 64

Reasoning Without Training

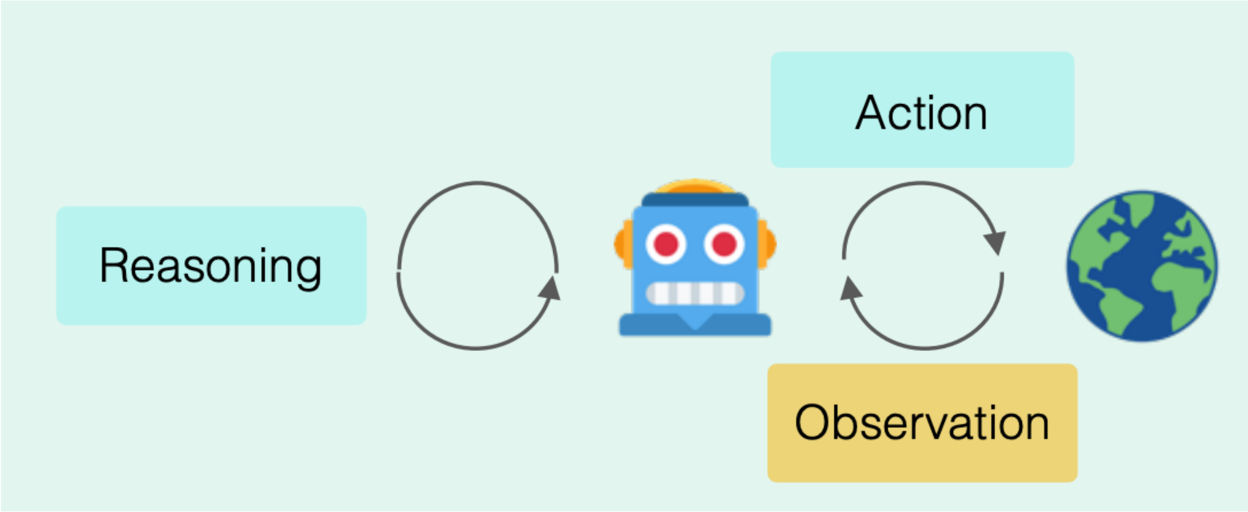
Mens et manus. (translated to “Mind and hand”)

- Motto of MIT

Reasoning Without Training

Thinking along is not meaningful, not must be combined with actions.

We can make the LLM think and interact with the external world.



Reference: https://llmagents-learning.org/slides/llm_agent_history.pdf

Reasoning Without Training

Who's the wife of the current president of the US?

Let me see. To answer this, I need to first know who is the current president of the US.

Web Search Result: the current president of the US is Donald Jr. Trump.

Ok! Now, I need to search for the wife of Donald Jr. Trump.

Web Search Result: the wife of Donald Jr. Trump is Melania Trump.

Answer: Melania Trump.

Reference: Yao et. Al., ReAct: Synergizing Reasoning and Acting in Language Models

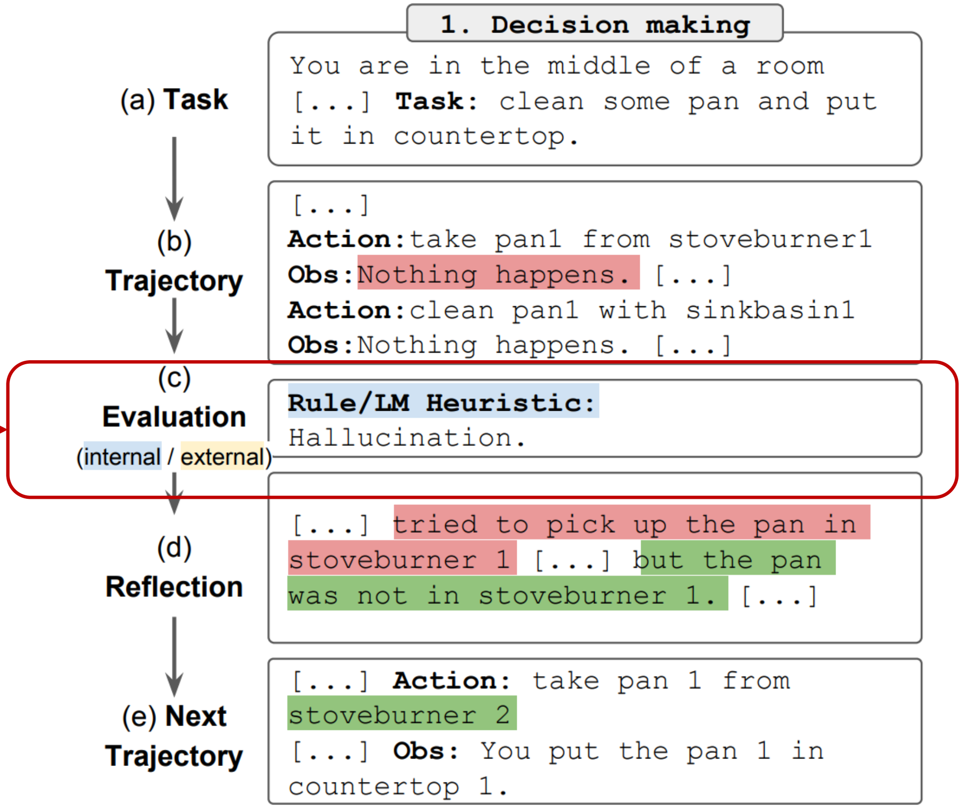
Reasoning Without Training

Other common reasoning processes are **reflection** and **self-consistency.**

Reasoning Without Training

Reflection: Having a buddy to critique what you are doing

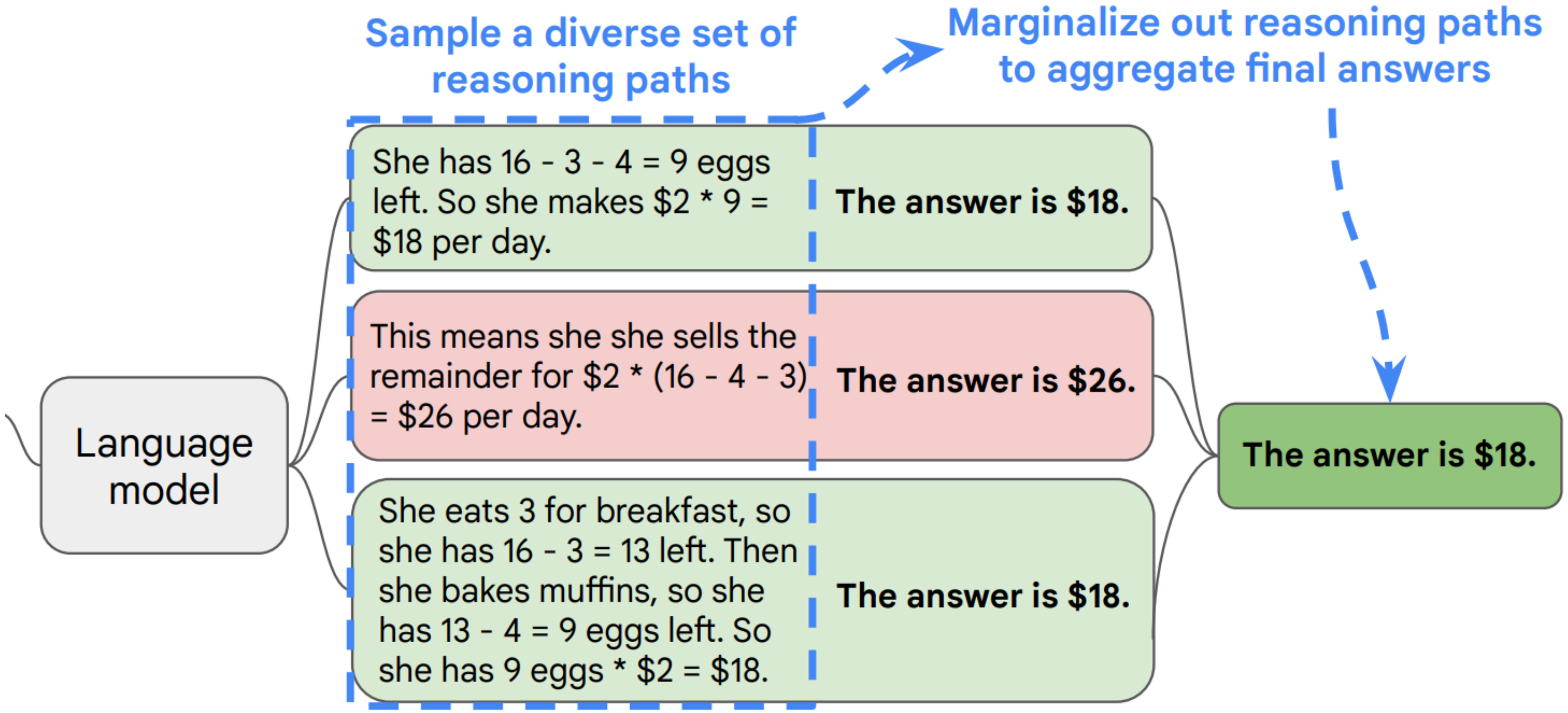
A "buddy" LLM



Reference: Shinn et. al., Reflextion: Language Agents with Verbal Reinforcement Learning

Reasoning Without Training

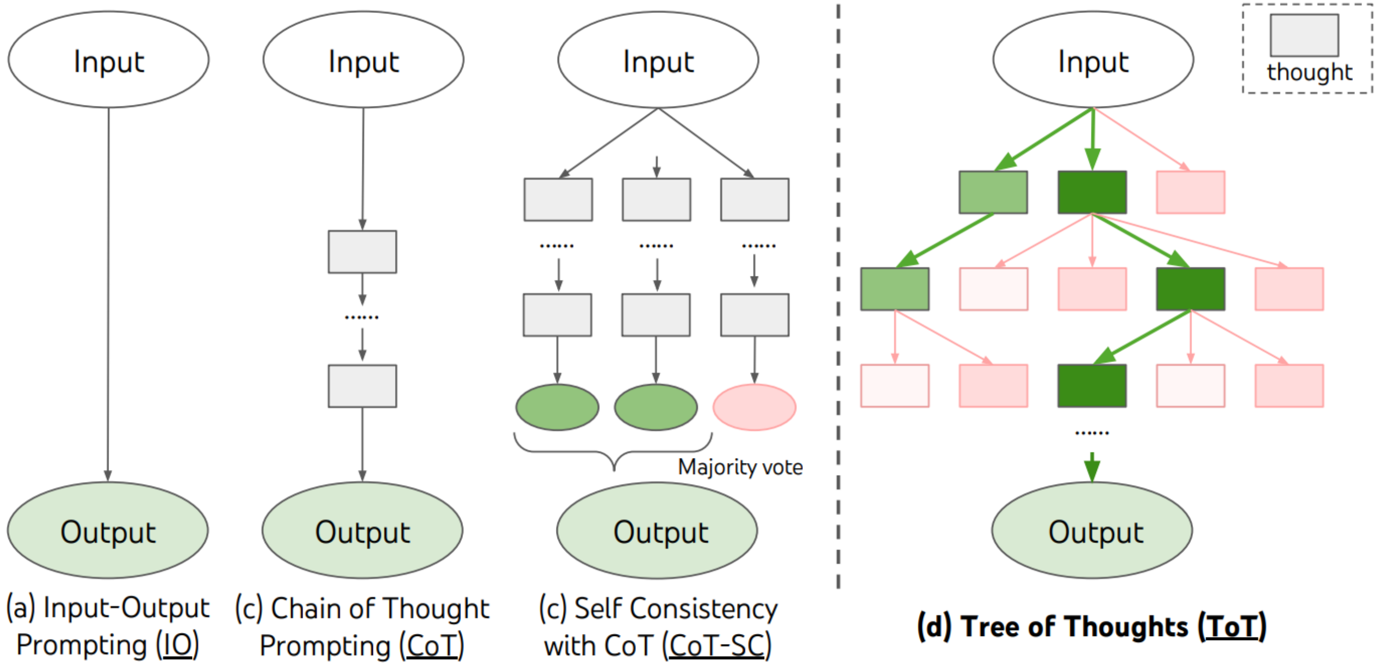
Self-consistency: think about something from different perspectives, and pick the majority decision.



Reference: Wang et. al., Self-Consistency Improves Chain of Thought Reasoning in Language Models

Reasoning Without Training

Tree-of-Thought (ToT): a conversation-level version of beam search. Each time, the LLM generates multiple candidates and only keeps the most promising ones (evaluated by a separate "buddy" model).



Reference: Yao et. al., Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Reasoning Without Training

Monte Carlo Tree Search (MCTS): similar to ToT but theoretically more robust.

MCTS balances **search cost** and **search quality**.

At each node, MCTS selects the next node that maximizes the Upper Confidence Bound (UCB) value for traversal.

Reference:

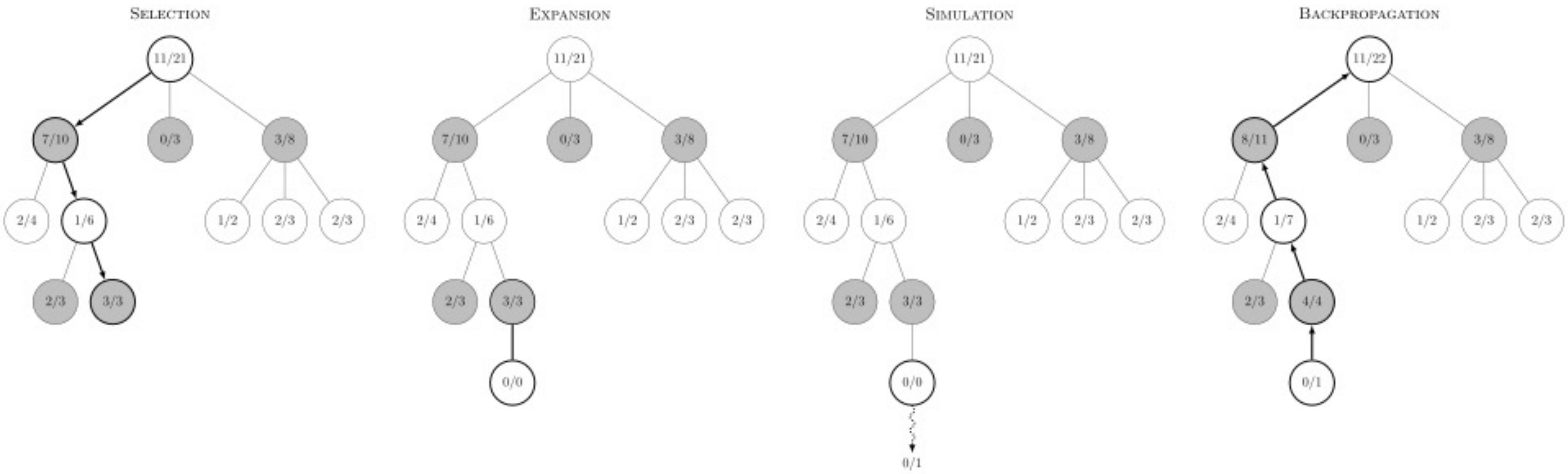
Wang et. al., Towards Self-Improvement of LLMs via MCTS: Leveraging Stepwise Knowledge with Curriculum Preference Learning

Xie et. al., Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning

Li etl. al., FastMCTS: A Simple Sampling Strategy for Data Synthesis

Reasoning Without Training

Monte Carlo Tree Search (MCTS): similar to ToT but theoretically more robust.

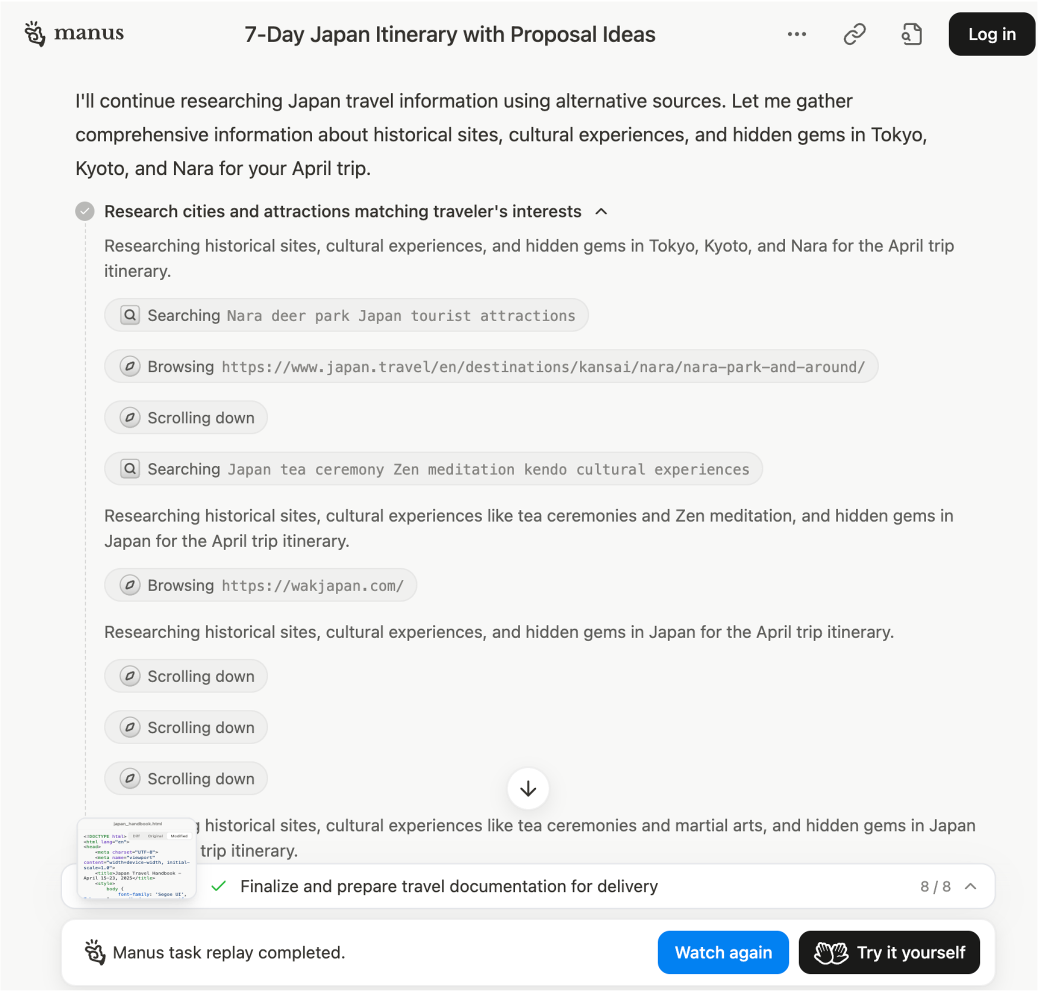


Reference: https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

Reasoning Without Training

Deep Research: A production-ready implementation of the aforementioned methods.

E.g., Interleaved thinking, and web browsing actions.



Reference: https://manus.im/share/brWKUSp51ItvVMBpcXNCZ1?replay=1

Two Kinds of Reasoning

- Without training
- With training

Trained Reasoning

The previously demonstrated reasoning rely on the LLM’s innate ability to reason.

For stronger reasoning, **practice** is required.

Trained Reasoning

LLMs can be trained to think harder.

How? Let the LLM do difficult math problems. Reward it for correct answers and punish it for wrong answers.

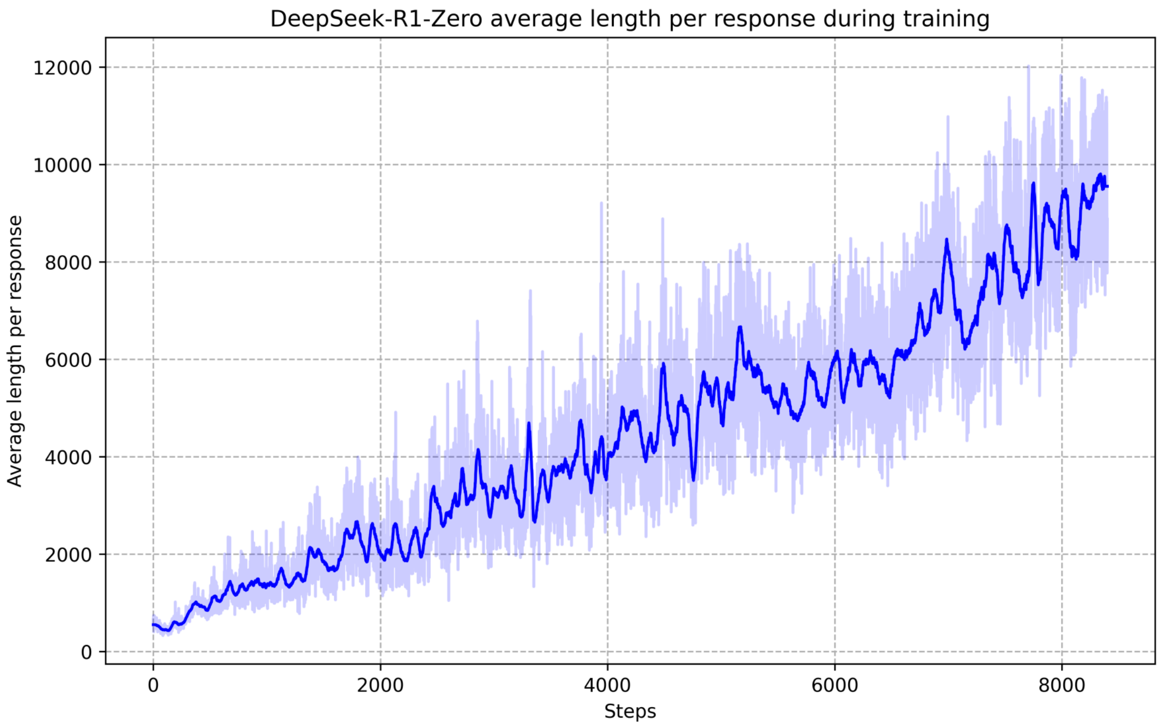
In the prompt, instruct the LLM to reason before answer.

> A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:

Reference: DeepSeek-AI, DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

Trained Reasoning

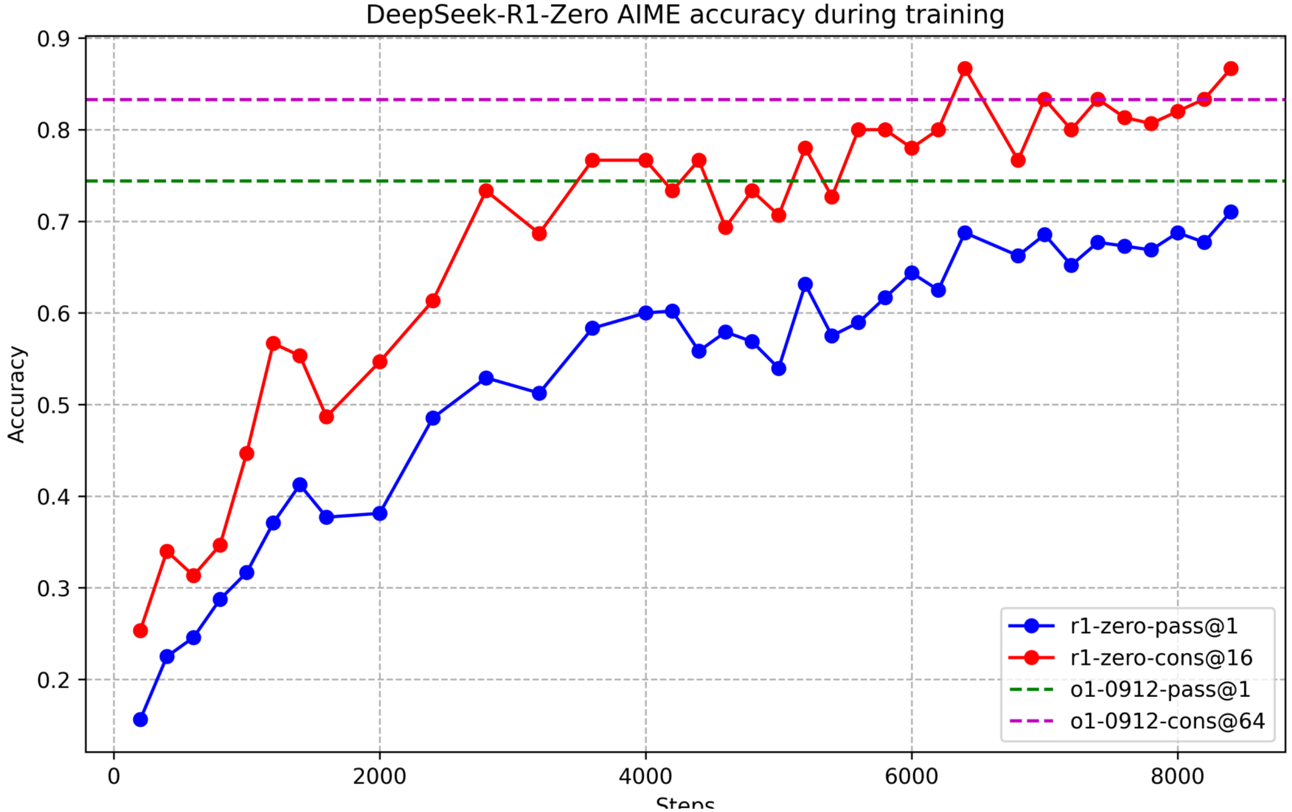
As the LLM is trained, the thinking process gets more and more complex.



Reference: DeepSeek-AI, DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

Trained Reasoning

And the performance gets better and better as well.



Reference: DeepSeek-AI, DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

Trained Reasoning

Now, the thinking process can be trained. Can we also train the actions part (the "Manus")?

Yes. Change the prompt and use a similar method to train the model.

> Answer the given question. You must conduct reasoning inside <think> and </think> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <search> query </search>, and it will return the top searched results between <information> and </information>. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer> xxx </answer>. Question: question.

Reference: Jin et. al., Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning

Trained Reasoning

And it works.

Methods	General QA			Multi-Hop QA				
	NQ†	TriviaQA⋆	PopQA⋆	HotpotQA†	2wiki⋆	Musique⋆	Bamboogle⋆	Avg.
Qwen2.5-7b-Base/Instruct								
Direct Inference	0.134	0.408	0.140	0.183	0.250	0.031	0.120	0.181
CoT	0.048	0.185	0.054	0.092	0.111	0.022	0.232	0.106
IRCoT	0.224	0.478	0.301	0.133	0.149	0.072	0.224	0.239
Search-o1	0.151	0.443	0.131	0.187	0.176	0.058	0.296	0.206
RAG	0.349	0.585	0.392	0.299	0.235	0.058	0.208	0.304
SFT	0.318	0.354	0.121	0.217	0.259	0.066	0.112	0.207
R1-base	0.297	0.539	0.202	0.242	0.273	0.083	0.296	0.276
R1-instruct	0.270	0.537	0.199	0.237	0.292	0.072	0.293	0.271
Search-R1-base	**0.480**	**0.638**	**0.457**	**0.433**	0.382	**0.196**	**0.432**	**0.431**
Search-R1-instruct	0.393	0.610	0.397	0.370	**0.414**	0.146	0.368	0.385

Reference: Jin et. al., Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning

Reasoning

Mathematically, what is reasoning for LLM?

Inference-time Scaling

Down to its core, all these reasoning capabilities emerge by making the LLM **generate more tokens** before finalizing the answer.

$$P(Y \mid X) \;=\; \prod_{i=1}^{N} P\bigl(y_i \;\mid\; y_{<i},\, X\bigr).$$

From a statistical point of view, the extra "reasoning" tokens shift the output distribution $P(Y|X)$, making it more "accurate".

Reference: Snell et. al., Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Inference-time Scaling

A trade-off between time and space.

Inference-time scaling: longer time, but smaller model size.

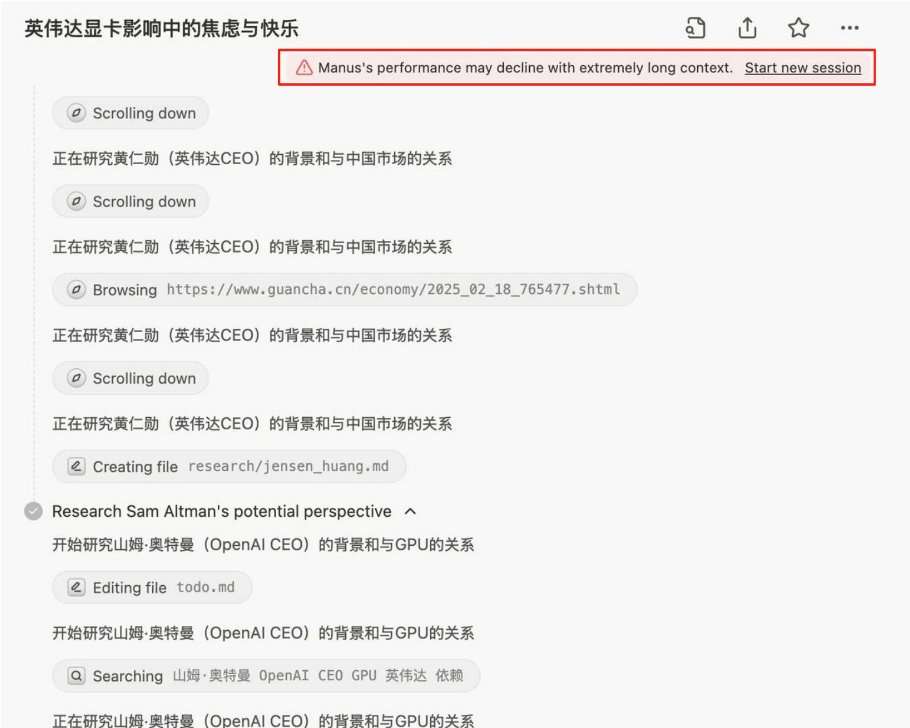
Parameter scaling: relatively shorter time, but larger model size.

Reference: Snell et. al., Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Inference-time Scaling

Another problem with inference-time scaling: the growing context size.

Everytime new information is gathered or new thinking is generated, the context grows.



Inference-time Scaling

Potential Solutions:

1. Training-free: Break the task into sub-tasks and assign them to sub-agents. Consolidate the context from time to time using external agents.

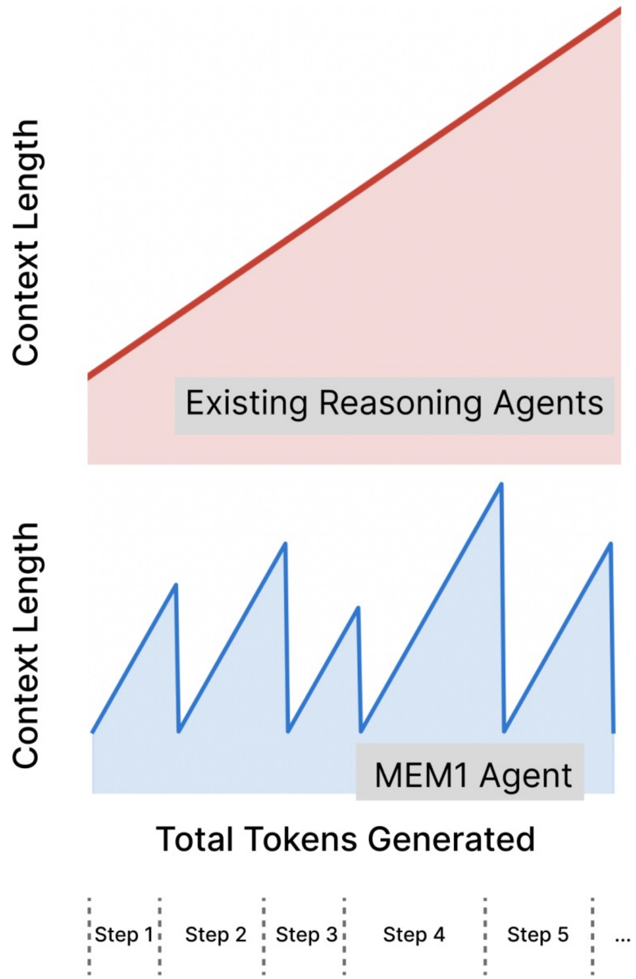
☹ Very tedious engineering work and probably does not provide satisfactory performance

Inference-time Scaling

Potential Solutions:

2. Trained: Dynamically compress the memory in the generation process and train it to enhance its consolidation ability.

More works need to be done for production-level reasoning tasks (e.g., deep research).



Reference: Zhou et. al., MEM1: Learning to Synergize Memory and Reasoning for Efficient Long-Horizon Agents

Inference-time Scaling

Efficiency: while models can improve problem solving with reasoning. The reasoning trace can be messy and redundant.

Imagine writing 10 pages long just to solve a linear equation!

Inference-time Scaling

Curbing overthinking:

Practical no-training methods:

- Prompt the model to be aware ("You must think for no more than 100 words")
- Force stop the generation (After 100 tokens, append a "You must output your answer now!" to the end of the output stream)

Reference: Sui et. al., Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models

Inference-time Scaling

Curbing overthinking:

Training-based methods:

- Gather high-quality reasoning traces and fine-tune the model

- Train the model to generate shorter reasoning by rewarding short traces and penalizing long traces

Reference: Sui et. al., Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models

Inference-time Scaling

Curbing overthinking in use

```
curl https://api.anthropic.com/v1/messages \
     --header "x-api-key: $ANTHROPIC_API_KEY" \
     --header "anthropic-version: 2023-06-01" \
     --header "content-type: application/json" \
     --data \
'{
    "model": "claude-sonnet-4-20250514",
    "max_tokens": 16000,
    "thinking": {
        "type": "enabled",
        "budget_tokens": 10000
    },
    "messages": [
        {
            "role": "user",
            "content": "Are there an infinite number of prime numbers such that n m
        }
    ]
}'
```

Reference: https://docs.anthropic.com/en/docs/build-with-claude/extended-thinking

Q & A

Quiz

1. Many LLM inference service providers (e.g., vLLM) offer the ability to make LLM generate “structured output”. How is it implemented?

Code

```python
from openai import OpenAI
client = OpenAI(
    base_url="http://localhost:8000/v1",
    api_key="-",
)
model = client.models.list().data[0].id

completion = client.chat.completions.create(
    model=model,
    messages=[
        {"role": "user", "content": "Classify this sentiment: vLLM is wonderful!"}
    ],
    extra_body={"guided_choice": ["positive", "negative"]},
)
print(completion.choices[0].message.content)
```

Reference: https://docs.vllm.ai/en/latest/features/structured_outputs.html#online-serving-openai-api

Quiz

2. In the slides, we mentioned that an LLM can be trained to generated longer and longer reasoning content. Can you achieve this long reasoning trace without training the LLM?