Large Language Models III

Philipp Kraehenbuehl

LLMs

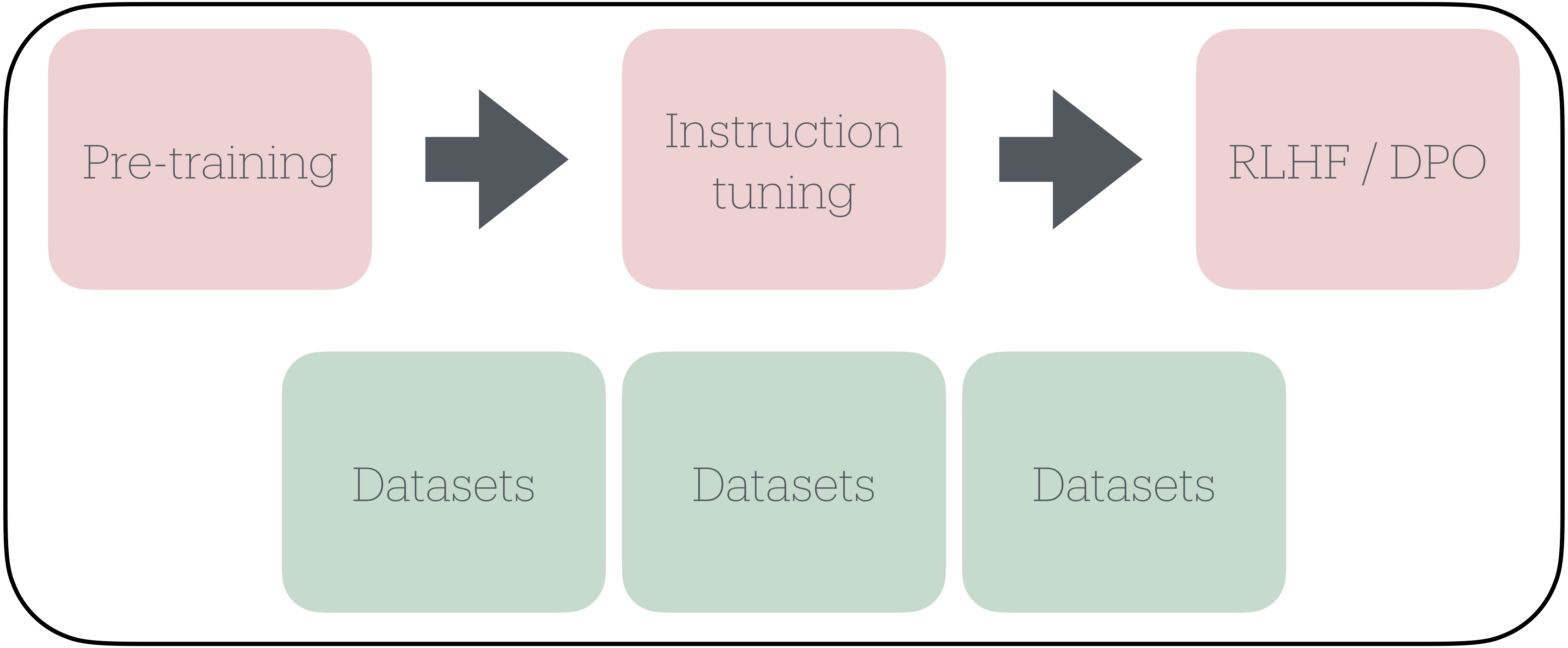
- Architectures
- Generation, Instruction Tuning, RLHF, DPO, Tasks and Datasets
- Tool use and Structured Outputs
- Long Context and RAGs
- Structured Dialogues, Reflection
- Limitations of LLMs

Long Context

Philipp Krähenbühl, UT Austin

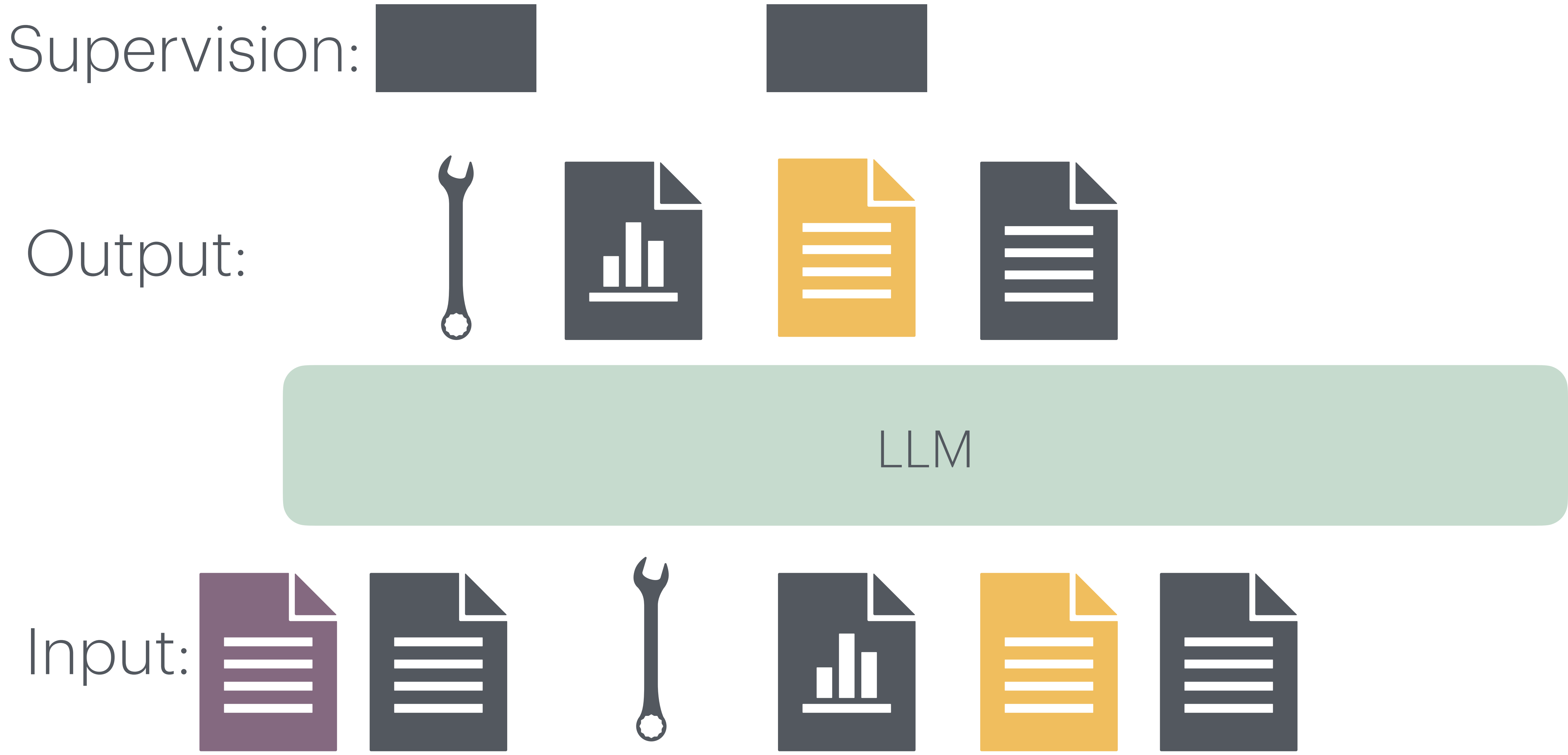
Full Picture

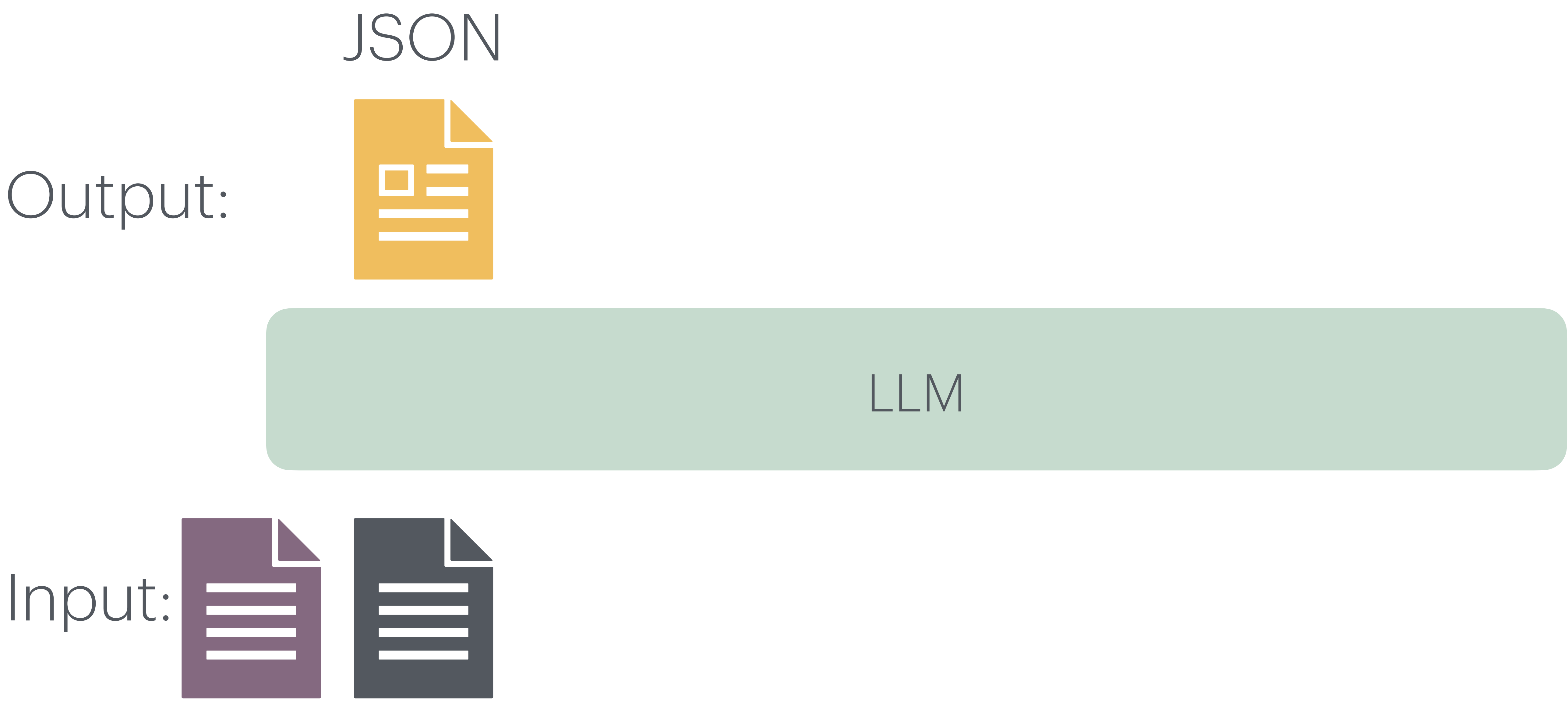
Basic LLM



Tools and Structured outputs

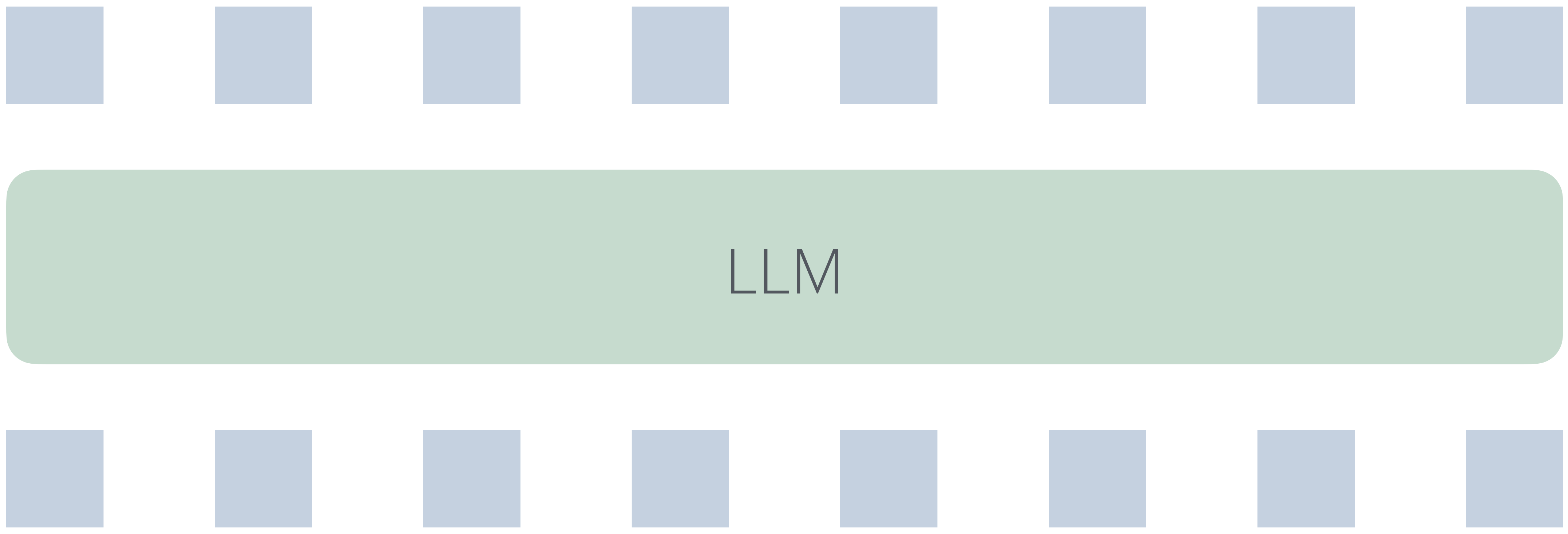
- Tools
  - Special tags, Special chat-template
- Structured output
  - Option 1.1: Write a robust parser (in python)
    - Let LLM know that you failed to parse
  - Option 1.2: Constrain output
  - Option 2: Use a tool, arguments = json fields





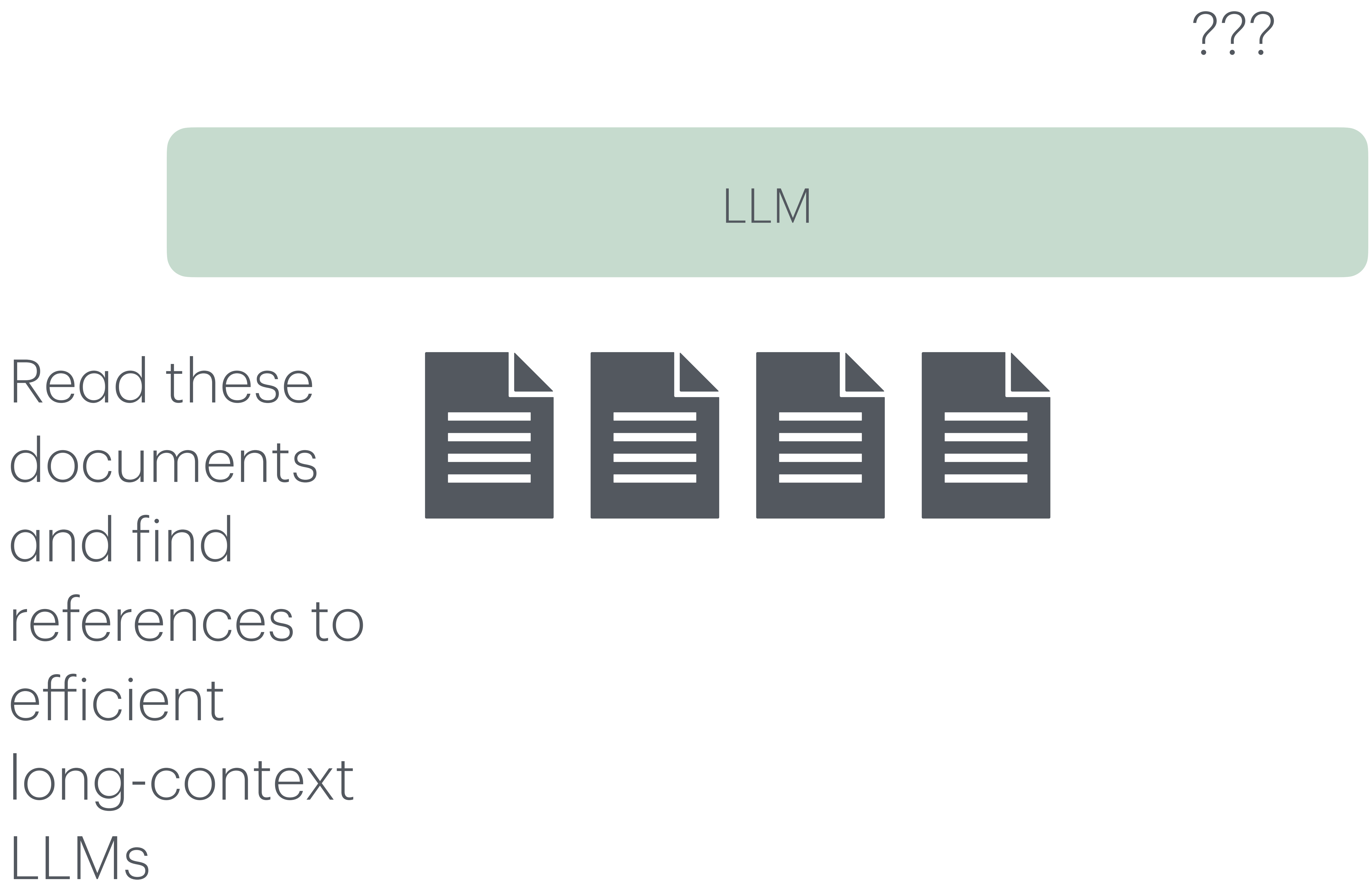
Long Context

- Current model are **pre-trained** on **2-8k** token sequences



Long Context

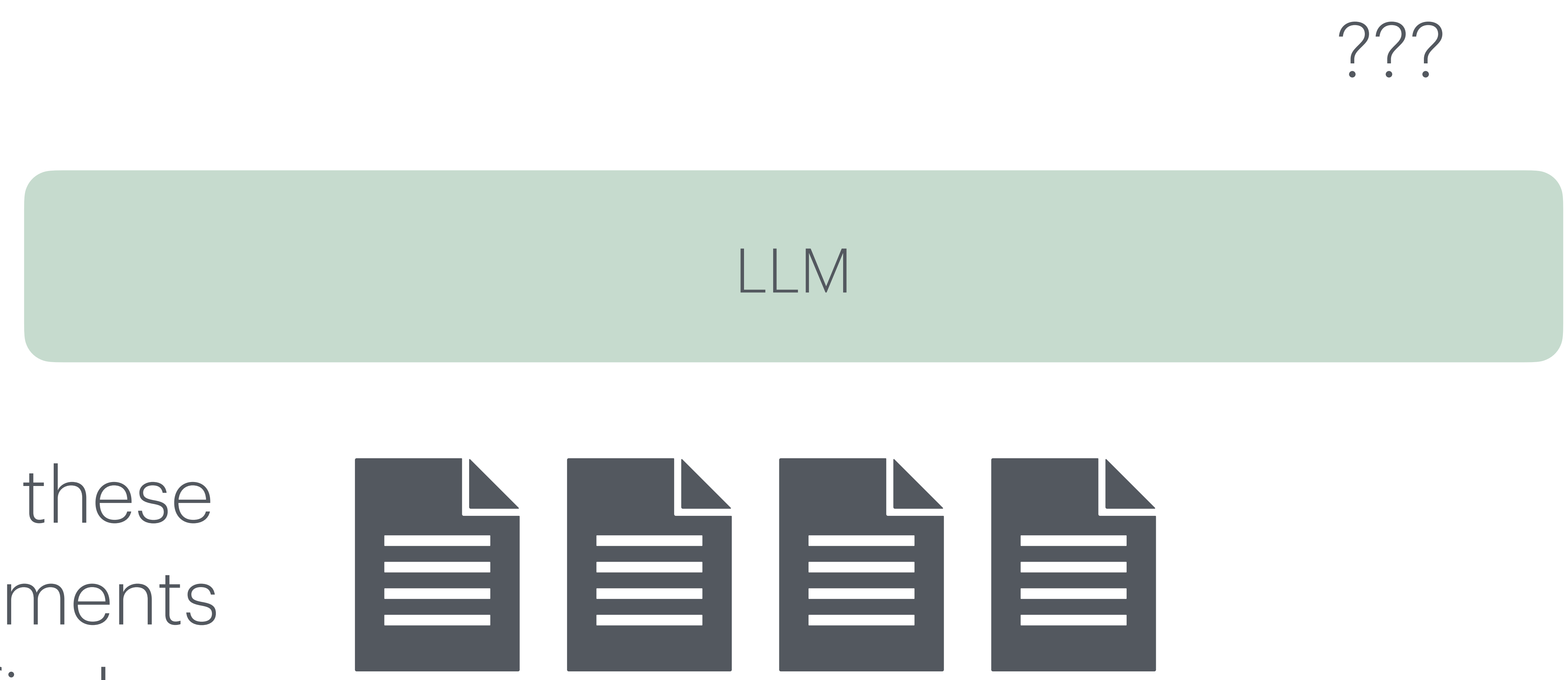
What happens if we feed ten's of thousands of tokens into an LLM?



Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. OOM (Out Of Memory)

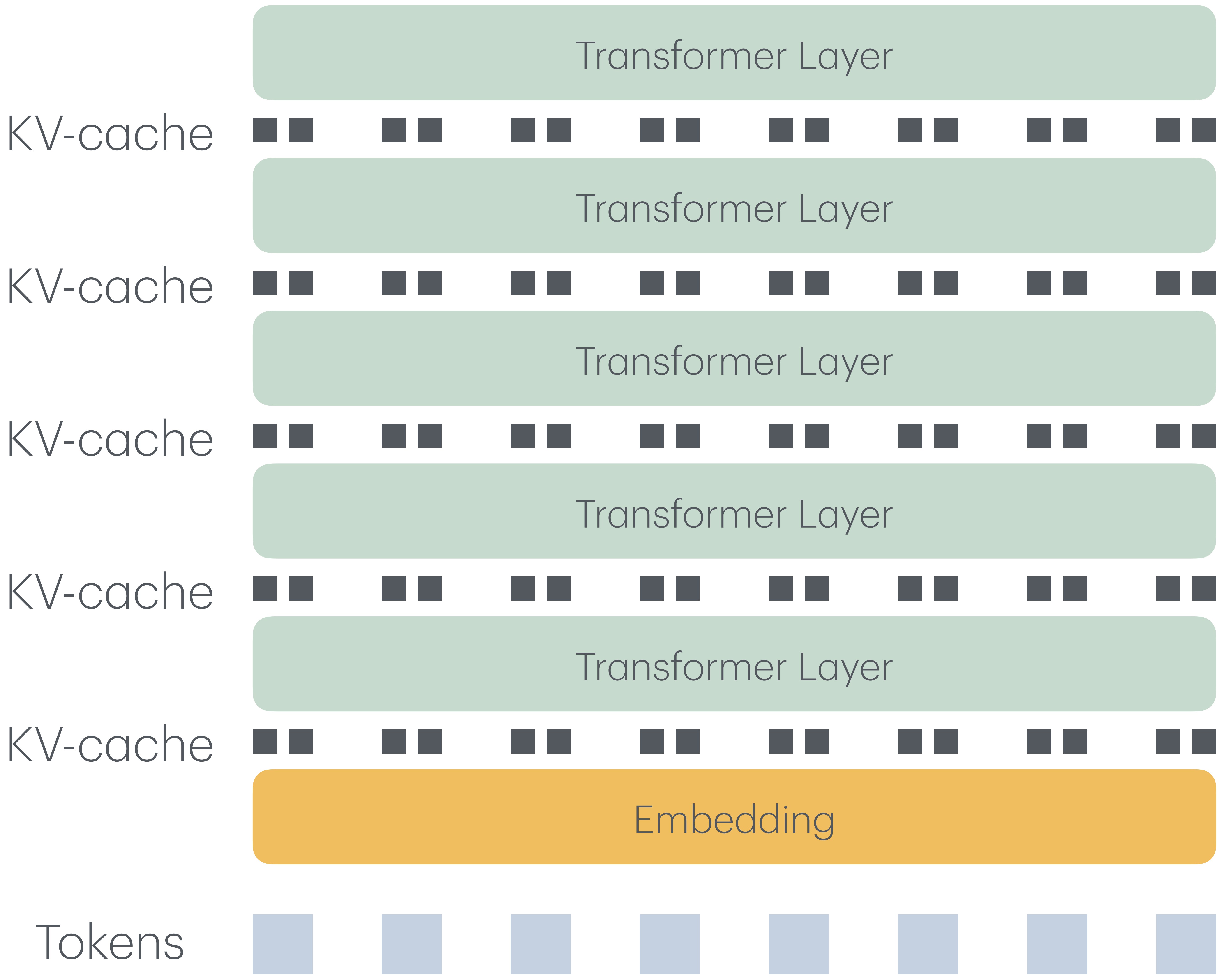


Read these documents and find references to efficient long-context LLMs

Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. OOM (Out Of Memory)

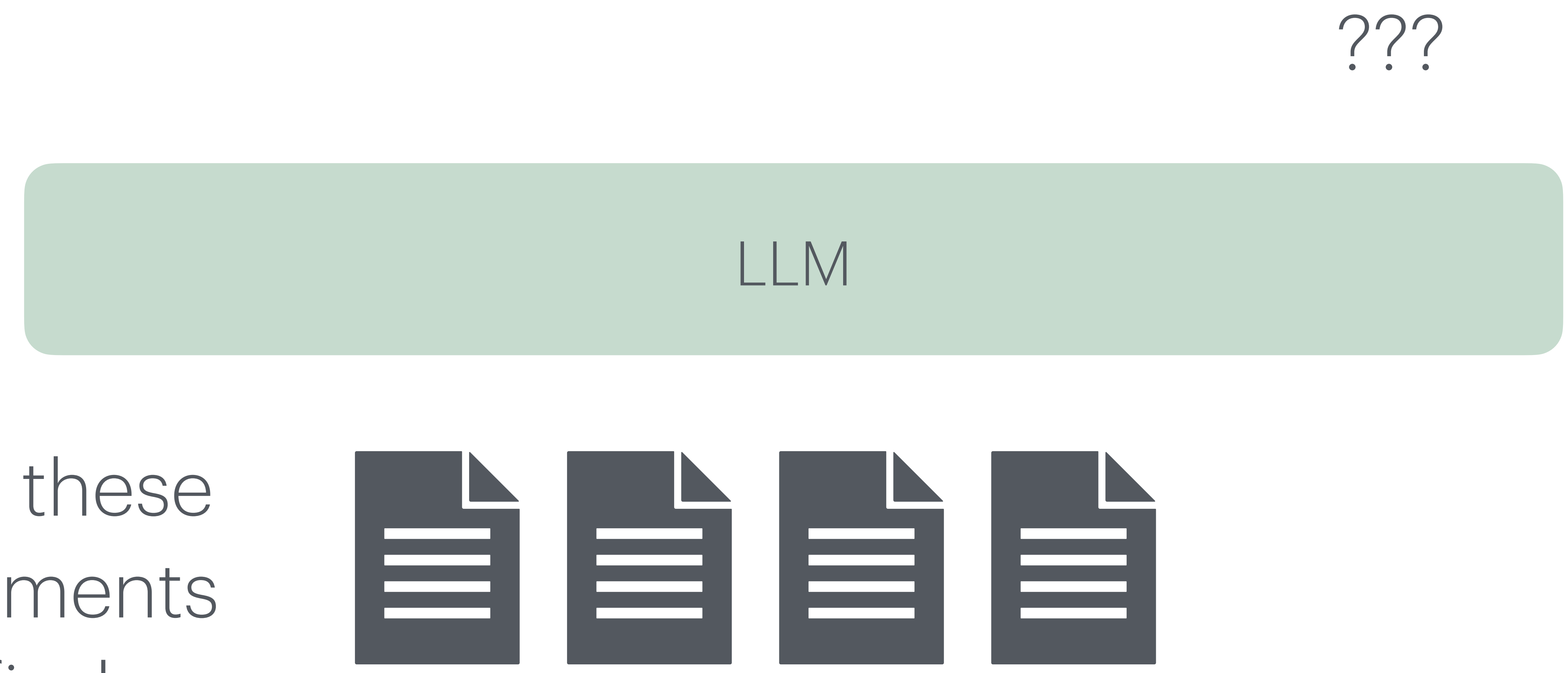


Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. OOM (Out Of Memory)

2. Model will be very slow



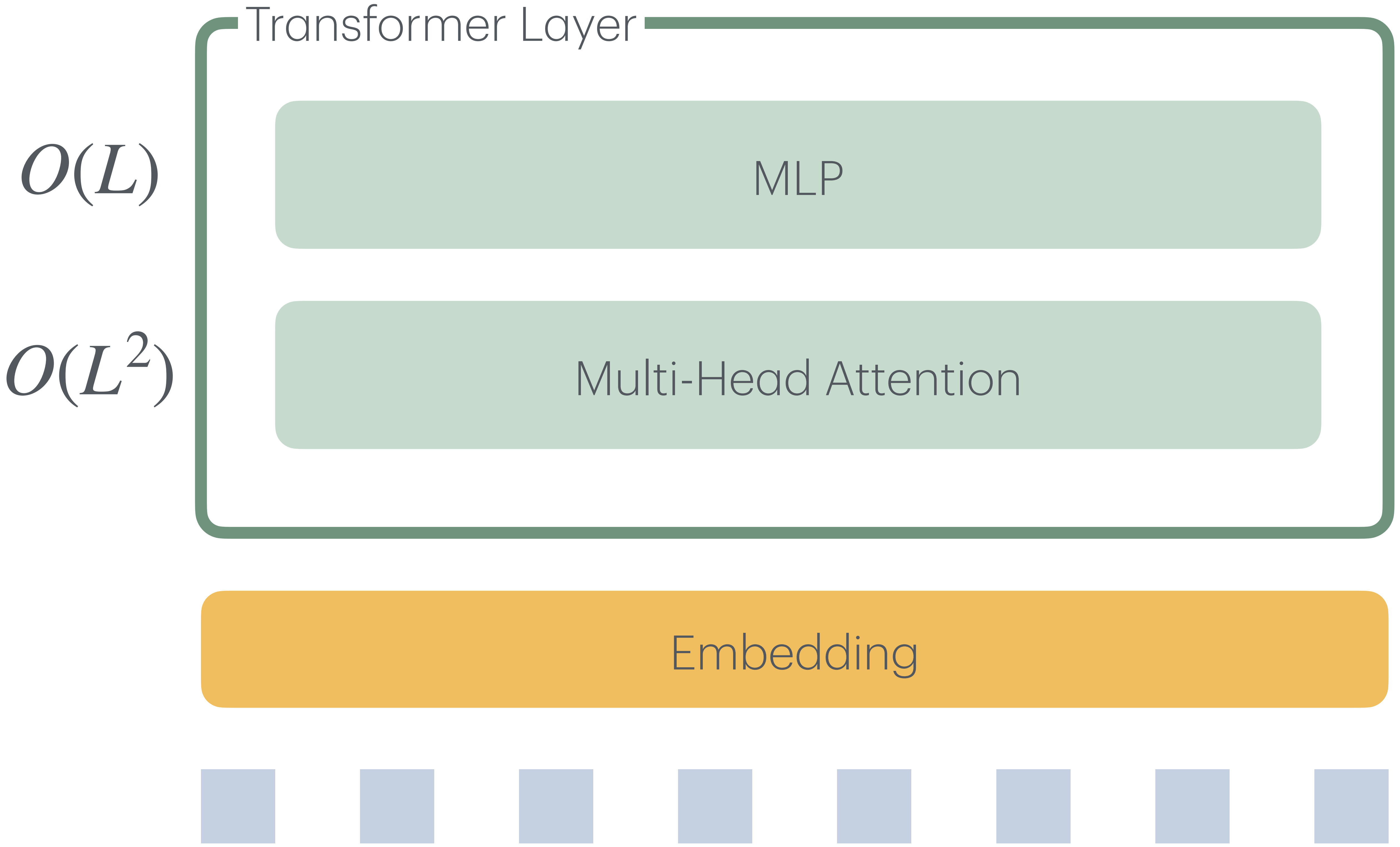
Read these documents and find references to efficient long-context LLMs

Long Context

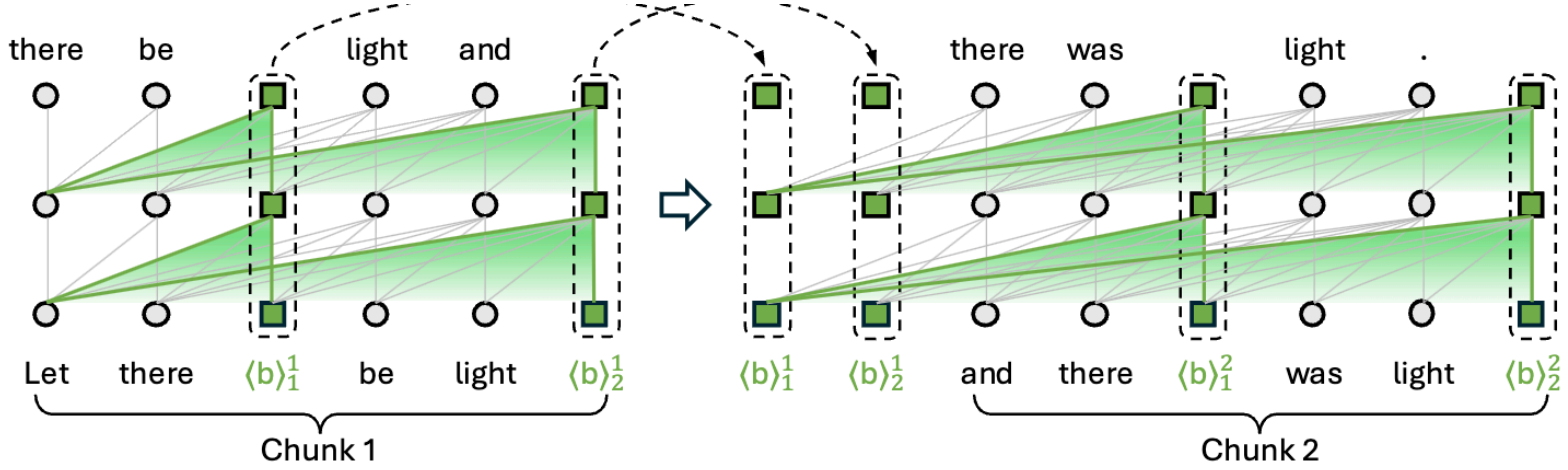
What happens if we feed ten's of thousands of tokens into an LLM?

1. OOM (Out Of Memory)

2. Model will be very slow

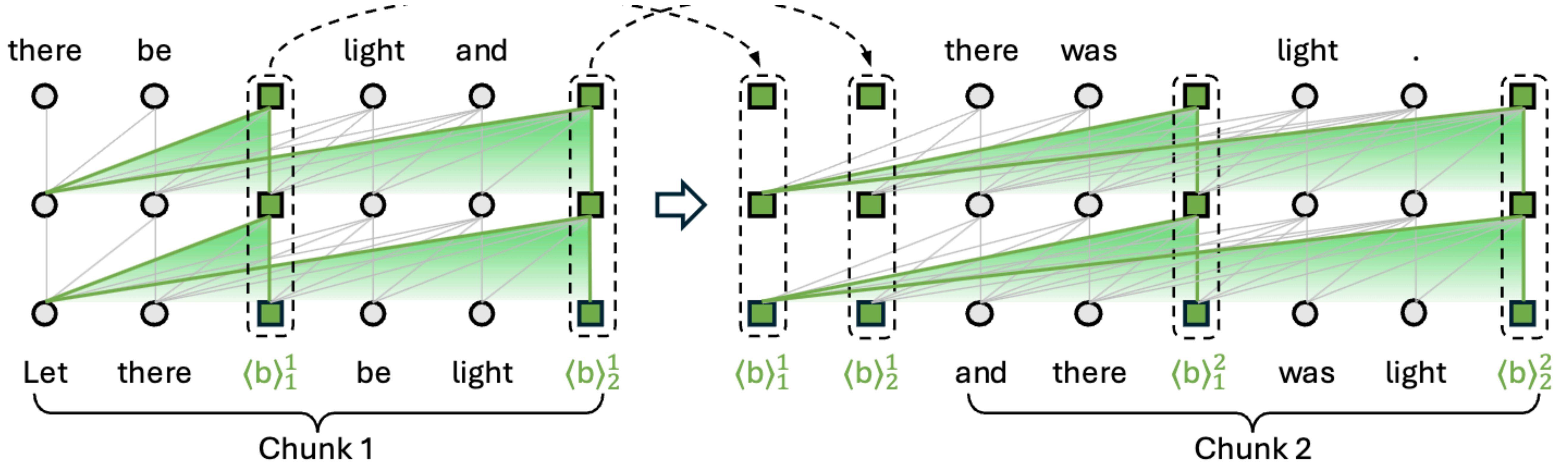


Activation Beacon

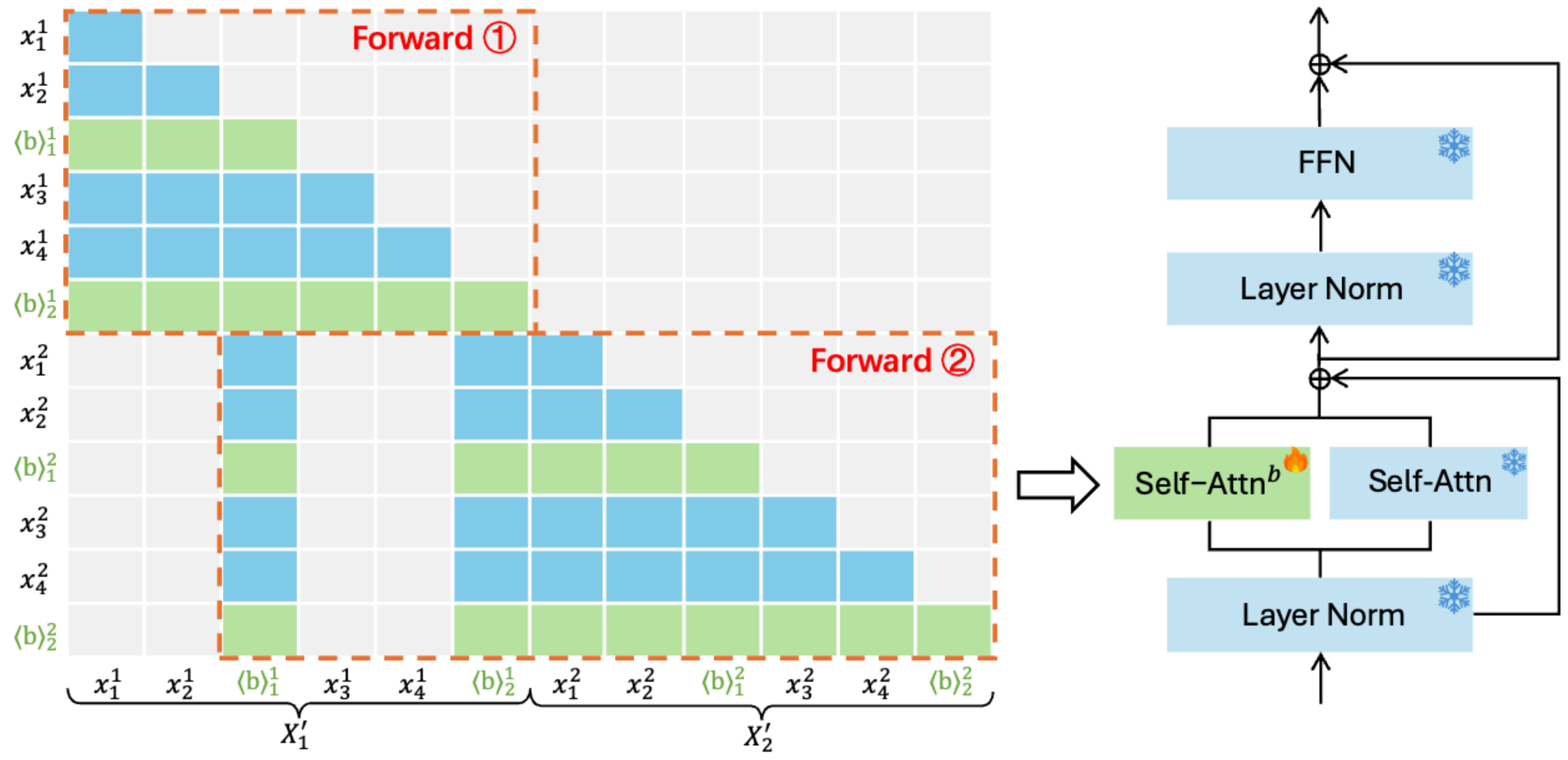


Long Context Compression with Activation Beacon, Zhang etal 2024

Activation Beacon



- Start from pre-trained model
- Partition sequence into chunks of 1024
- Pick k “beacons” per chunk
- Chunk n only sees beacons of chunks 1…n-1
- Fine-tune



Long Context Compression with Activation Beacon, Zhang etal 2024

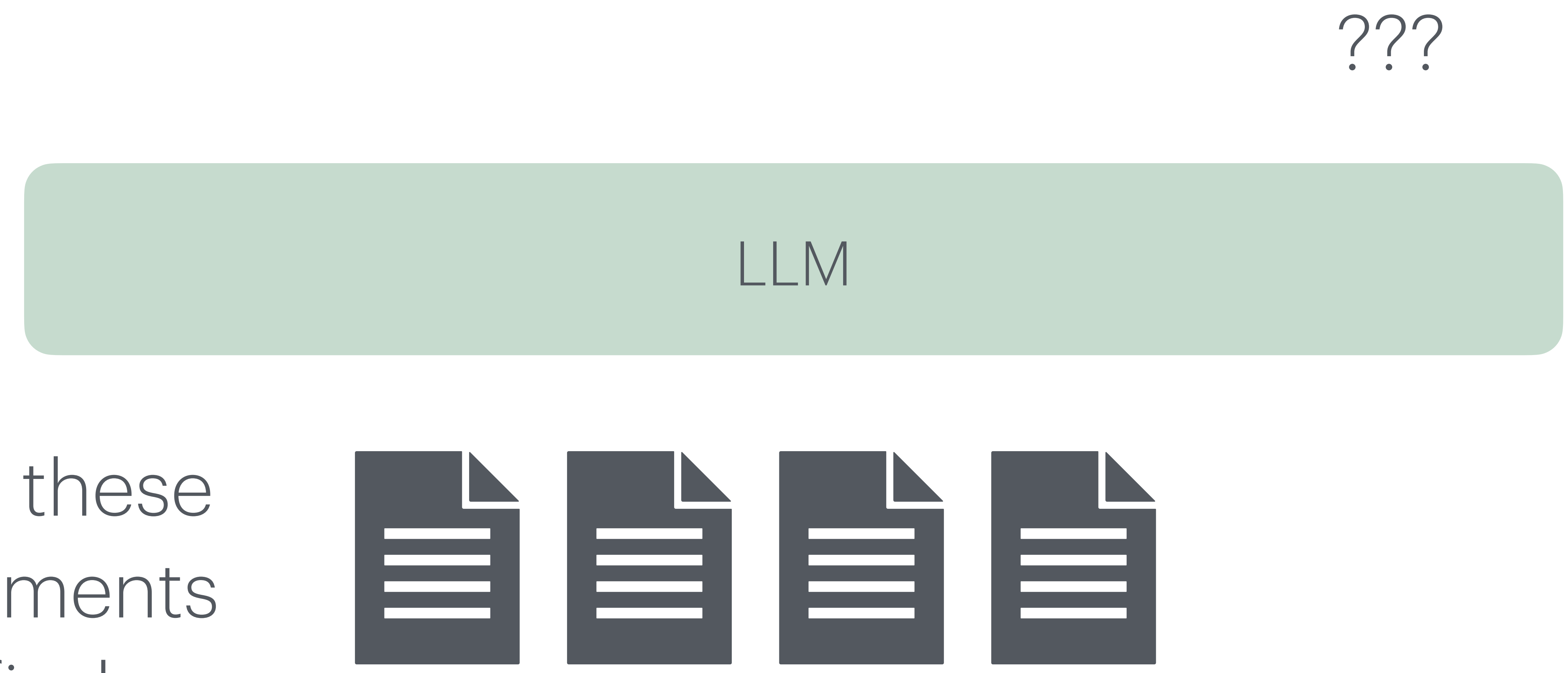
Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. ~~OOM (Out Of Memory)~~
2. ~~Model will be very slow~~

Activation Beacons and friends

Read these documents and find references to efficient long-context LLMs



Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. ~~OOM (Out Of Memory)~~
2. ~~Model will be very slow~~
3. Model will produce garbage outputs

Activation Beacons and friends

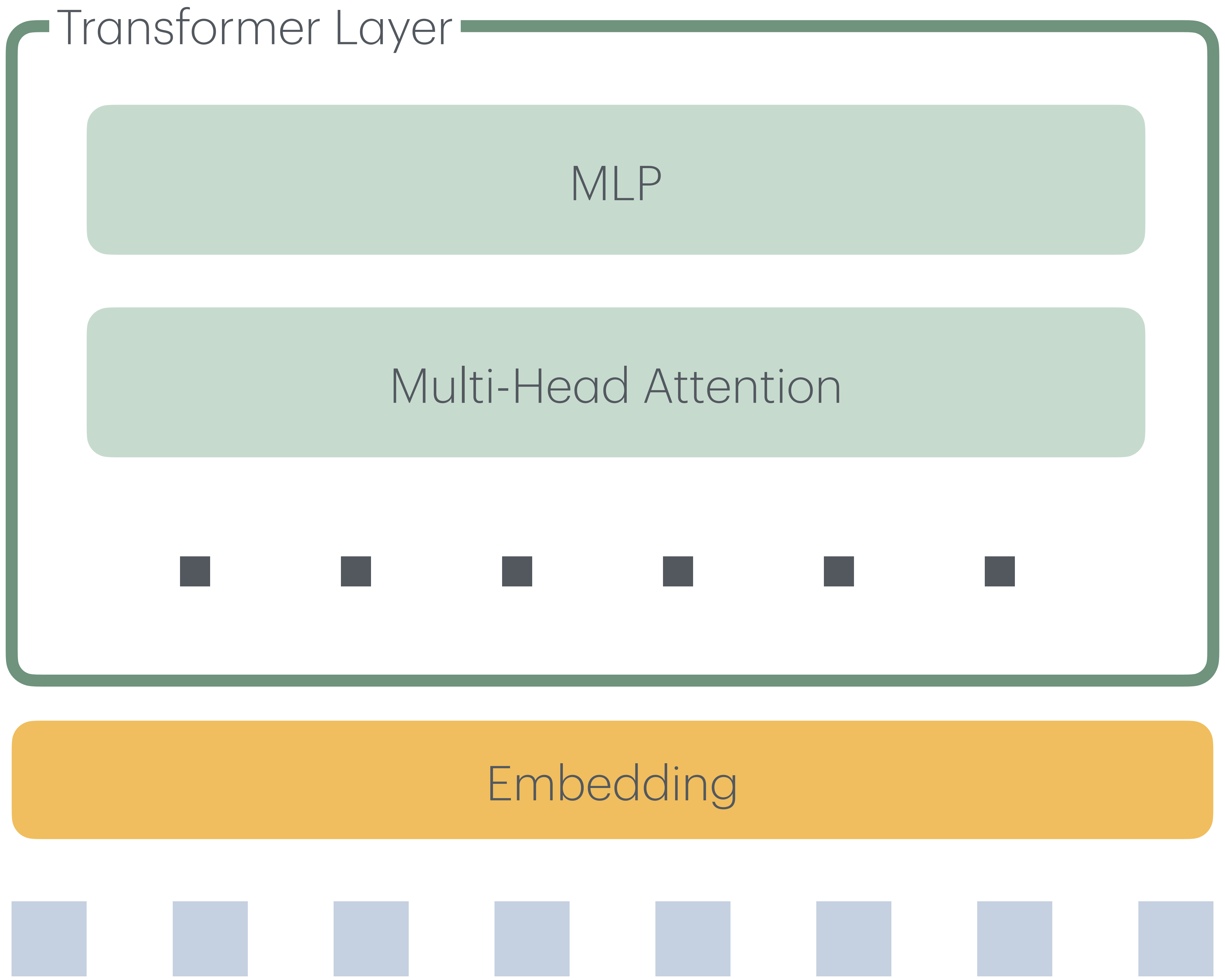
Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. ~~OOM (Out Of Memory)~~
2. ~~Model will be very slow~~
3. Model will produce garbage outputs

Activation Beacons and friends

Positional embedding



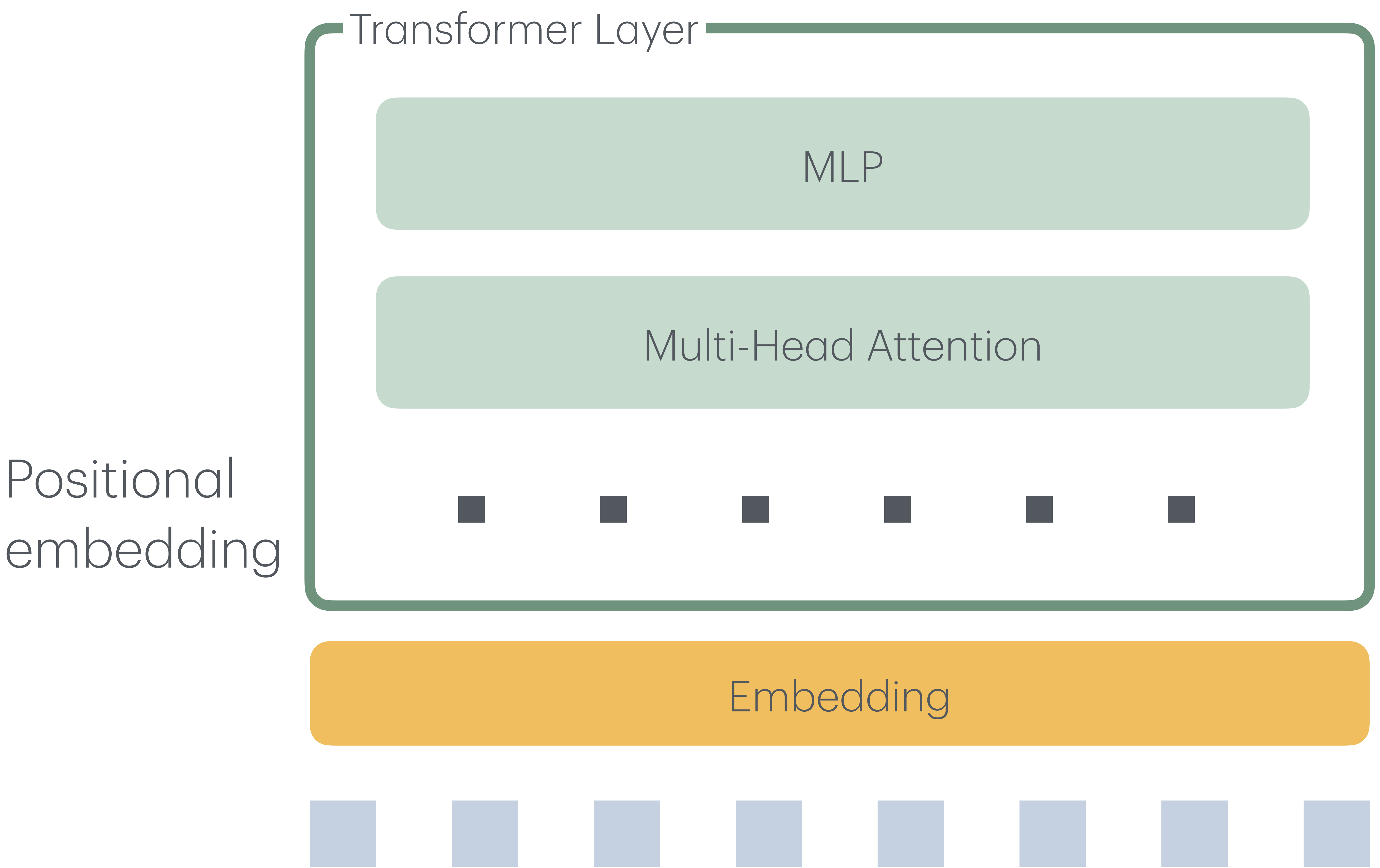
Positional Embedding

- Rotary Embeddings

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_q \boldsymbol{x}_m)^\intercal (\boldsymbol{R}^d_{\Theta,n} \boldsymbol{W}_k \boldsymbol{x}_n) = \boldsymbol{x}^\intercal \boldsymbol{W}_q R^d_{\Theta,n-m} \boldsymbol{W}_k \boldsymbol{x}_n$$



RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021

Positional Embedding

- Rotary Embeddings
- Fixed context length during training
  - Longer context for inference

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

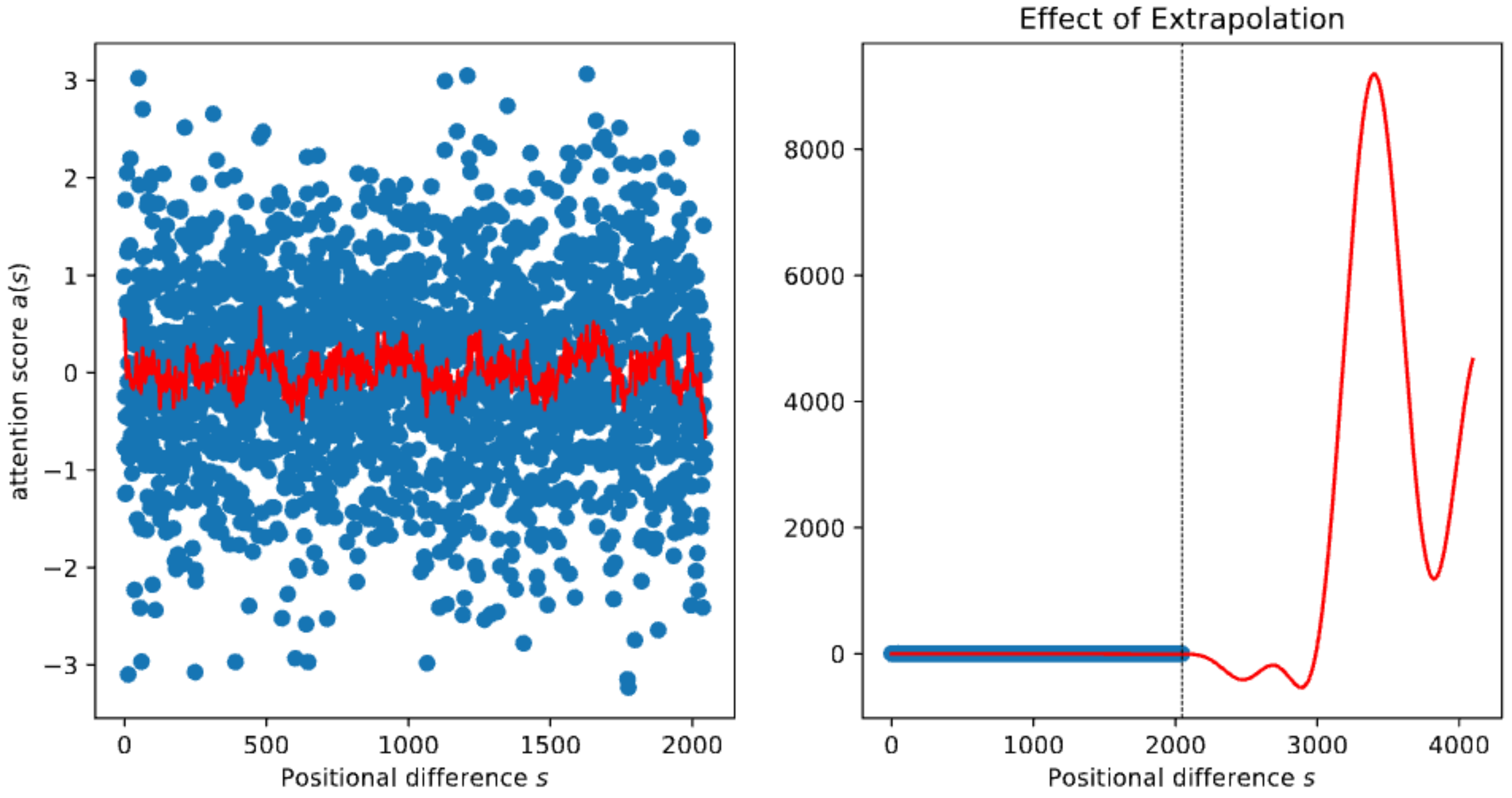
$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_q \boldsymbol{x}_m)^\intercal (\boldsymbol{R}^d_{\Theta,n} \boldsymbol{W}_k \boldsymbol{x}_n) = \boldsymbol{x}^\intercal \boldsymbol{W}_q R^d_{\Theta,n-m} \boldsymbol{W}_k \boldsymbol{x}_n$$

RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021
Extending Context Window of Large Language Models via Positional Interpolation, Chen etal 2023

Positional Embedding

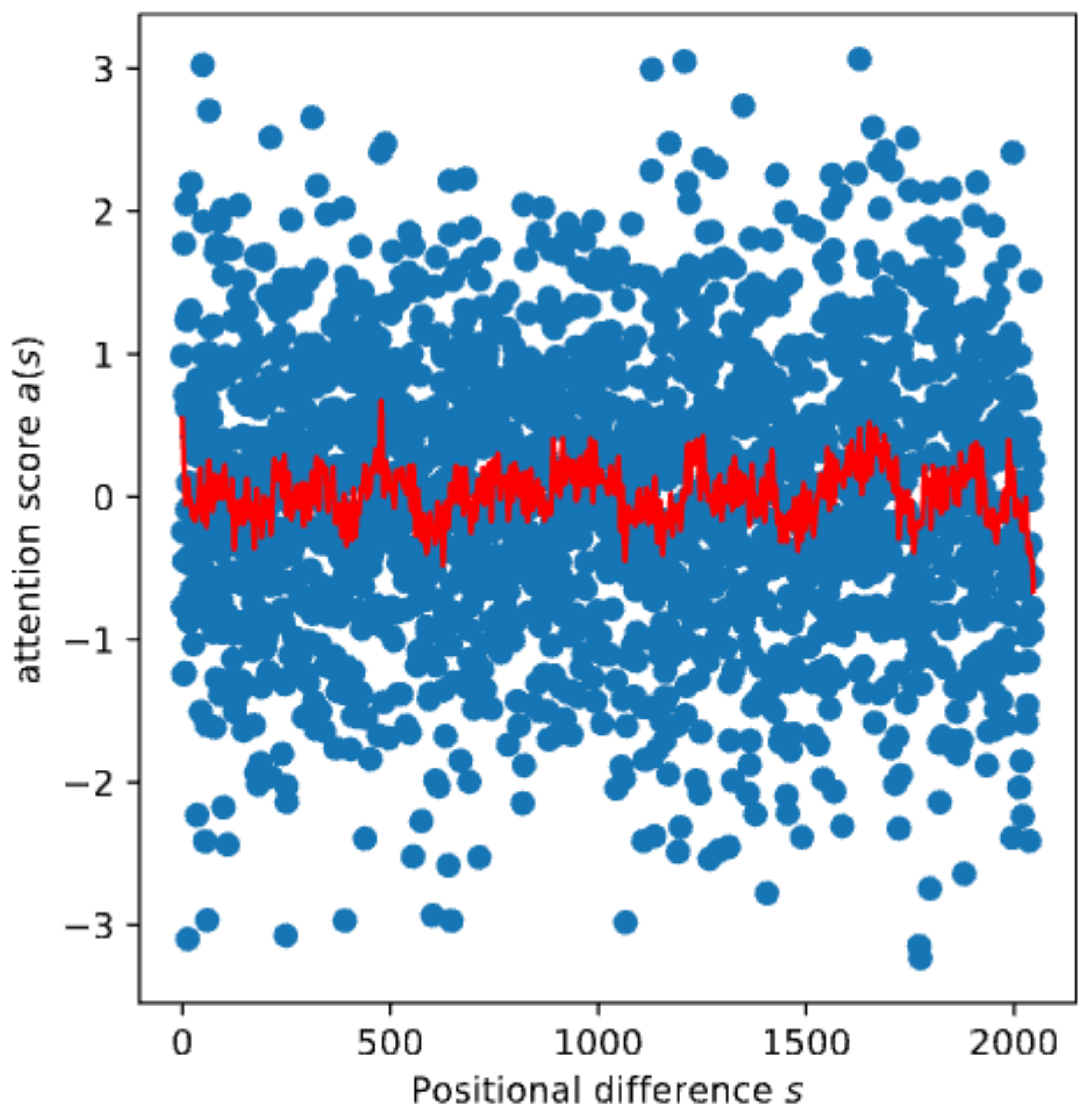
- Rotary Embeddings
  - Do not extrapolate well

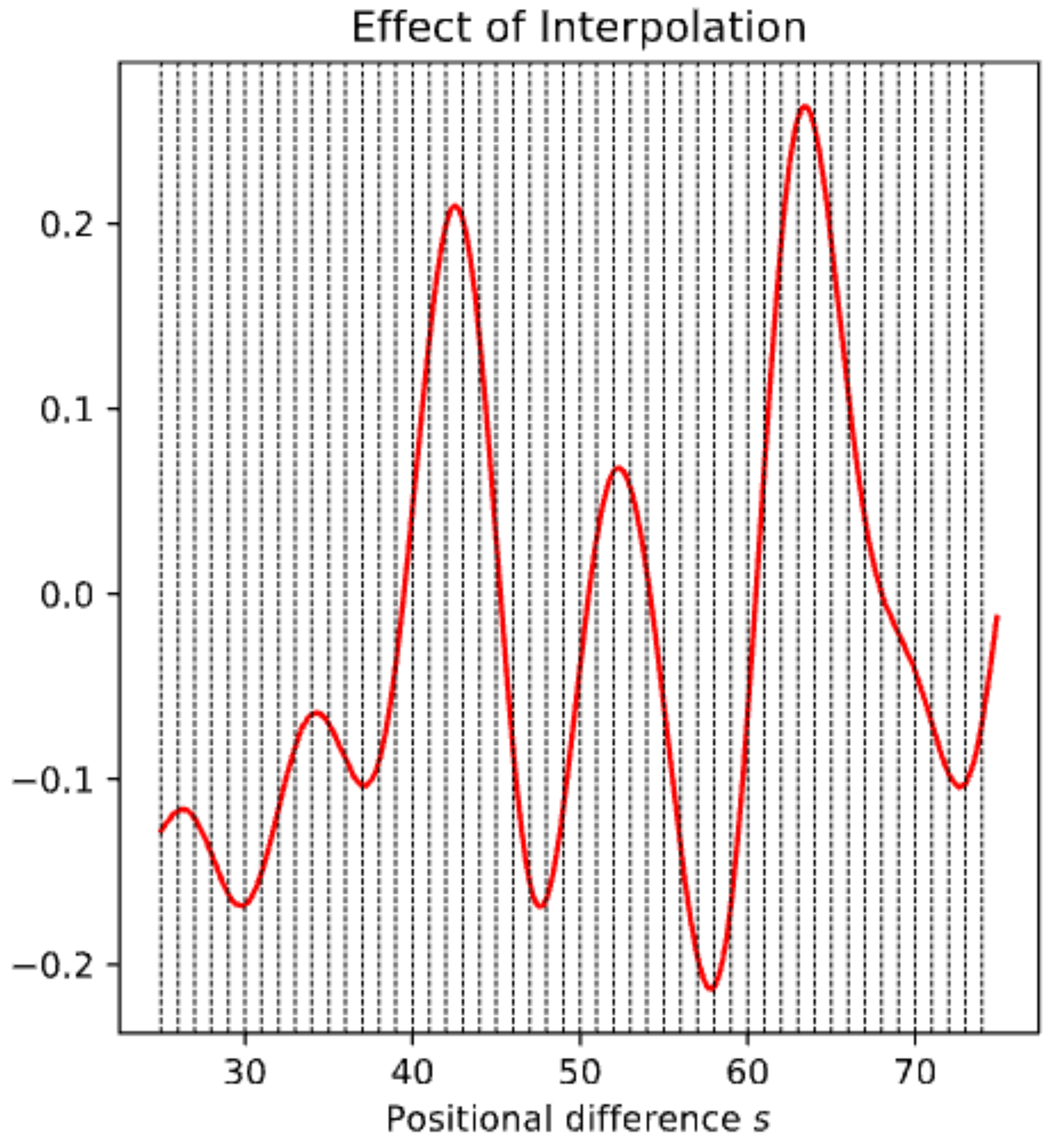


RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021
Extending Context Window of Large Language Models via Positional Interpolation, Chen etal 2023

Positional Embedding

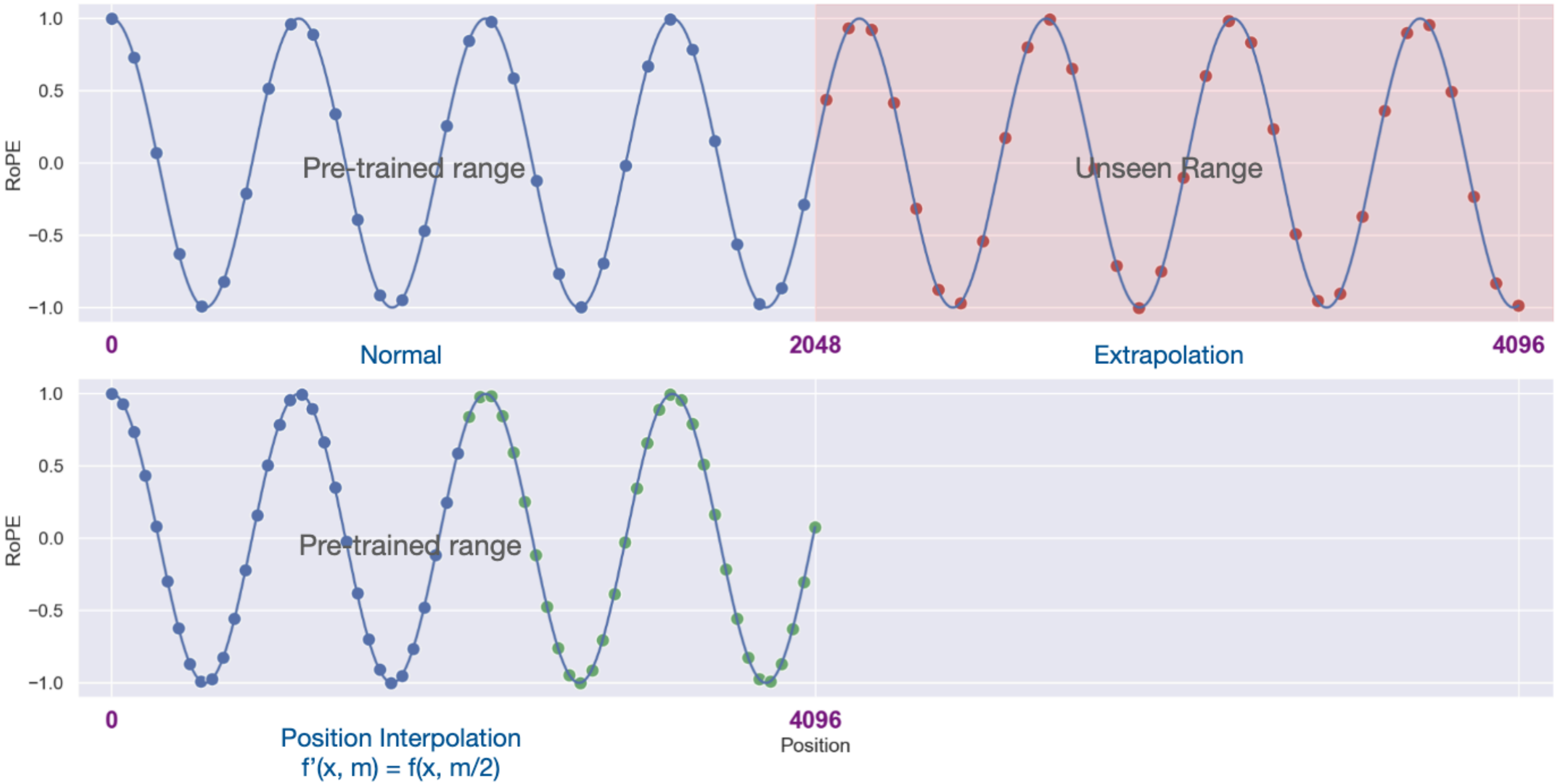
- Rotary Embeddings
  - Do not extrapolate well
  - But they interpolate





RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021
Extending Context Window of Large Language Models via Positional Interpolation, Chen etal 2023

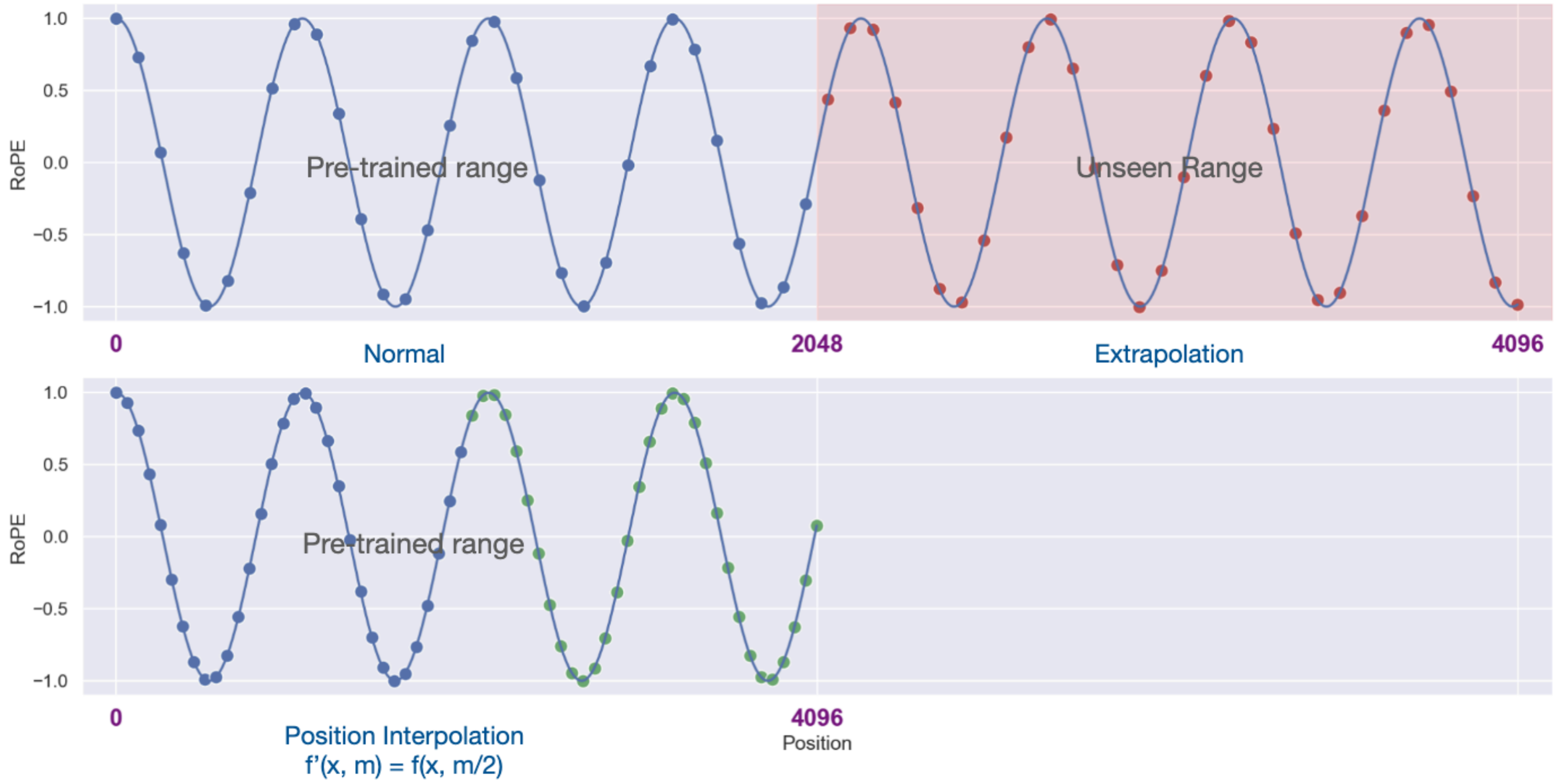
RoPE Scaling



RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021
Extending Context Window of Large Language Models via Positional Interpolation, Chen etal 2023

RoPE Scaling

- Extrapolation
  - Make token stream **longer**
  - Does not generalize
- RoPE Scaling
  - Make token stream **denser**
  - Model generalizes
- Widely used



RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021
Extending Context Window of Large Language Models via Positional Interpolation, Chen etal 2023

Long Context

What happens if we feed ten's of thousands of tokens into an LLM?

1. ~~OOM (Out Of Memory)~~
2. ~~Model will be very slow~~
3. ~~Model will produce garbage outputs~~

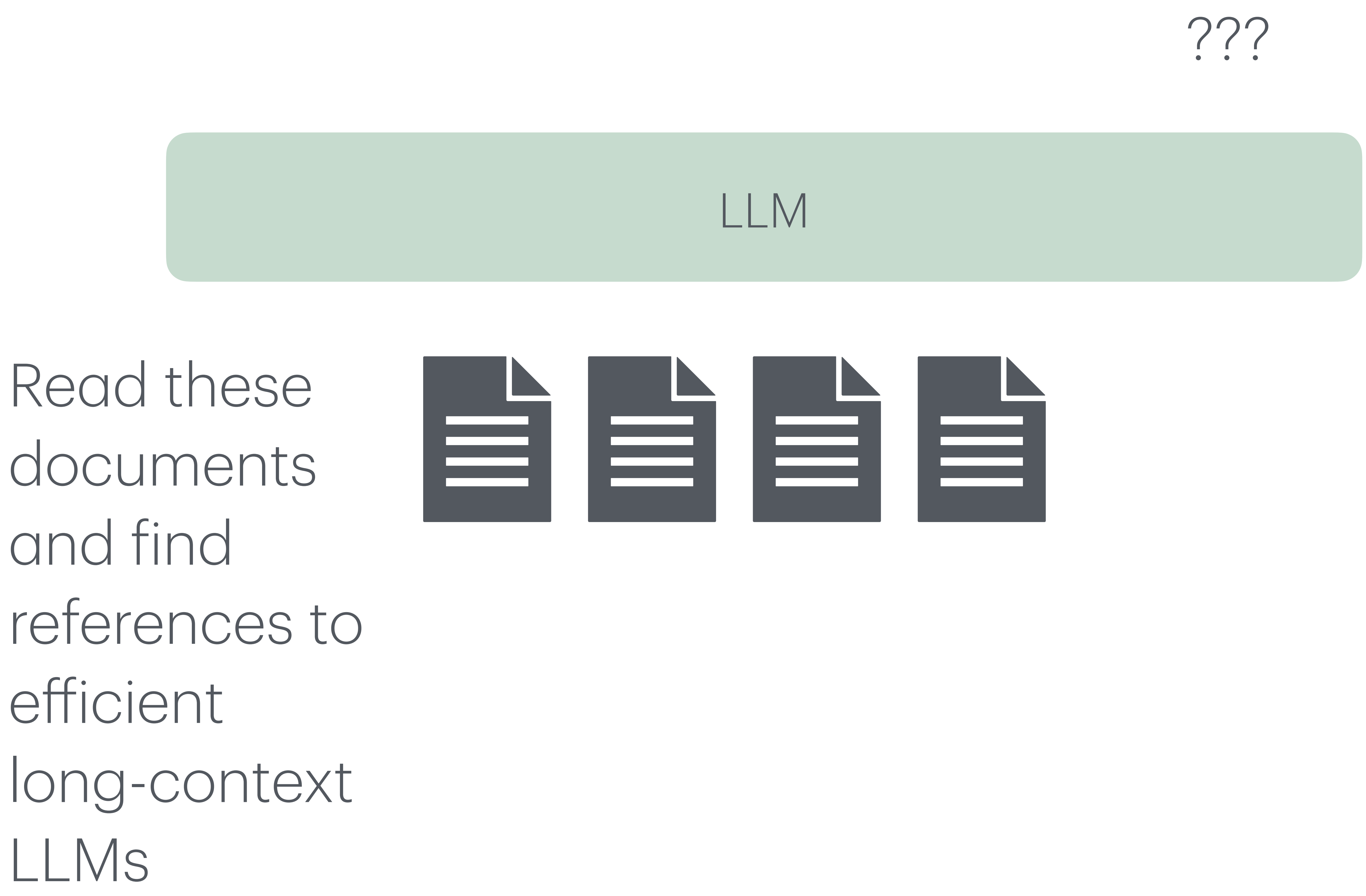
Activation Beacons and friends

RoPE scaling

Read these documents and find references to efficient long-context LLMs

Long Context

- Current model are **pre-trained** on **2-8k** token sequences
- Late stage pre-training **8k-128k**
    - RoPE Scaling
- Fine-tuned on variable length sequences



References

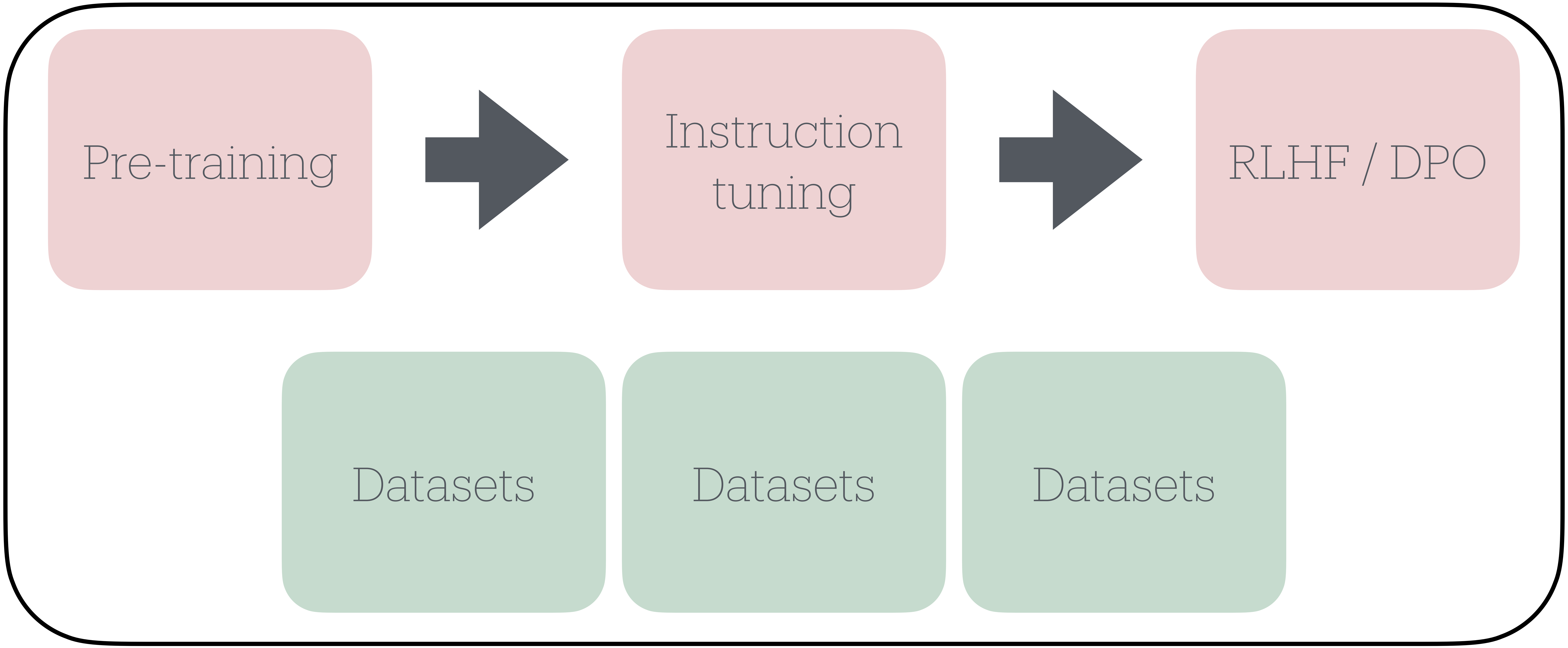
- [1] Long Context Compression with Activation Beacon, Zhang et al. 2024 (link)
- [2] RoFormer: Enhanced Transformer with Rotary Position Embedding, Su etal 2021 (link)
- [3] Extending Context Window of Large Language Models via Positional Interpolation, Chen et al 2023 (link)

Retrieval Augmented Generation

Philipp Krähenbühl, UT Austin

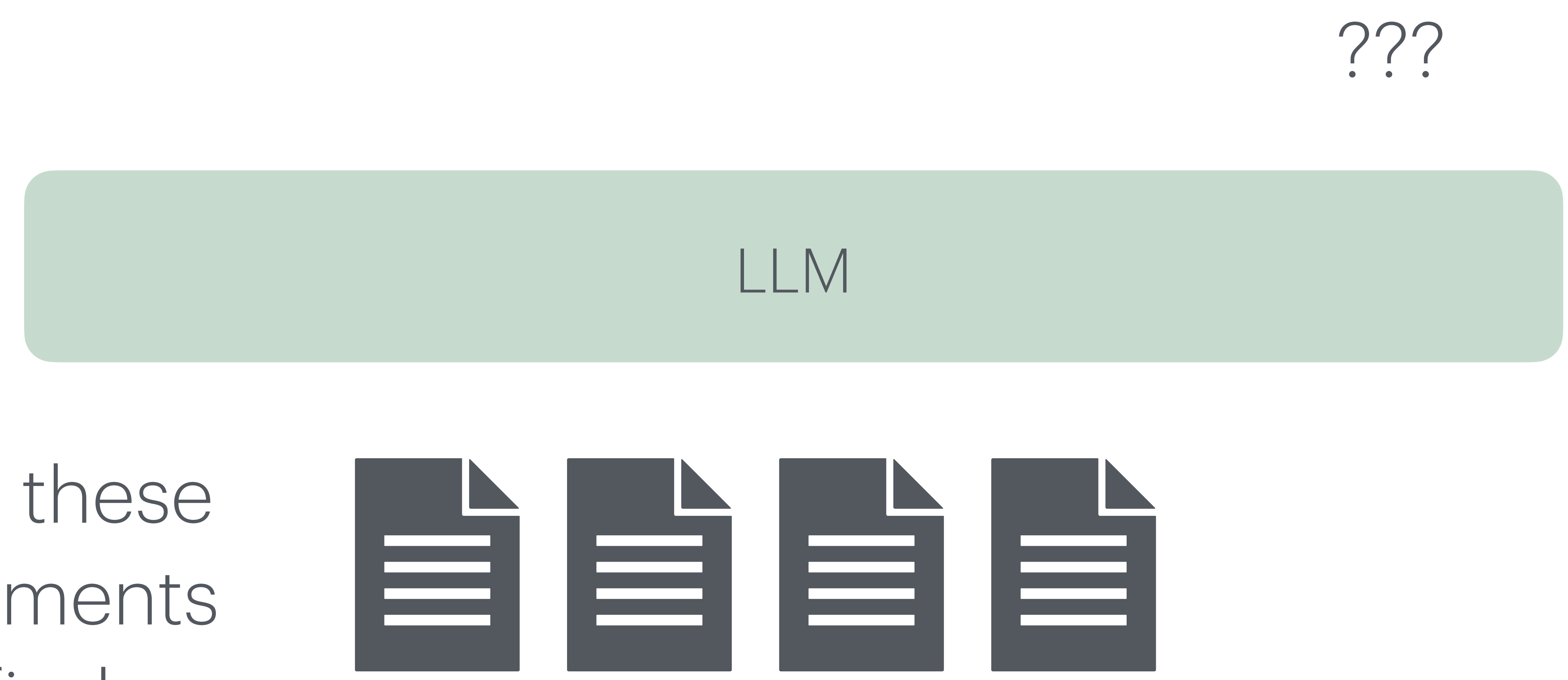
Full Picture

Basic LLM



Long Context

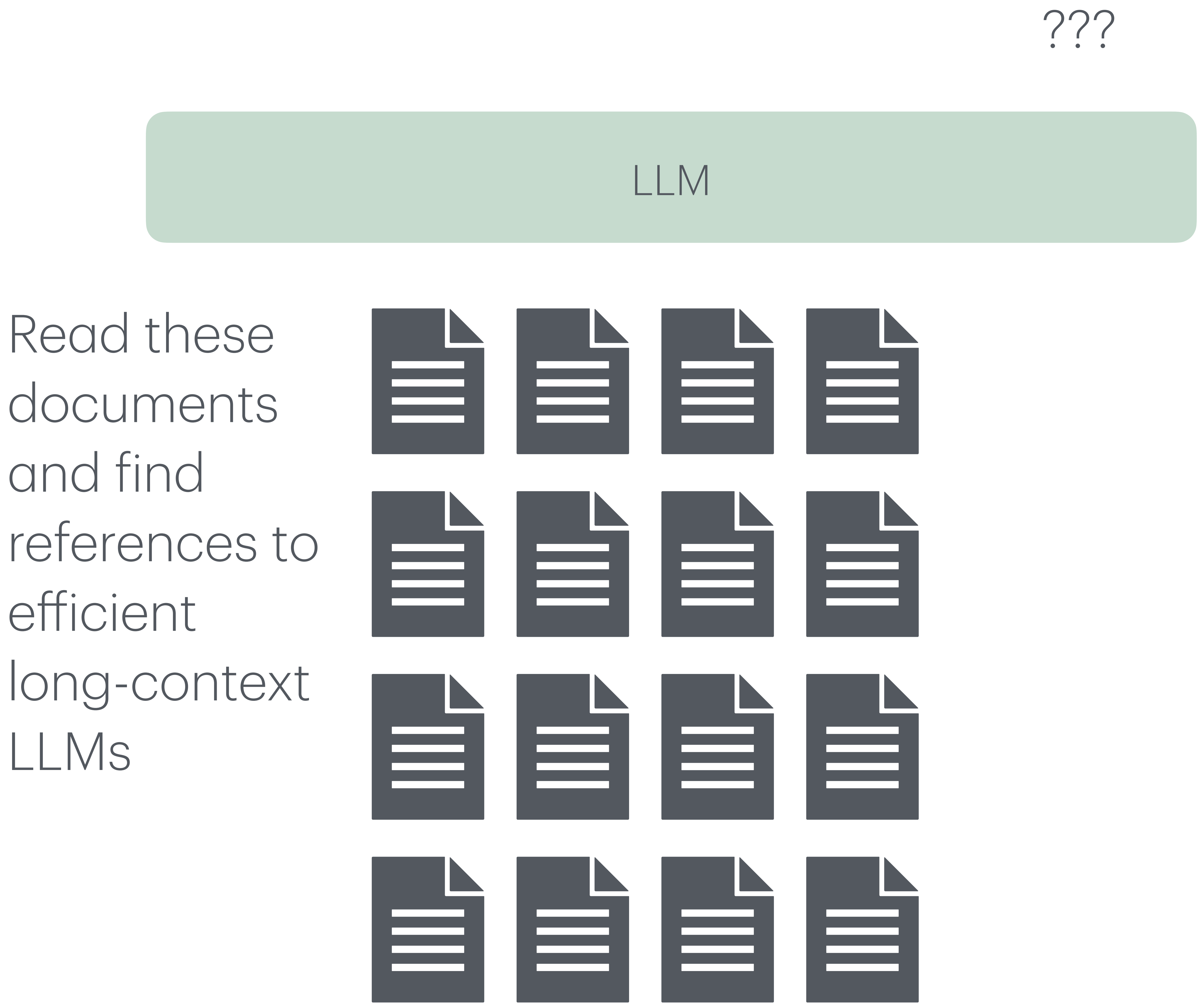
- Current model are **pre-trained** on **2-8k** token sequences
- Late stage pre-training **8k-128k**
  - RoPE Scaling
- Fine-tuned on variable length sequences



Read these documents and find references to efficient long-context LLMs

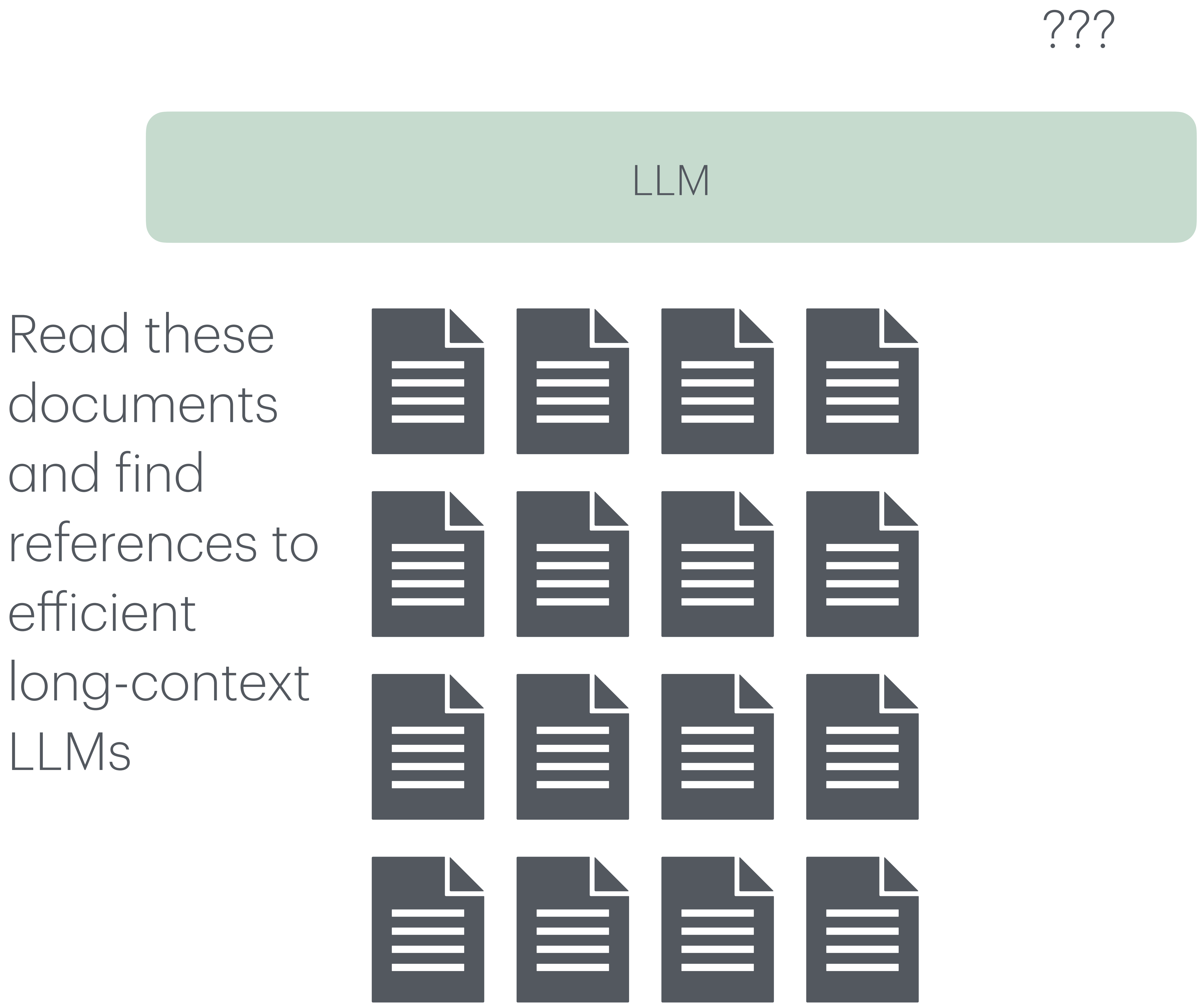
Longer Context

- What is we have even more inputs



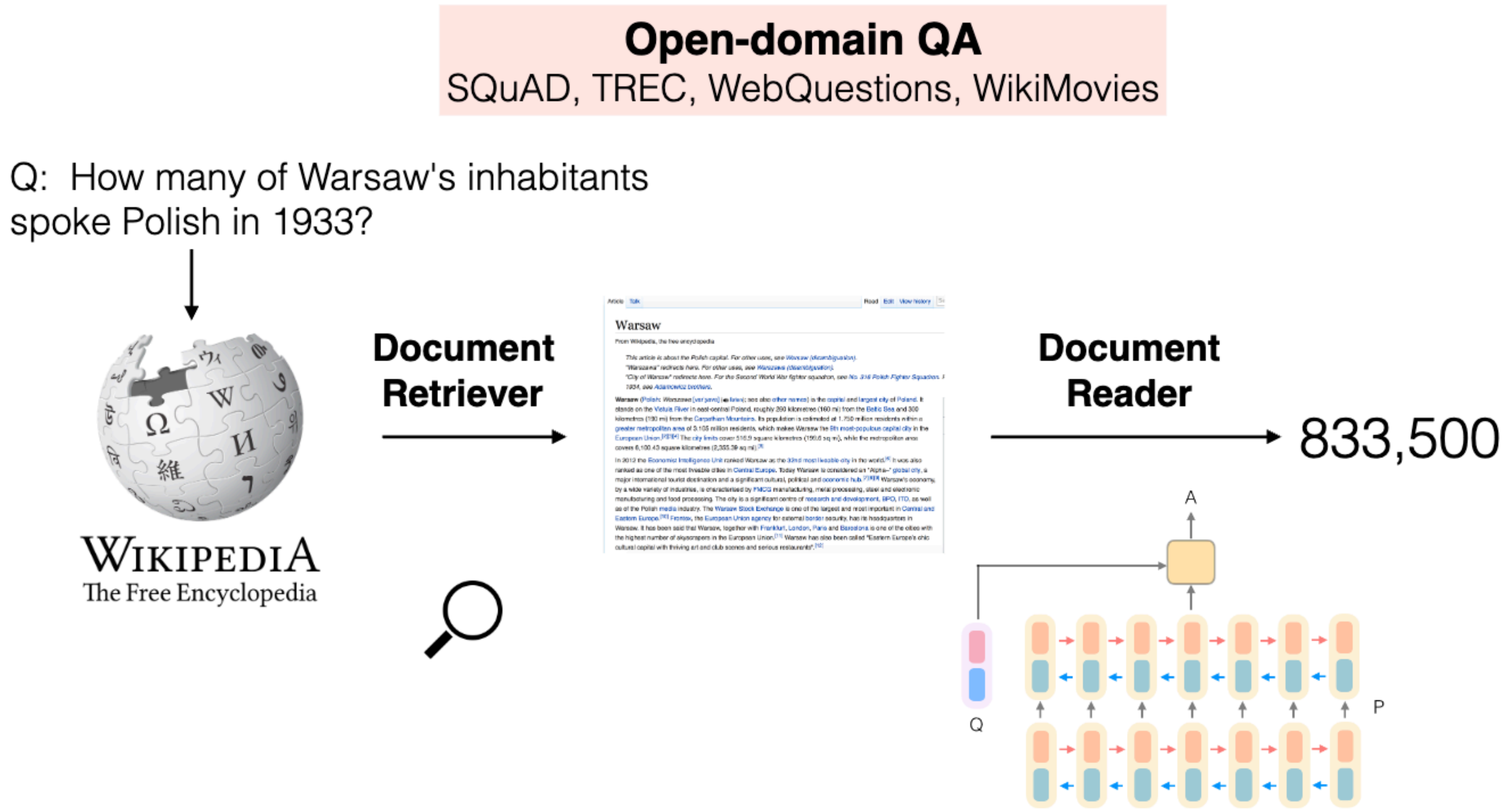
Longer Context

- What is we have even more inputs
  - We have to manage context



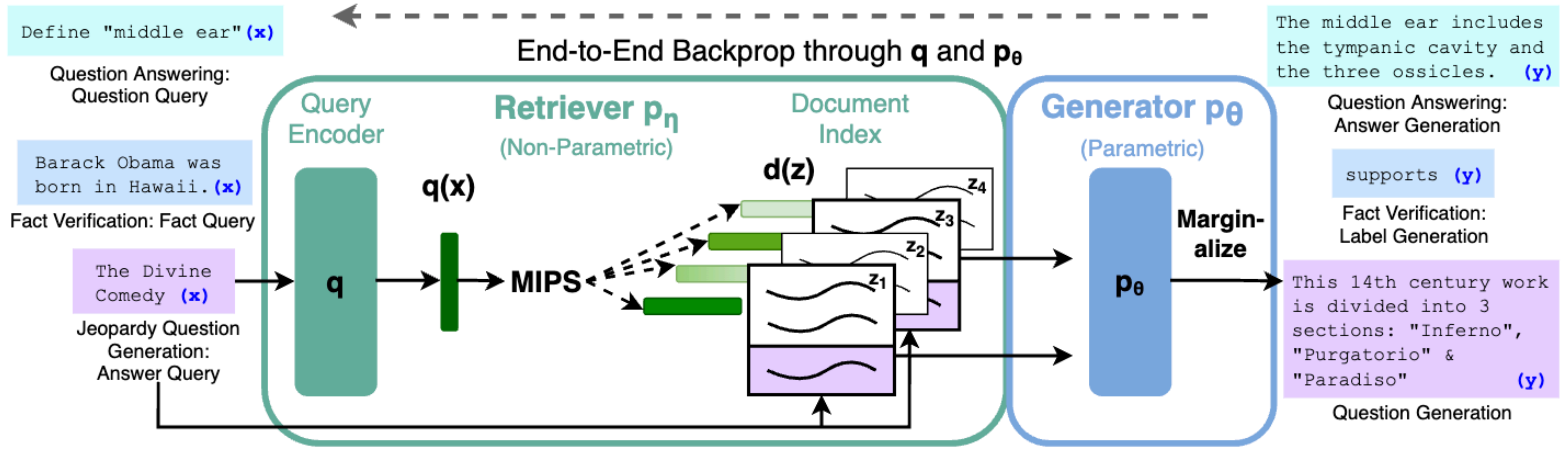
Longer Context

- Solution: Build a “system”
- Option 1
  - Document Retriever: LLM to retrieve most relevant document
  - Document Reader: LLM to answer request

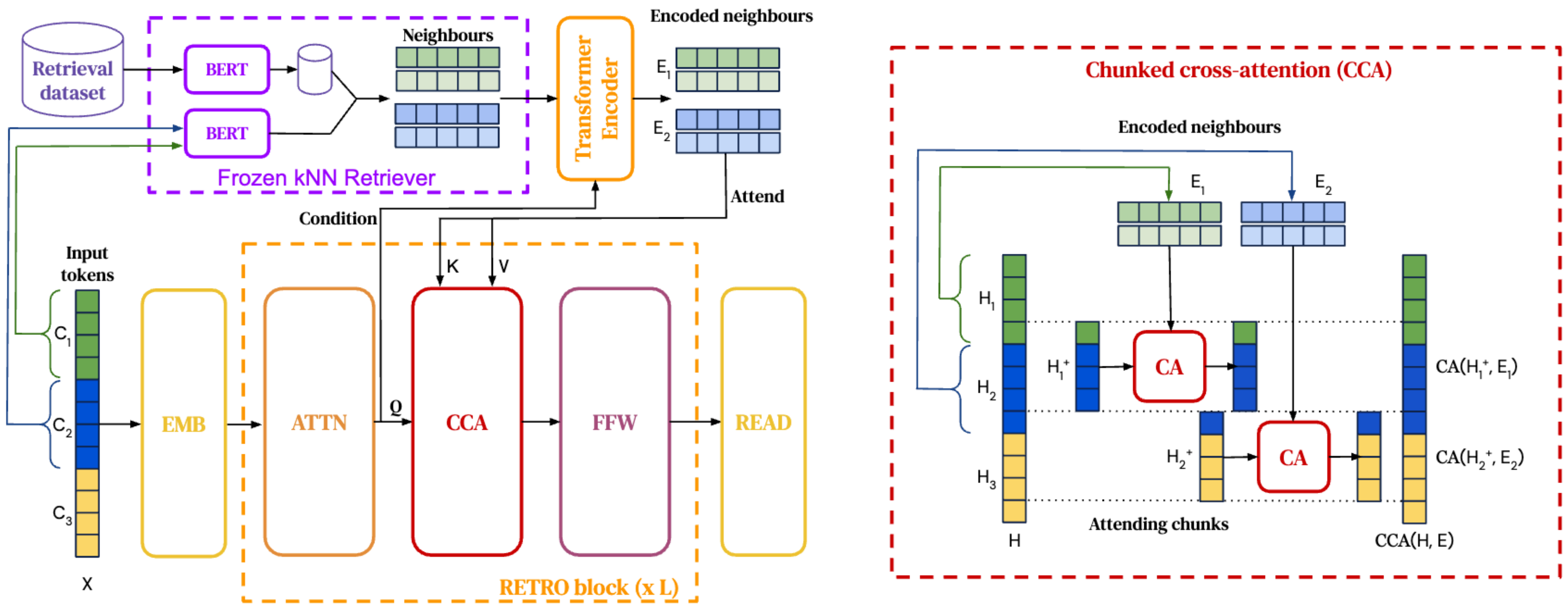


Reading Wikipedia to Answer Open-Domain Questions, Chen etal 2017

Longer Context



- Solution: Build a "system"
- Option 2
  - Document Retriever: LLM to retrieve all relevant documents
  - LLM to answer request with documents in context
  - Fine-tuned for task



Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Lewis etal 2020
REALM: Retrieval-Augmented Language Model Pre-Training, Guu etal 2020
Improving language models by retrieving from trillions of tokens, Borgeaud 2021

Longer Context

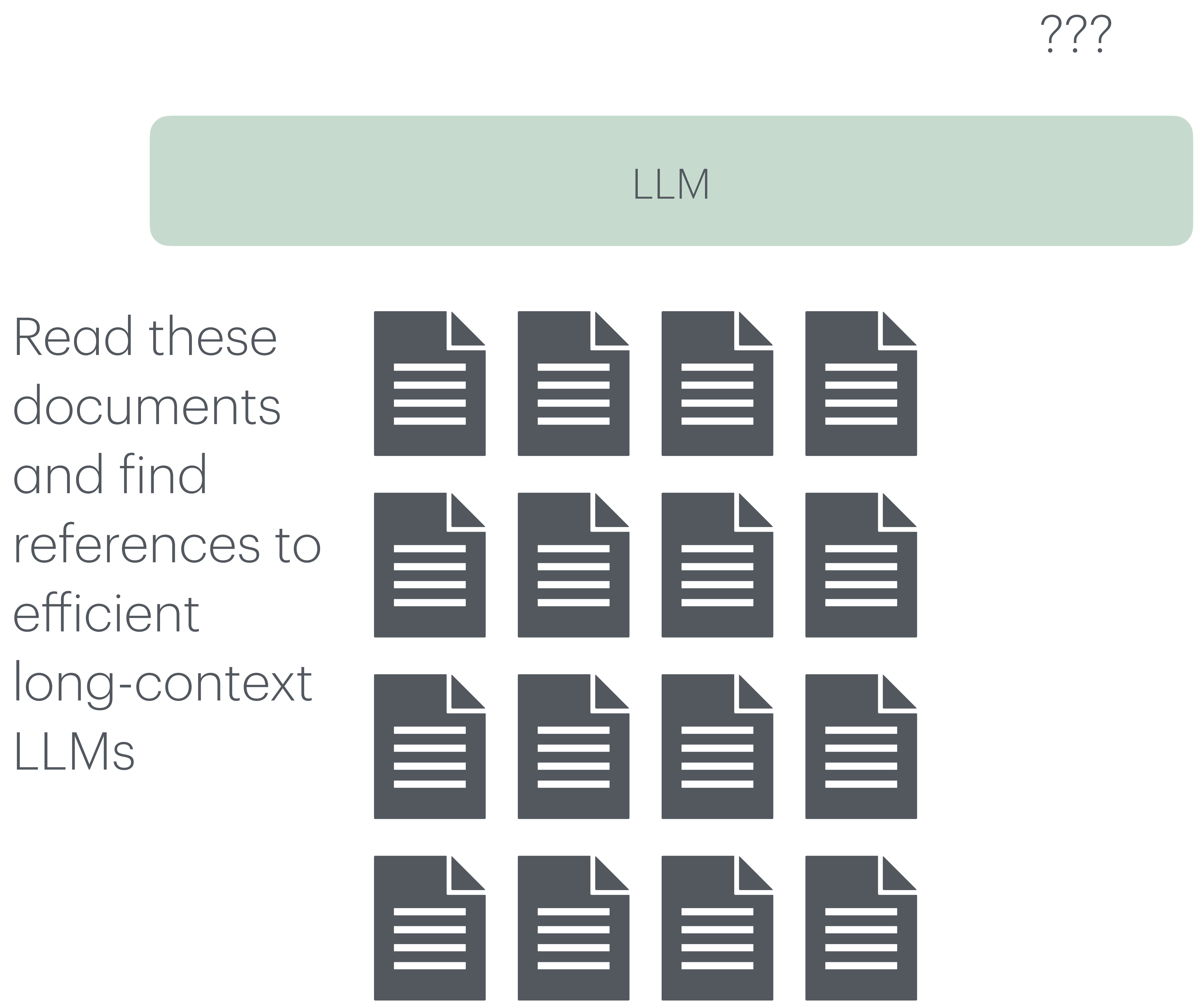
- Solution: Build a "system"
- Option 3
  - Document Retriever: LLM to retrieve all relevant documents
  - LLM to answer request with documents in context
  - ~~Fine-tuned for task~~ Model is prompted instead

In-Context Retrieval-Augmented Language Models, Ram etal 2023

Retrieval Augmented Generation

RAG

- A series of methods to manage the LLMs context
  - Some are trained
  - Some are just prompted



References

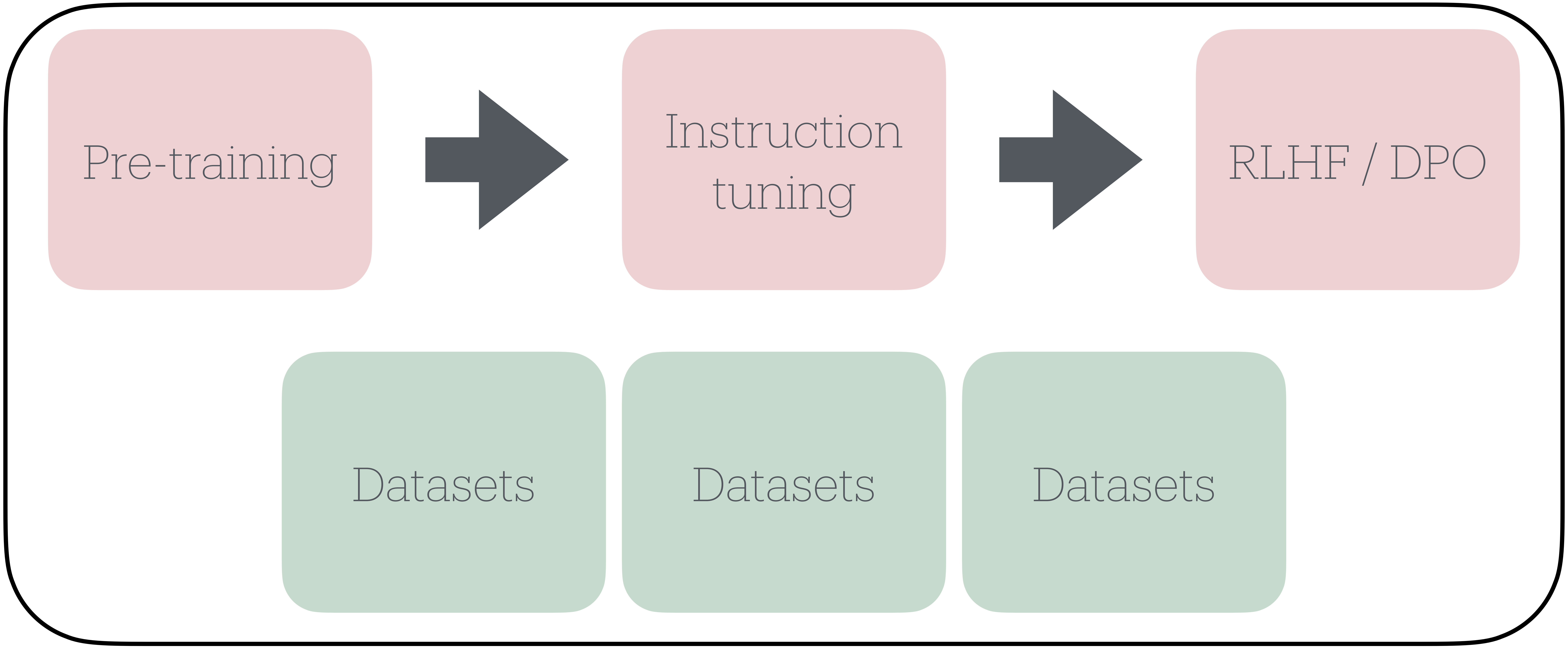
- [1] Reading Wikipedia to Answer Open-Domain Questions, Chen etal 2017 (link)
- [2] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Lewis etal 2020 (link)
- [3] REALM: Retrieval-Augmented Language Model Pre-Training, Guu etal 2020 (link)
- [4] Improving language models by retrieving from trillions of tokens, Borgeaud 2021 (link)
- [5] In-Context Retrieval-Augmented Language Models, Ram etal 2023 (link)

Structured Dialogues

Philipp Krähenbühl, UT Austin

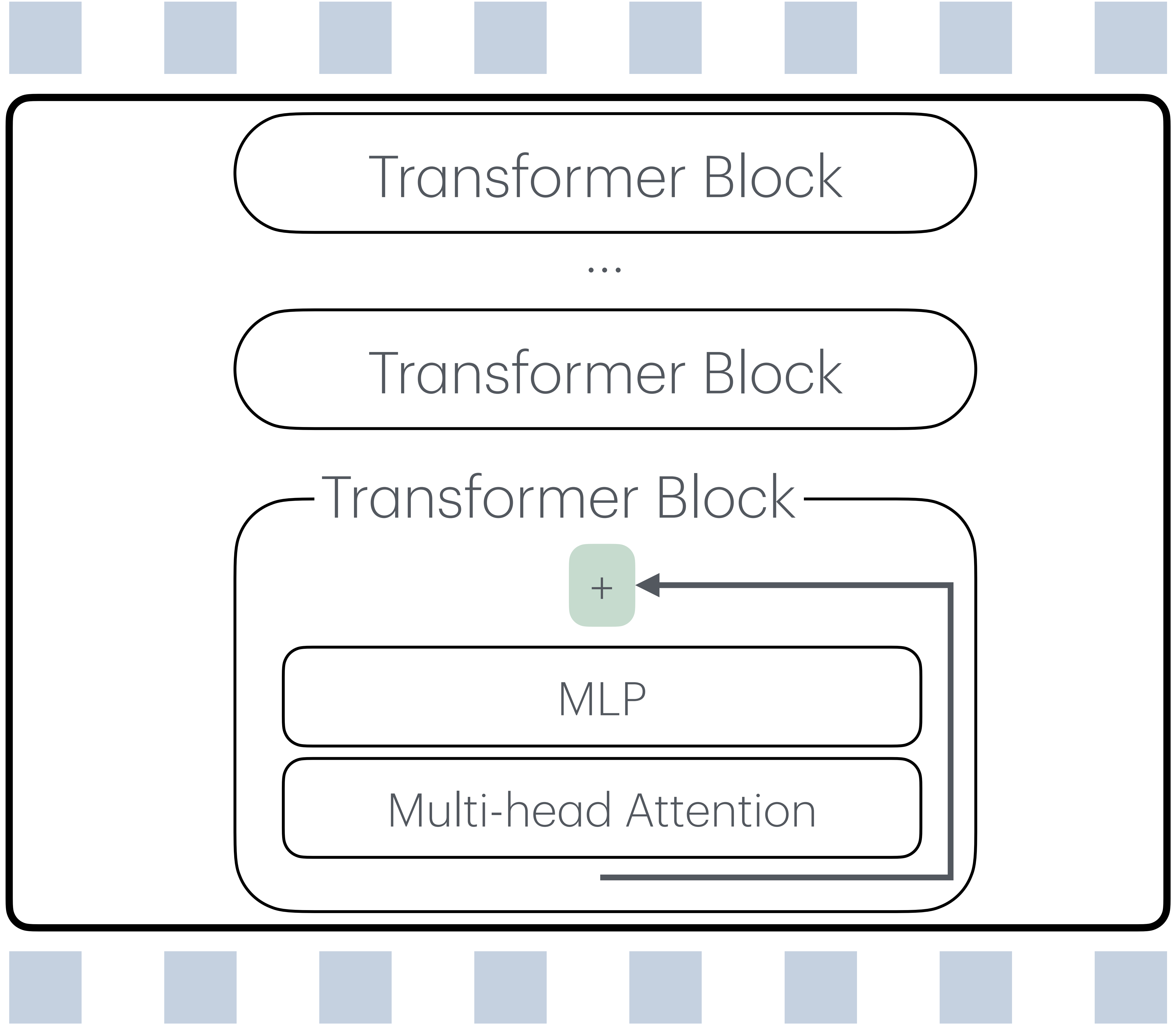
Full Picture

Basic LLM



Where does a LLM store information?

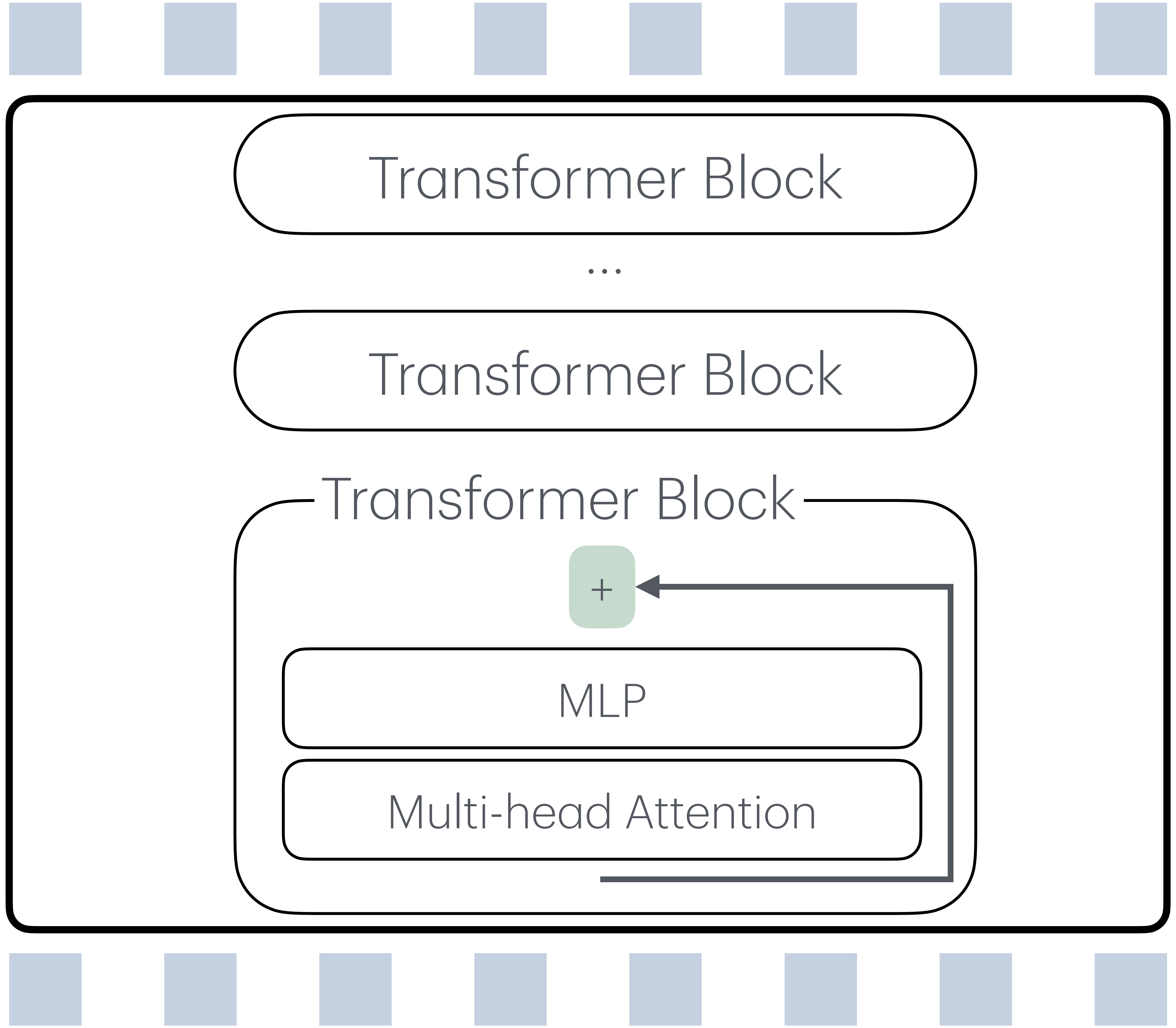
- Their weights
  - MLP and attention [1]
- Special tokens / activations [2,3]
  - Large activations or registers
- Their context



[1] Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws, Allen-Zhu 2024
[2] Vision Transformers Need Registers, Darcet etal 2023
[3] Massive Activations in Large Language Models, Sun etal 2024

Information in weights

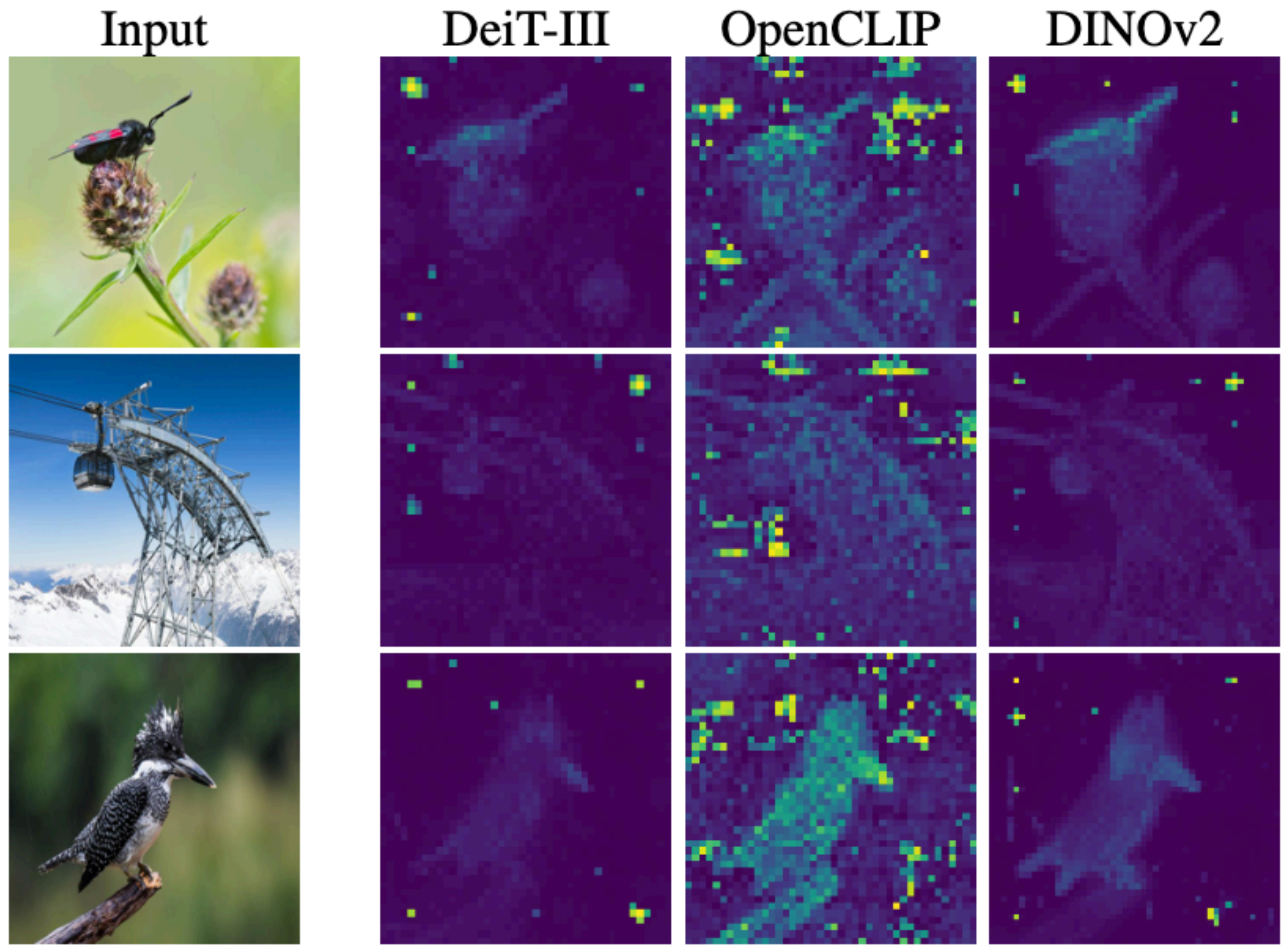
- LLMs can store up to 2 bits of information per weight [1]
  - In MLP
  - In Attention
  - 2 bits require very long training and multiple (up to 1000) augmentations of same information

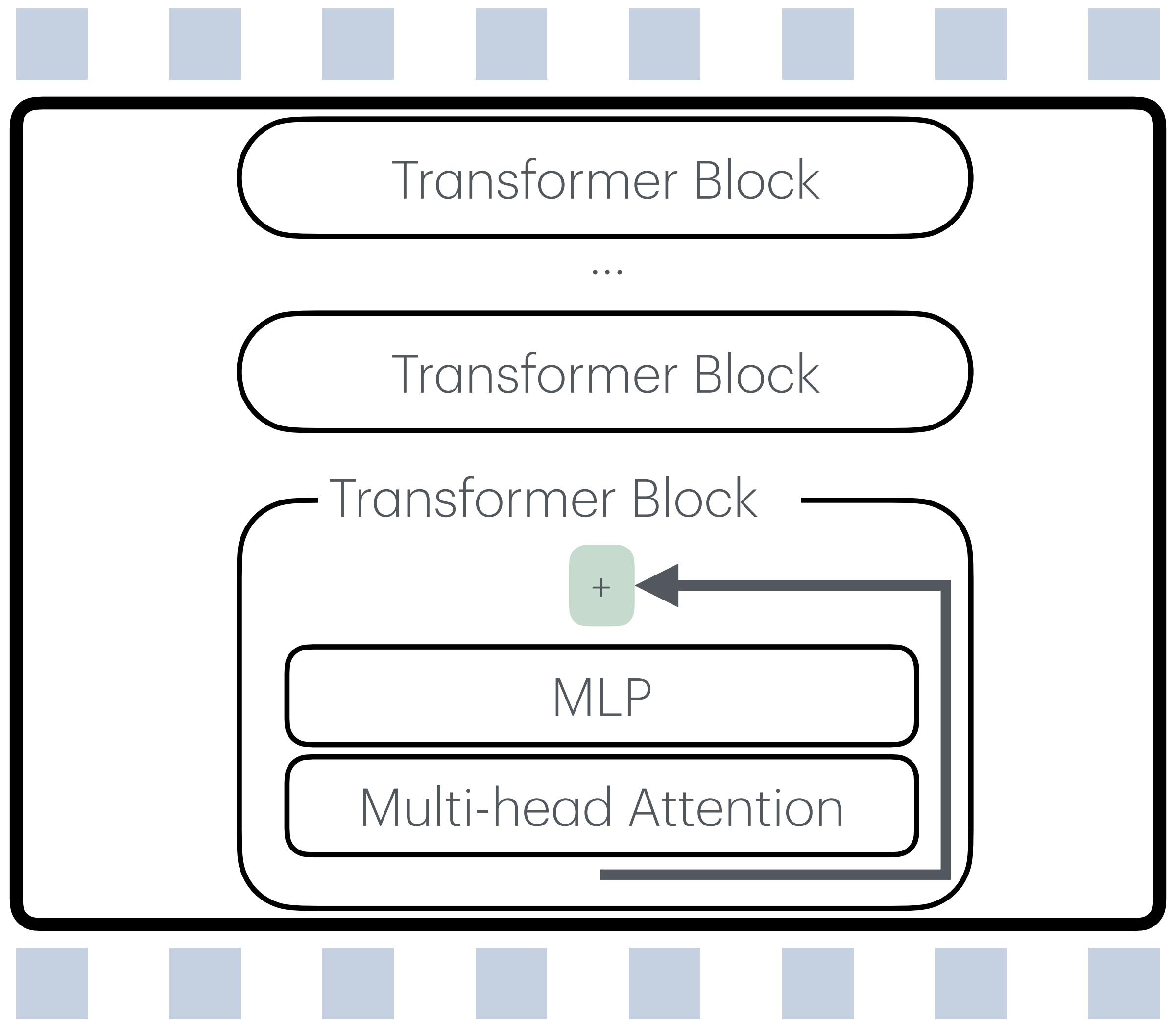


Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws, Allen-Zhu 2024

Special tokens / activations

- LLMs use special tokens to store information
  - LLMs attend to <BOS> token
  - VLMs attend to background

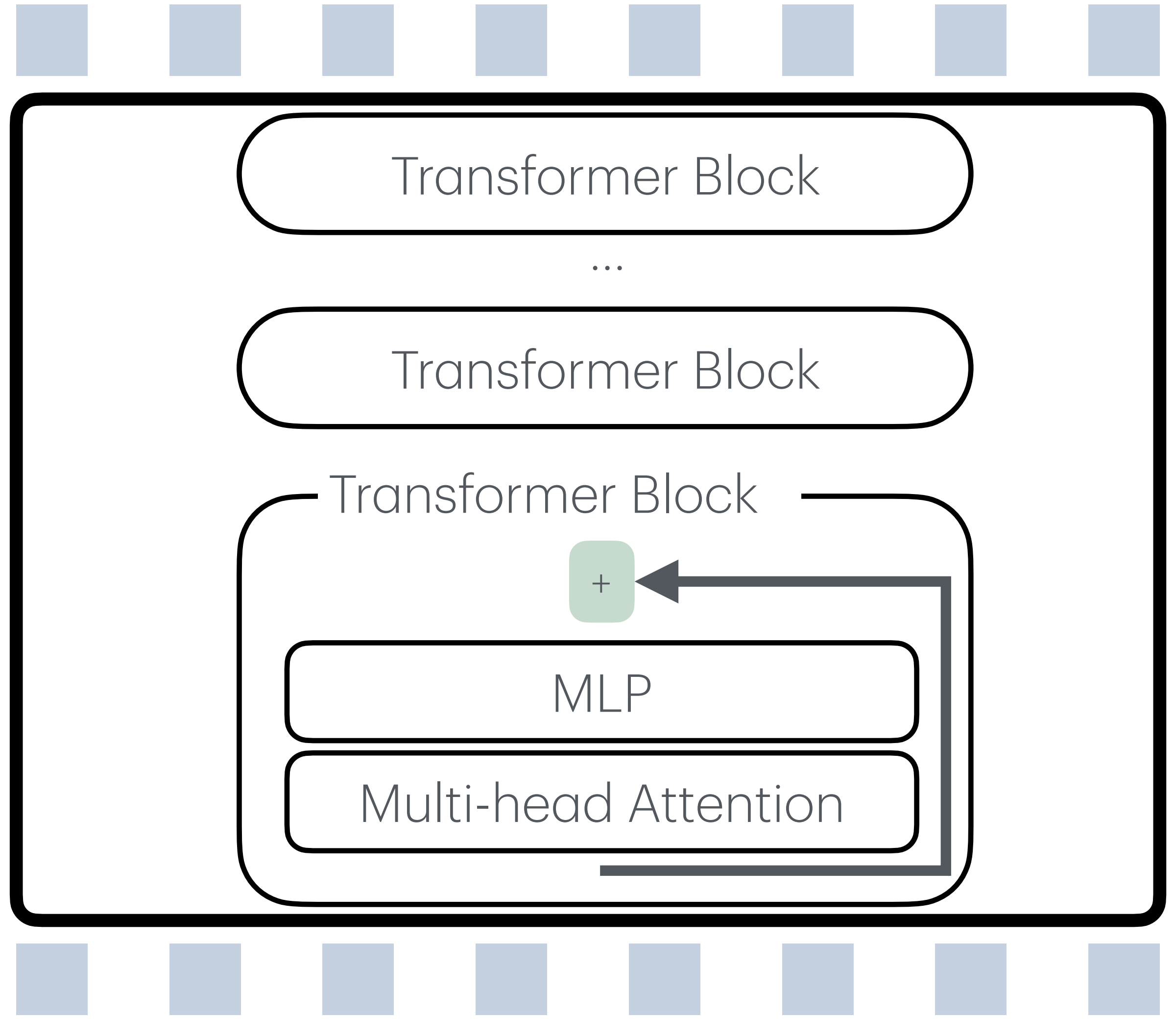




[1] Vision Transformers Need Registers, Darcet etal 2023
[2] Massive Activations in Large Language Models, Sun etal 2024

Context

- LLMs store information in their context
- Examples
  - System prompt
  - Retrieval Augmented Generation
  - ...



[1] Vision Transformers Need Registers, Darcet etal 2023
[2] Massive Activations in Large Language Models, Sun etal 2024

In context learning

- Describe the task
  - Give examples input - output pairs
  - Then ask for your specific

```
Translate words from English to German using
JSON as an output. Here are some examples

Car
{“English”: “Car”, “German”: “Auto”}

Sun
{“English”: “Sun”, “German”: “Sonne”}

Moon
```

Language Models are Few-Shot Learners, Brown etal 2020

In context learning

Why does it work?

- LLMs like repeating patterns
  - Likely exist in pre-training data
- Examples of in-context prompts and answers during training (instruction tuning, alignment)

```
Translate words from English to German using
JSON as an output. Here are some examples

Car
{“English”: “Car”, “German”: “Auto”}

Sun
{“English”: “Sun”, “German”: “Sonne”}

Moon
```

Language Models are Few-Shot Learners, Brown etal 2020

In context learning

What does it work for?

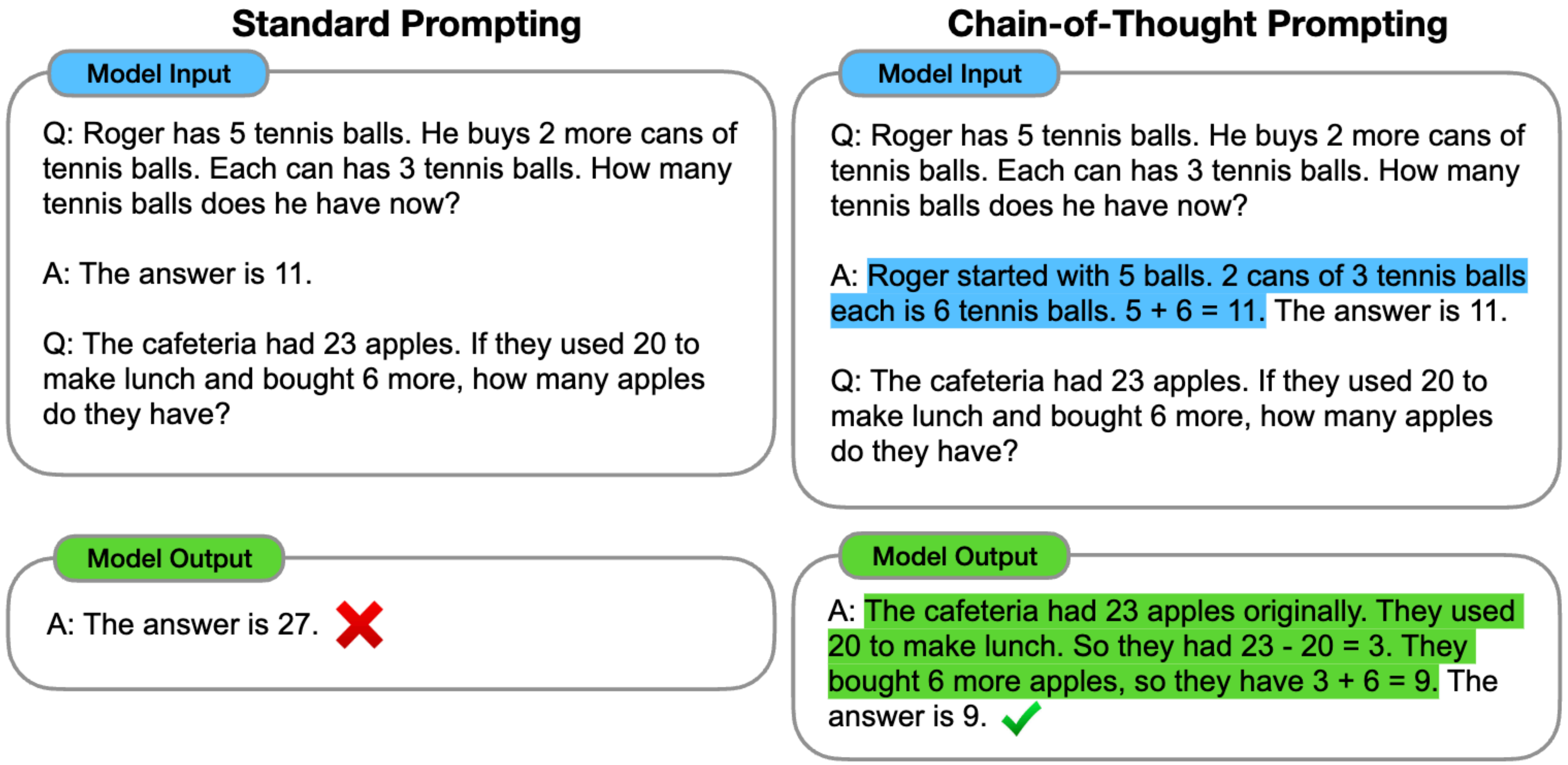
- Formatting outputs
- Simple requests

```
Translate words from English to German using
JSON as an output. Here are some examples

Car
{“English”: “Car”, “German”: “Auto”}

Sun
{“English”: “Sun”, “German”: “Sonne”}

Moon
```

Language Models are Few-Shot Learners, Brown etal 2020

Chain of thought

- Ask model to derive answer
  - Pre-instruction tuning: In-context example of reasoning
  - Post-instruction tuning
    - Ask model to think step-by-step before giving the answer
    - Guide model through thinking process

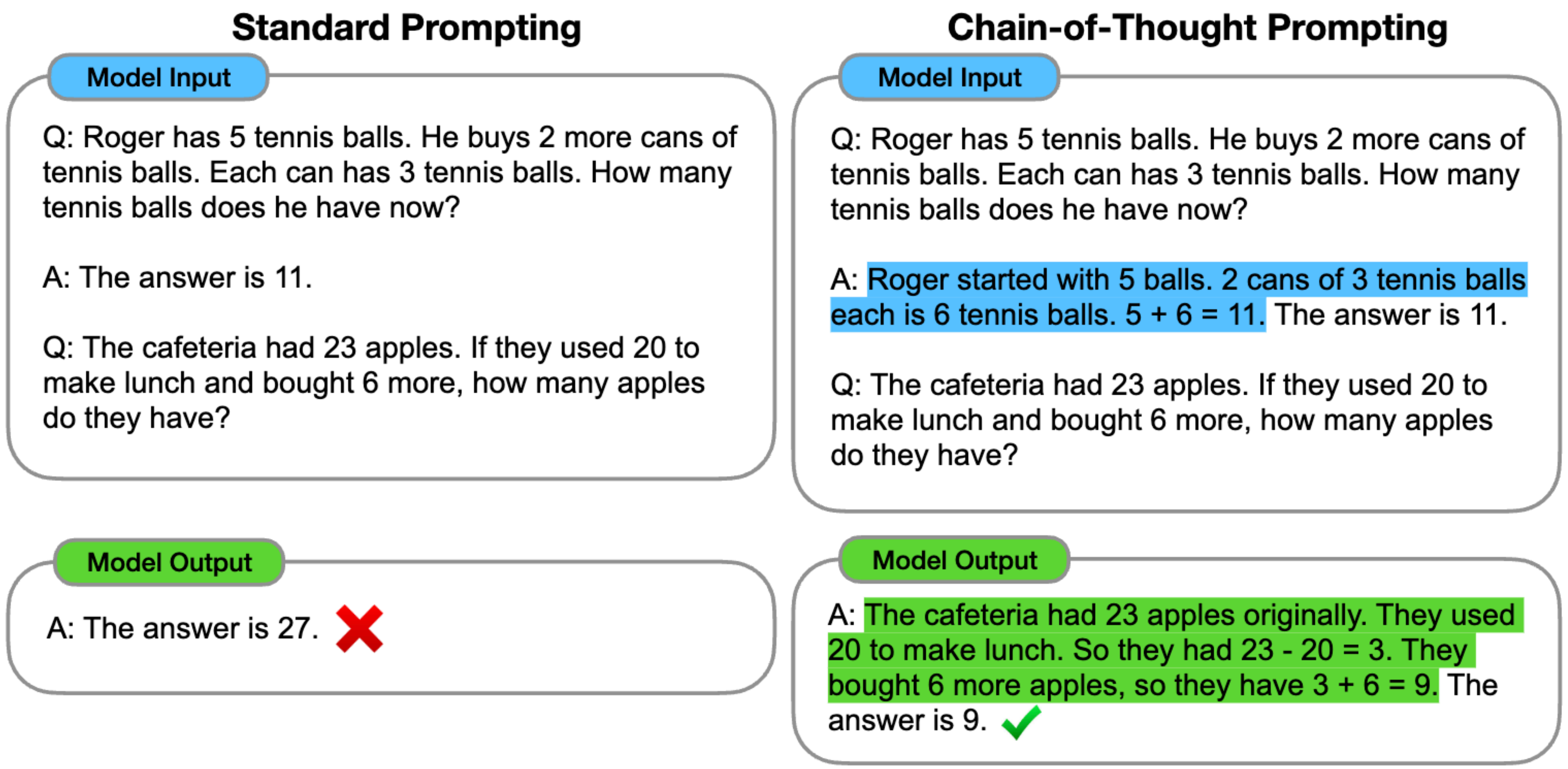


Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei etal 2022

Chain of thought

Why does it work?

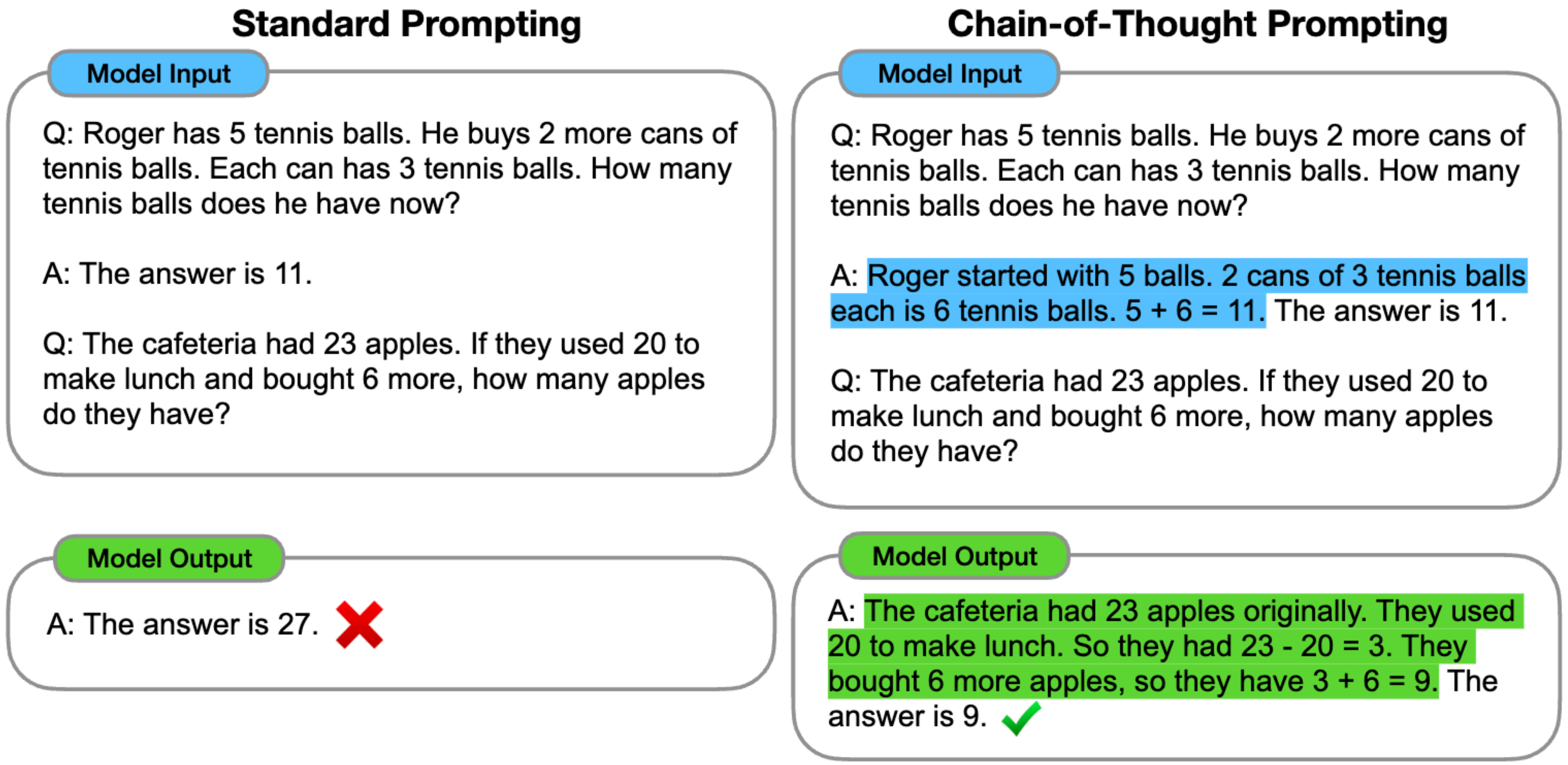
- More output tokens = better performance
  - Delays making a decision
- Can work around tokenization issues
  - Break up numbers



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei etal 2022

Chain of thought

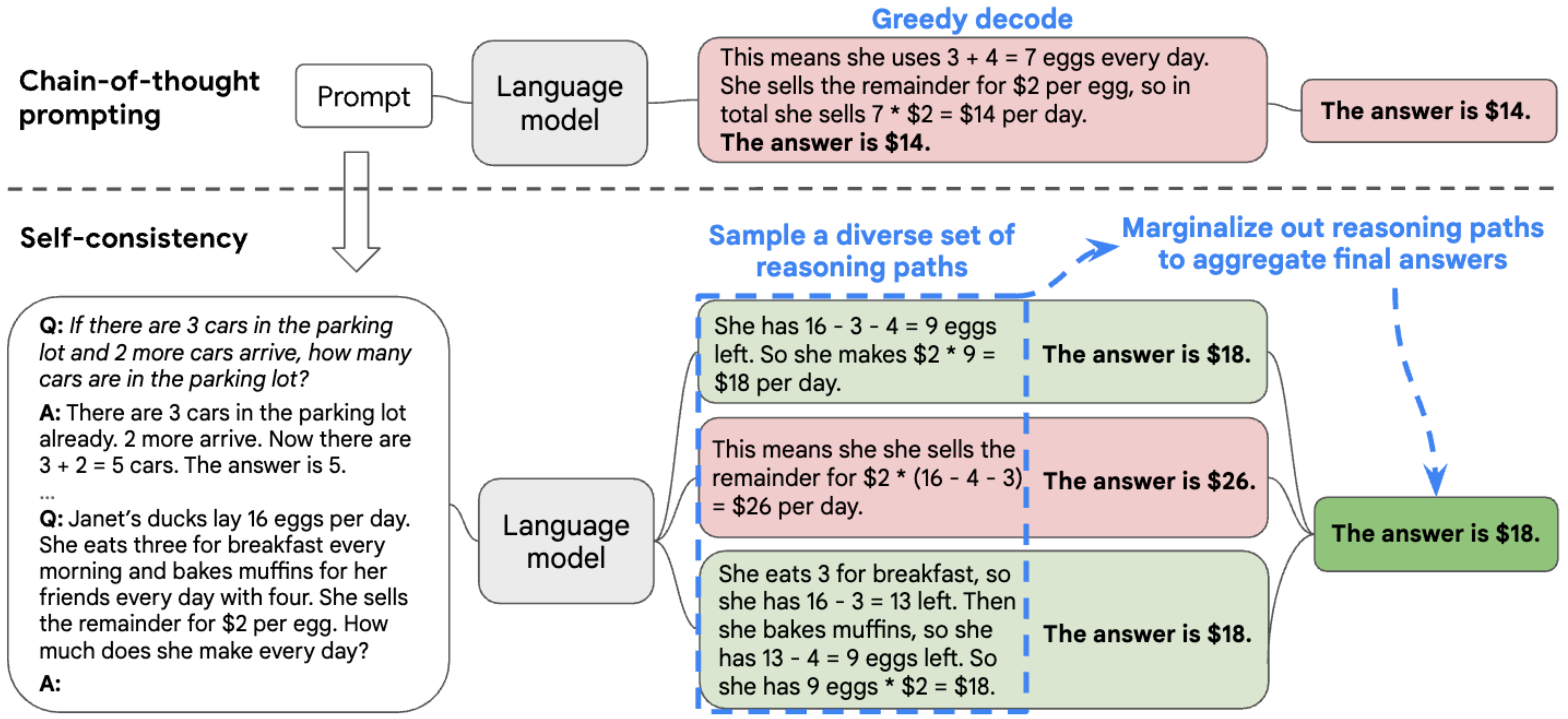
- Order matters
  - Think first, then answer
  - Chain-of-BS: Ask model to give answer and justify it



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei etal 2022

Self-Consistency

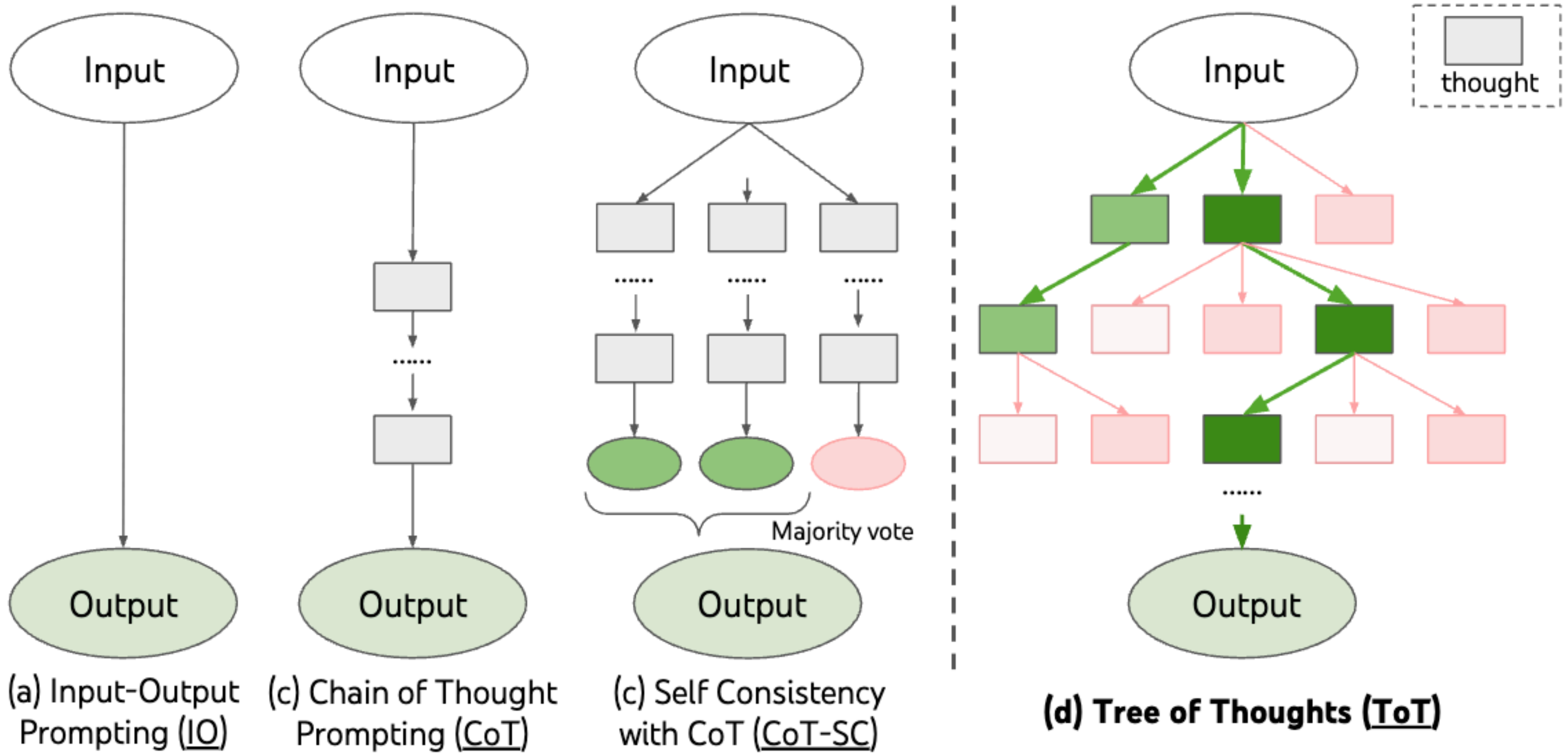
- Let the model reason multiple times
  - Pick the most frequent answer
  - Mathematically: Marginalize out reasoning to obtain most likely answer

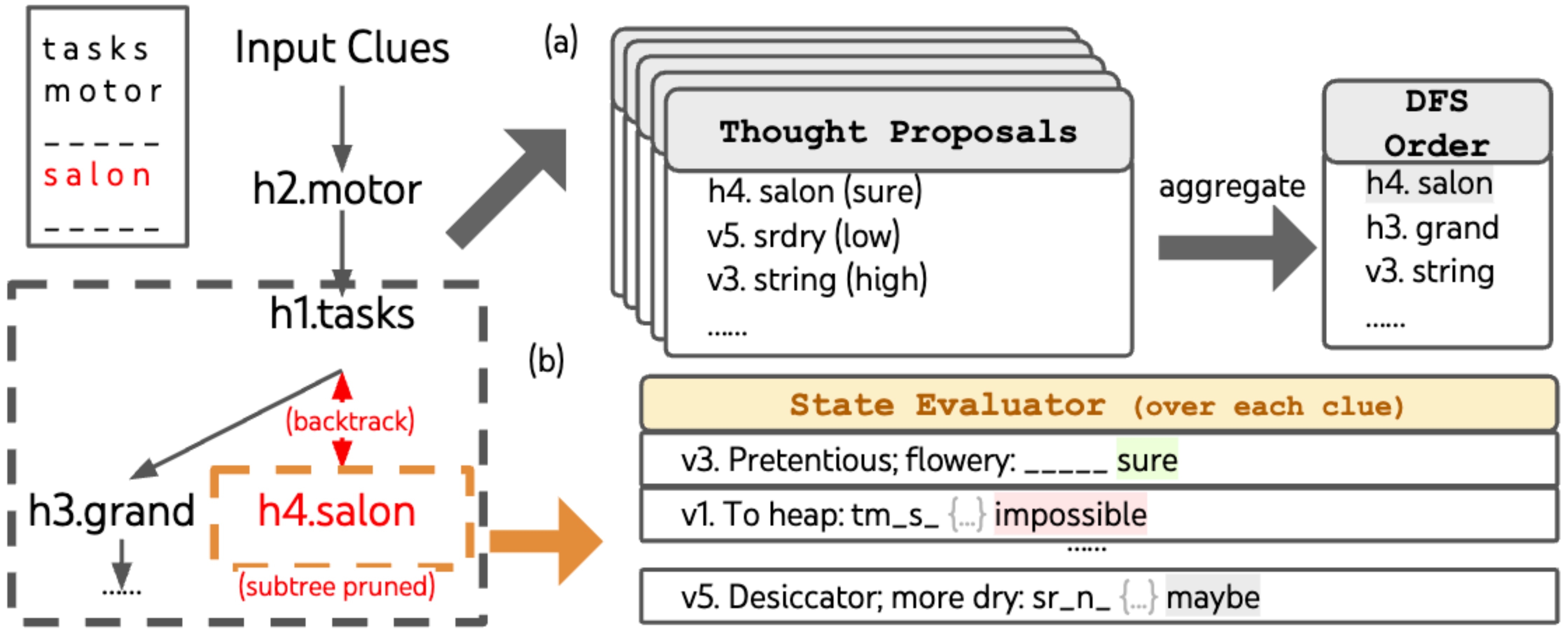


Self-Consistency Improves Chain of Thought Reasoning in Language Models, Wang etal 2022

Tree of Thoughts

- Combine tree search with CoT
  - Requires a state-evaluator (i.e. reward/cost/scoring function or second LLM)

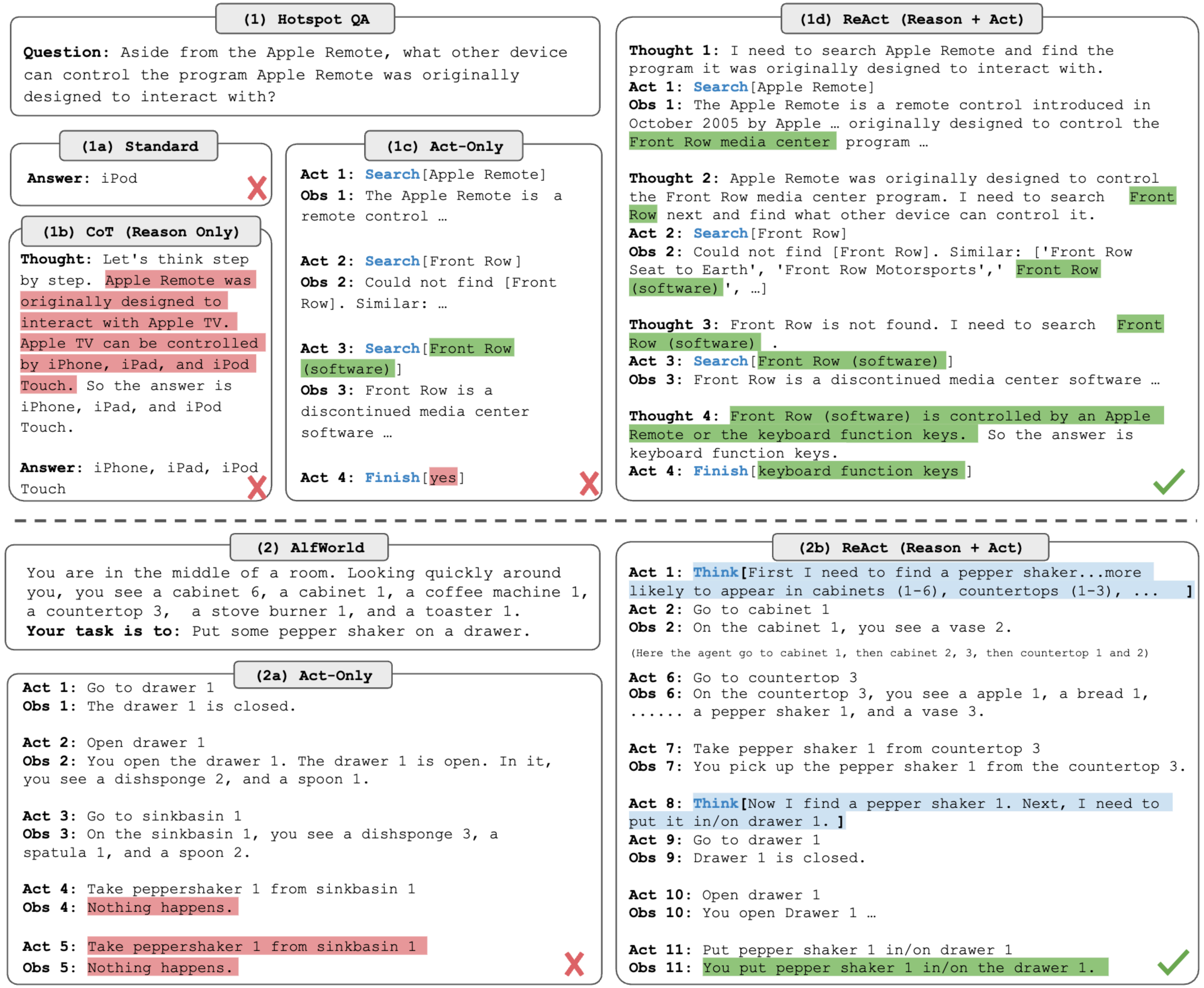




Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao etal 2023

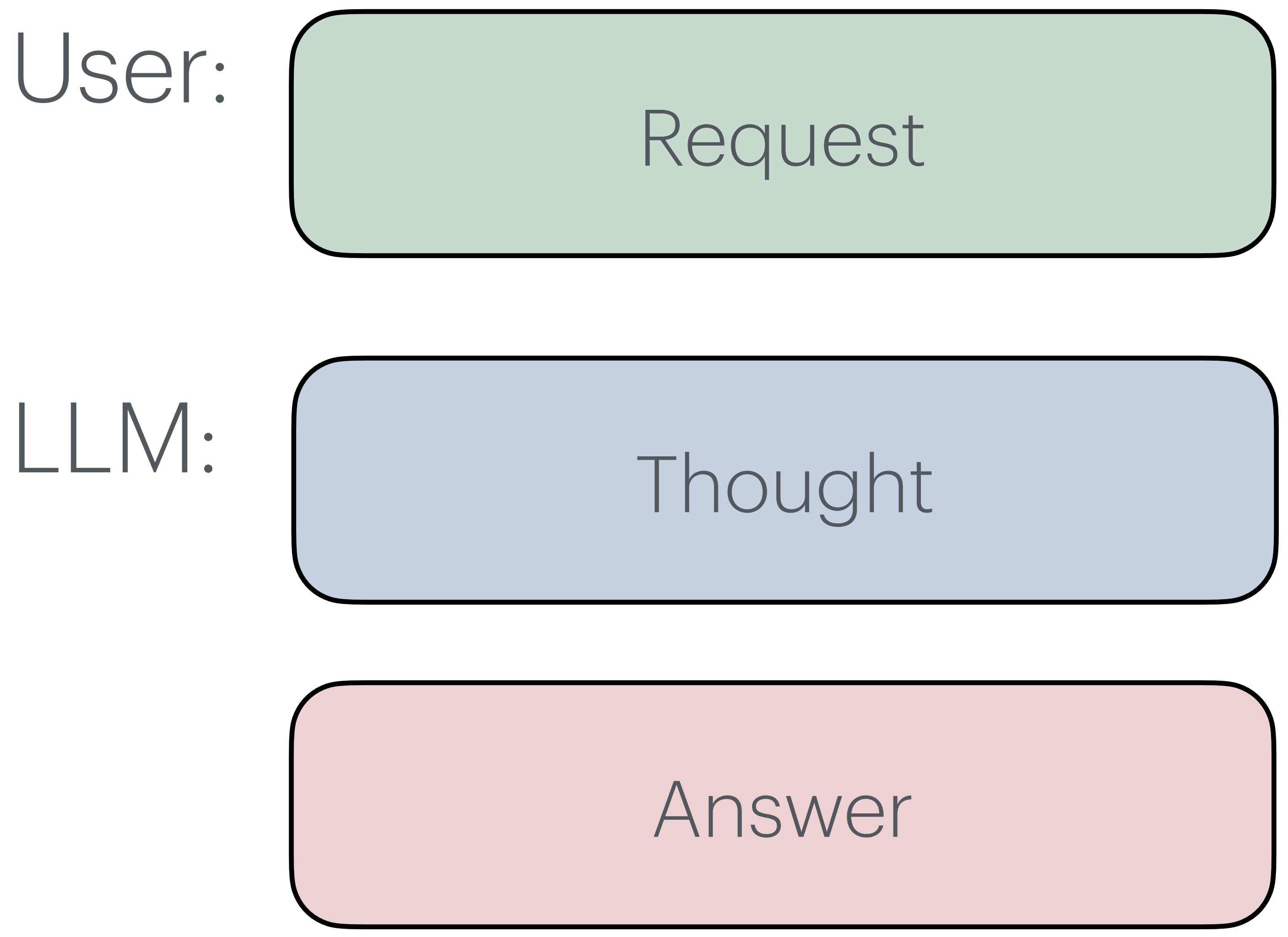
ReACT

- Chain of thought for iterative actions / tool use
  - Thought
  - Action
  - Observation (from external tool)



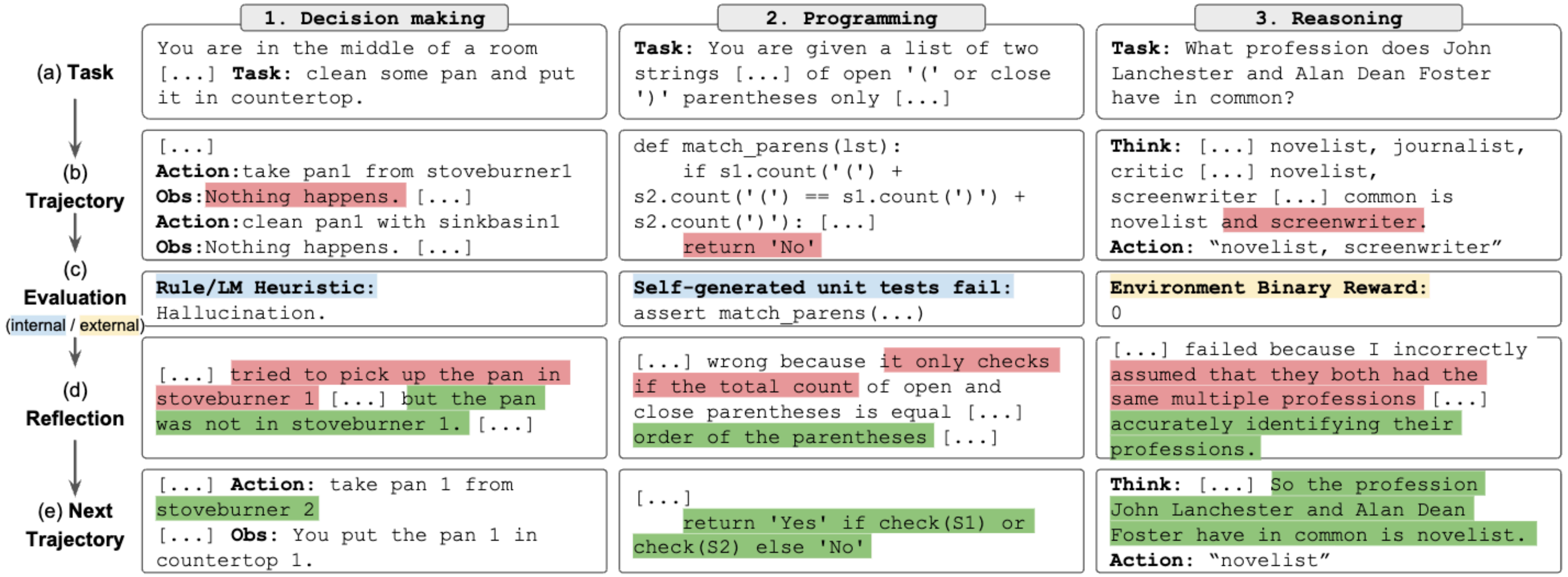
ReAct: Synergizing Reasoning and Acting in Language Models, Yao etal 2022

Structured Dialogues



- Break down problem / tasks for LLM
  - Higher performance
  - Lots of human engineering / prompting

Reflexion

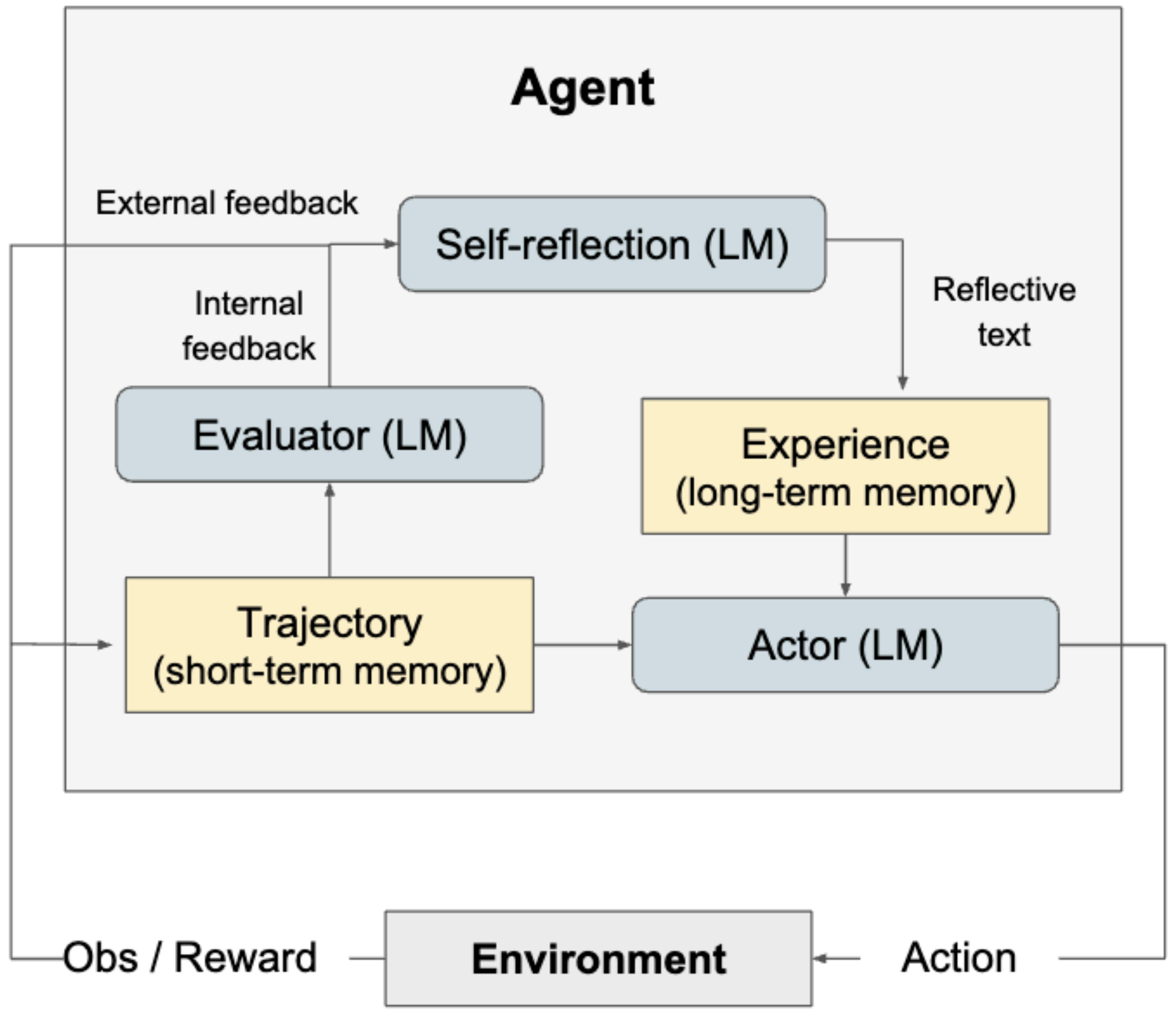


- Chain of Thought / ReACT
- Obtain observation / result
- Reflect on outcome
- Repeat

Reflexion: Language Agents with Verbal Reinforcement Learning, Shin etal 2023

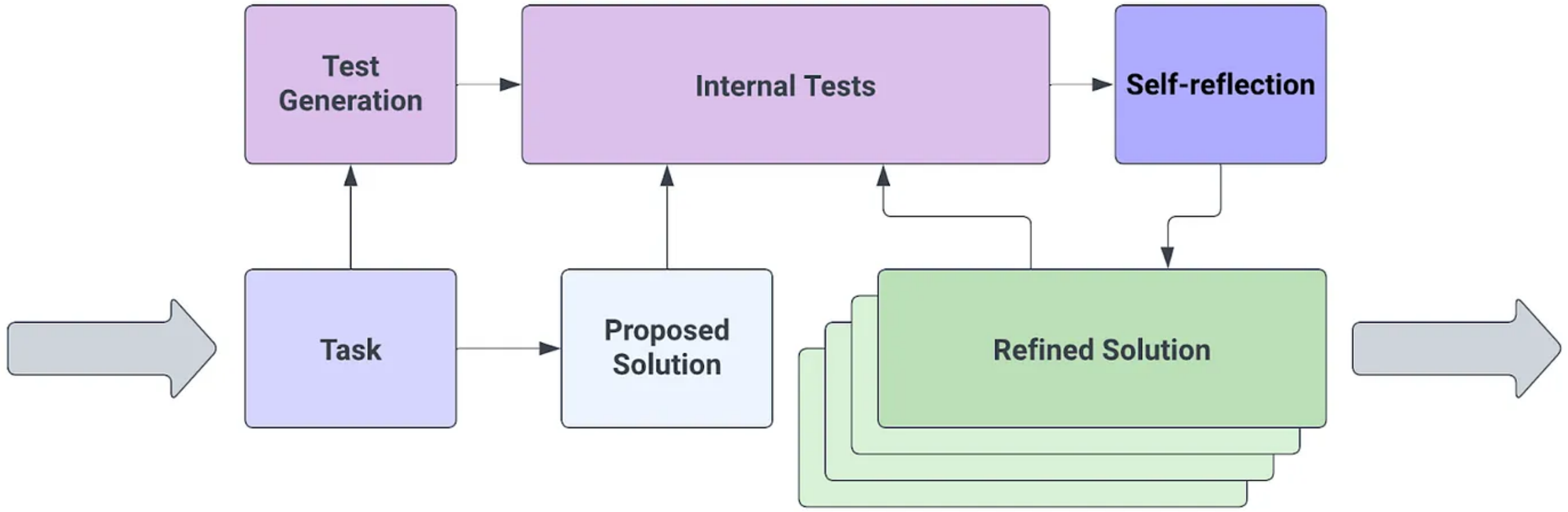
Reflexion

- Connections to reinforcement learning
  - More strictly planning
- Requires a evaluator (cost function)
  - External environment (i.e. simulator, code interpreter)
  - LLM generated tests
  - Trained LLM verifier [1]



Algorithm 1 Reinforcement via self-reflection

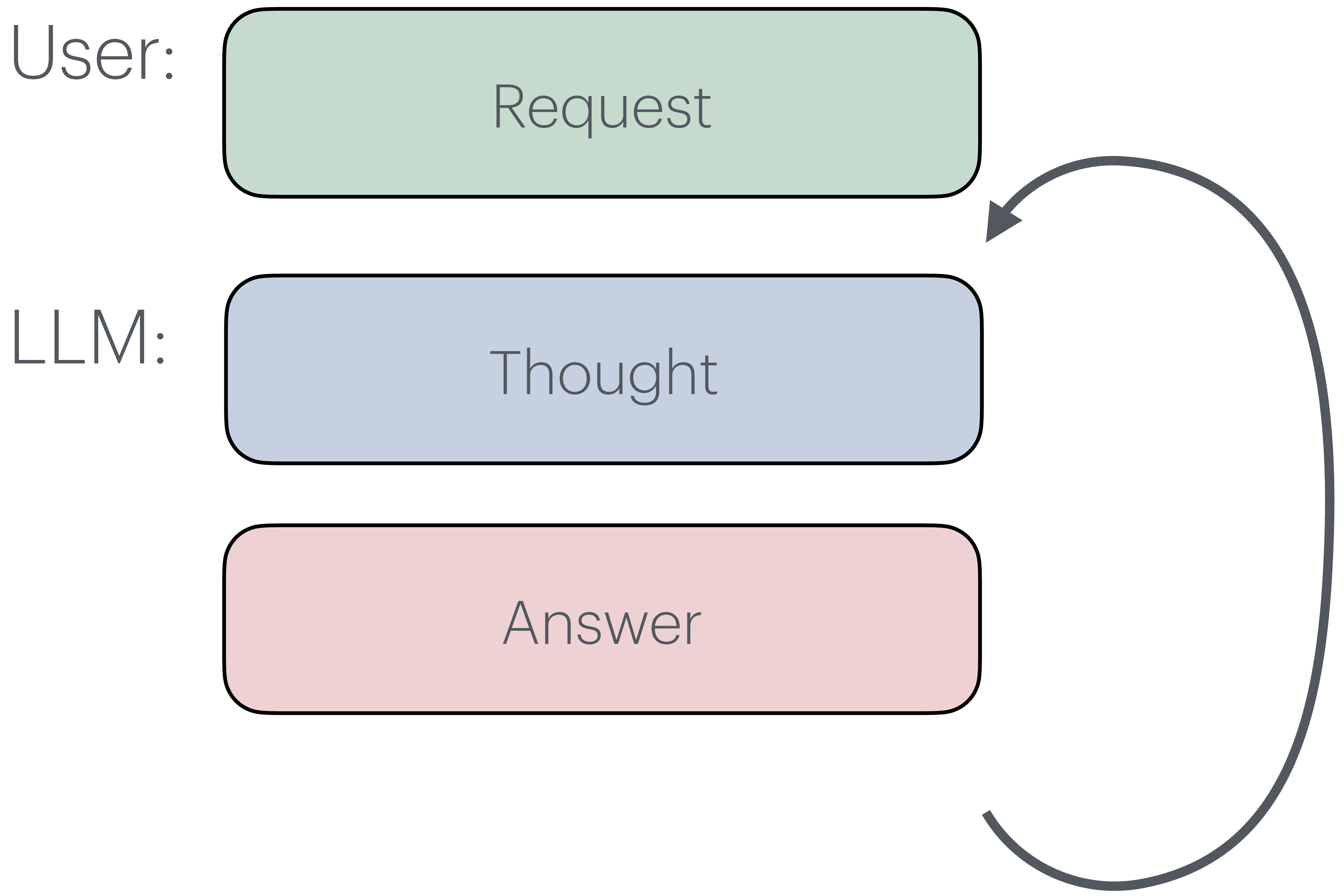
Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i)$, $\theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
  Generate $\tau_t = [a_0, o_0, \dots a_i, o_i]$ using π_θ
  Evaluate τ_t using M_e
  Generate self-reflection sr_t using M_{sr}
  Append sr_t to mem
  Increment t
end while
return



[1] Generative Verifiers: Reward Modeling as Next-Token Prediction, Zhang etal 2024

Reflexion

- Break down problem / tasks for LLM
  - Higher performance
  - Lots of human engineering / prompting



References

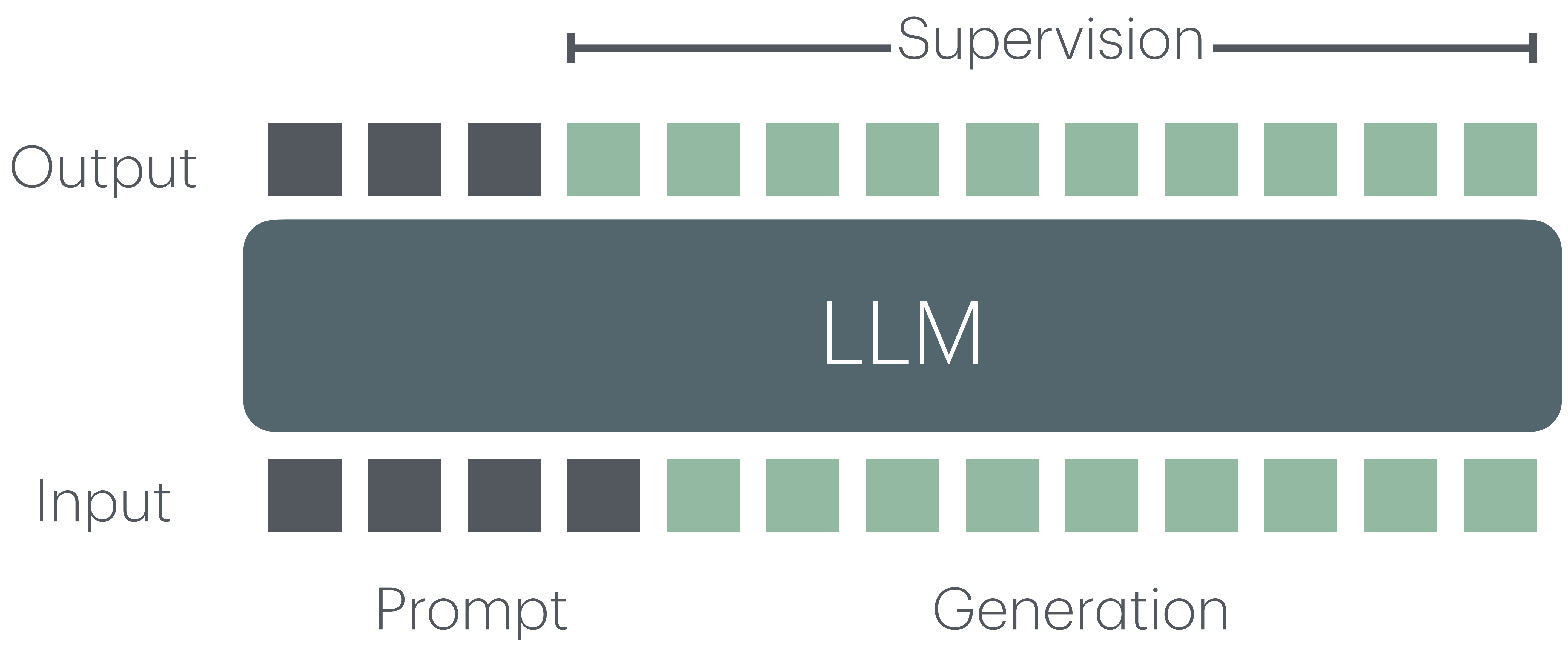
- [1] Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws, Allen-Zhu 2024
- [2] Vision Transformers Need Registers, Darcet etal 2023
- [3] Massive Activations in Large Language Models, Sun etal 2024
- [4] Language Models are Few-Shot Learners, Brown etal 2020
- [5] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei etal 2022
- [6] Self-Consistency Improves Chain of Thought Reasoning in Language Models, Wang etal 2022
- [7] Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao etal 2023
- [8] ReAct: Synergizing Reasoning and Acting in Language Models, Yao etal 2022
- [9] Reflexion: Language Agents with Verbal Reinforcement Learning, Shin etal 2023
- [10] Generative Verifiers: Reward Modeling as Next-Token Prediction, Zhang etal 2024

Reinforcement Learning and LLMs

Philipp Krähenbühl, UT Austin

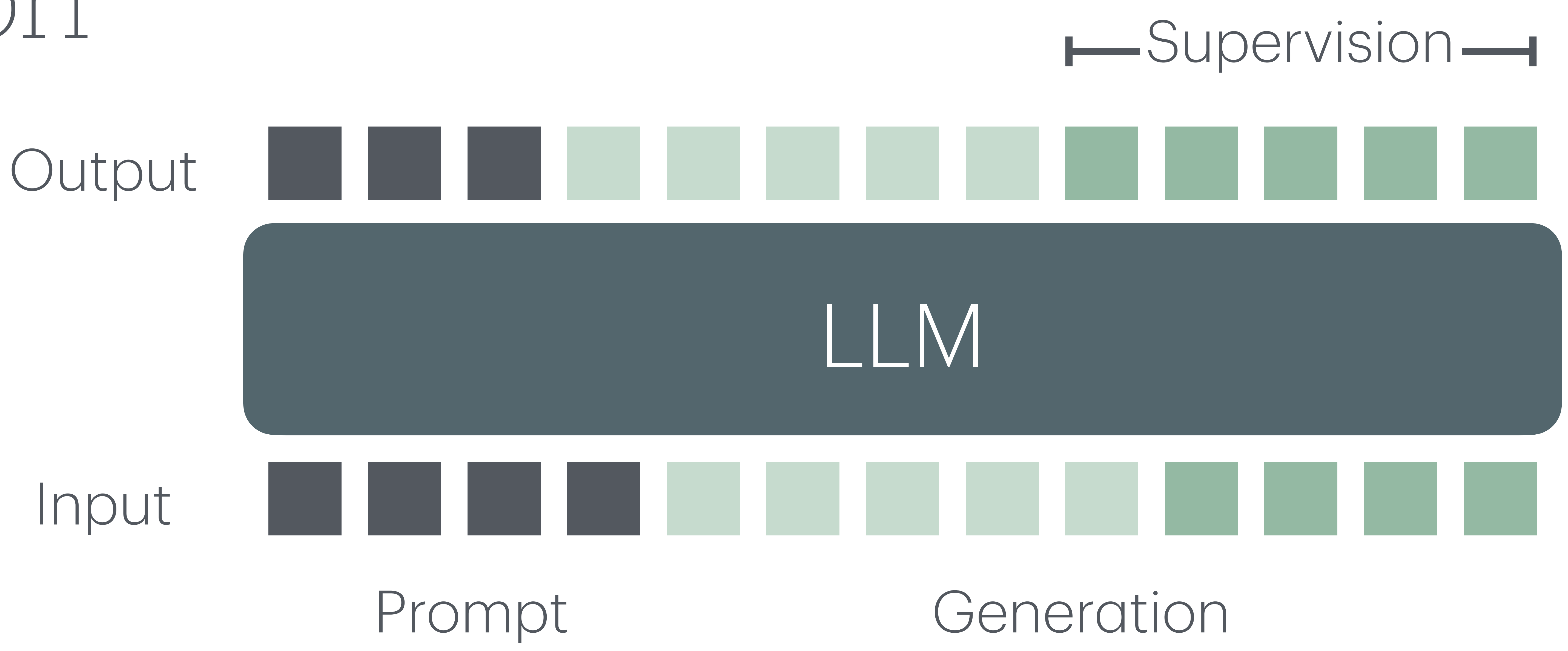
Teacher forcing

- Simple supervised learning
  - Input = Prompt + Target[0:-1]
  - Loss(output, Target[1:])



Outcome supervision

- What if we only supervise the final result?
- Generation
    - Loss(Generation)
- Teacher-forcing not possible
    - No supervised loss
- Solution: RL



Outcome supervision

Reinforcement Learning

- LLM $\quad p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$

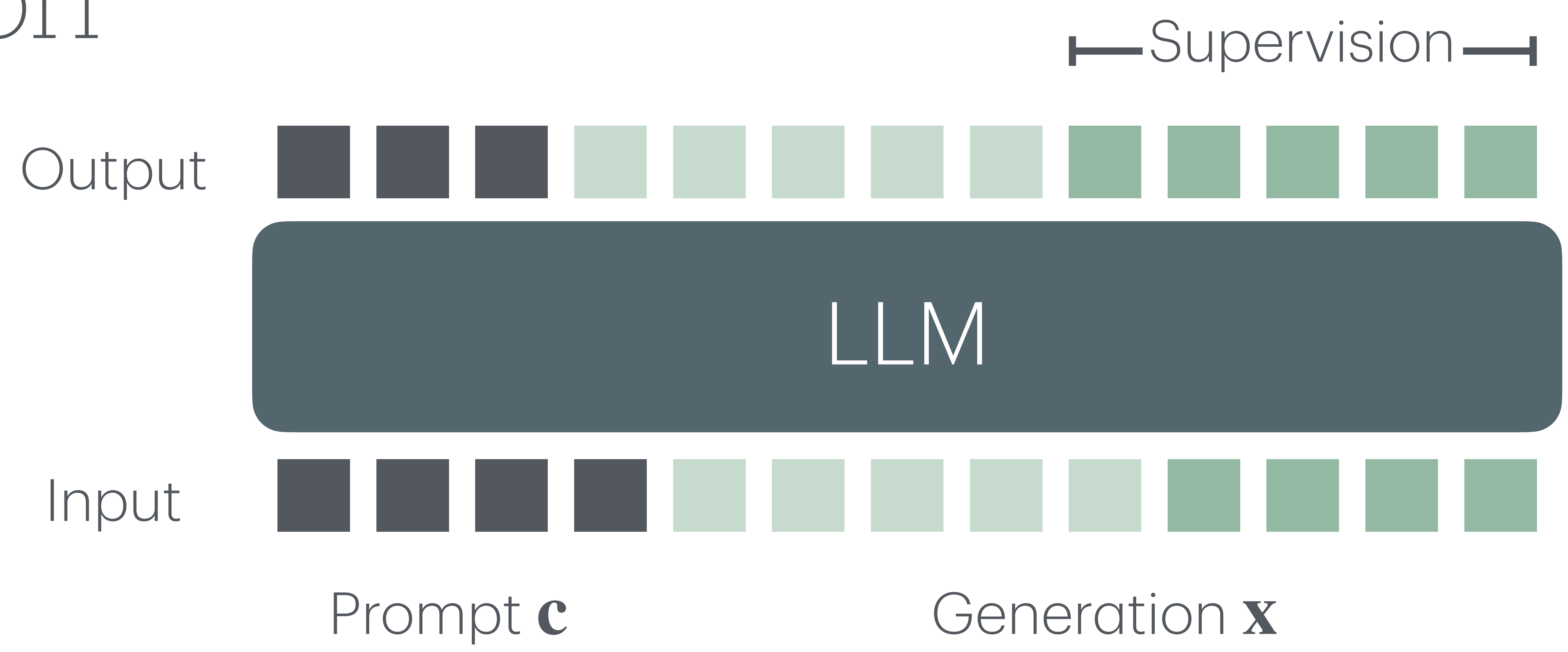
$$p_\theta(\mathbf{x} \mid \mathbf{c}) = \prod_{t=1}^{N} p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$$

- Sampling / Generation

$$x_{t+1} \sim p_\theta(\,\cdot \mid \mathbf{c}, x_1 \ldots x_t)$$

- MDP

$$E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})}\Bigg[\underbrace{\sum_{t=1}^{N} r(x_t \mid \mathbf{c}, x_1 \ldots x_{t-1})}_{R(\mathbf{c},\mathbf{x})}\Bigg]$$



REINFORCE

maximize $E_{\mathbf{x} \sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\right]$

- Using gradient ascent

  $$\nabla_\theta E_{\mathbf{x} \sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\right] = E_{\mathbf{x} \sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\, \nabla_\theta \log p_\theta(\mathbf{x} \mid \mathbf{c})\right]$$

- With a Monte-Carlo estimate

  $$\nabla_\theta E_{\mathbf{x} \sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\right] \approx \frac{1}{K} \sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)\, \nabla_\theta \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

  for $\mathbf{x}_k \sim p_\theta(\,\cdot \mid \mathbf{c})$

- REINFORCE K=1 works!!!

Initialize θ

for ever:

  Sample (or iterate over) $\mathbf{c}$

  $\mathbf{x} \sim p_\theta(\,\cdot \mid \mathbf{c})$

  $\theta \leftarrow \theta + \epsilon R(\mathbf{c}, \mathbf{x})\, \nabla \log p_\theta(\mathbf{x} \mid \mathbf{c})$

Policy Gradient

$$E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[A(\mathbf{c},\mathbf{x})\,\nabla_\theta \log p_\theta(\mathbf{x}\,|\,\mathbf{c})\right]$$

- Even better

$$E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[\sum_{t=1}^{T} A(\mathbf{c},x_1\ldots x_t)\,\nabla_\theta \log p_\theta(x_t\,|\,\mathbf{c},x_1\ldots x_{t-1})\right]$$

Algorithm 1 Vanilla Policy Gradient Algorithm

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0

2: **for** $k = 0, 1, 2, \ldots$ **do**

3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

4: Compute rewards-to-go $\hat{R}_t$.

5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k}\hat{A}_t.$$

7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\left(V_\phi(s_t) - \hat{R}_t\right)^2,$$

typically via some gradient descent algorithm.

9: **end for**

Credit: OpenAI

Proximal Policy Optimization

PPO

- Policy gradient
- Reuse rollouts (go slightly off-policy)
  - Basic Option: Importance weighting

    maximize $E_{\mathbf{x} \sim p_\psi(\cdot|\mathbf{c})} \left[\frac{p_\theta(\mathbf{x}\,|\,\mathbf{c})}{p_\psi(\mathbf{x}\,|\,\mathbf{c})} A(\mathbf{c}, \mathbf{x}) \right]$

  - Better Option: PPO-Clipping

    maximize $E_{\mathbf{x} \sim p_\psi(\cdot|\mathbf{c})} \left[\frac{1}{T} \sum_{t=1}^{T} CLIP\left(\frac{p_\theta(x_t\,|\,\mathbf{c}, x_1 \ldots x_{t-1})}{p_\psi(x_t\,|\,\mathbf{c}, x_1 \ldots x_{t-1})}, A(\mathbf{c}, \mathbf{x}) \right) \right]$

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Credit: OpenAI

REINFORCE vs PPO

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x} \sim p_\theta(\,\cdot\,|\,\mathbf{c})$

$\theta \leftarrow \theta + \epsilon R(\mathbf{c}, \mathbf{x}) \nabla \log p_\theta(\mathbf{x}\,|\,\mathbf{c})$

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

8: **end for**

Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs

Arash Ahmadian
Cohere For AI

Chris Cremer
Cohere

Matthias Gallé
Cohere

Marzieh Fadaee
Cohere For AI

Julia Kreutzer
Cohere For AI

Olivier Pietquin
Cohere

Ahmet Üstün
Cohere For AI

Sara Hooker
Cohere For AI

{arash,olivier,ahmet,sarahooker}@cohere.com

Outcome supervision

Reinforcement Learning

- LLM $p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$

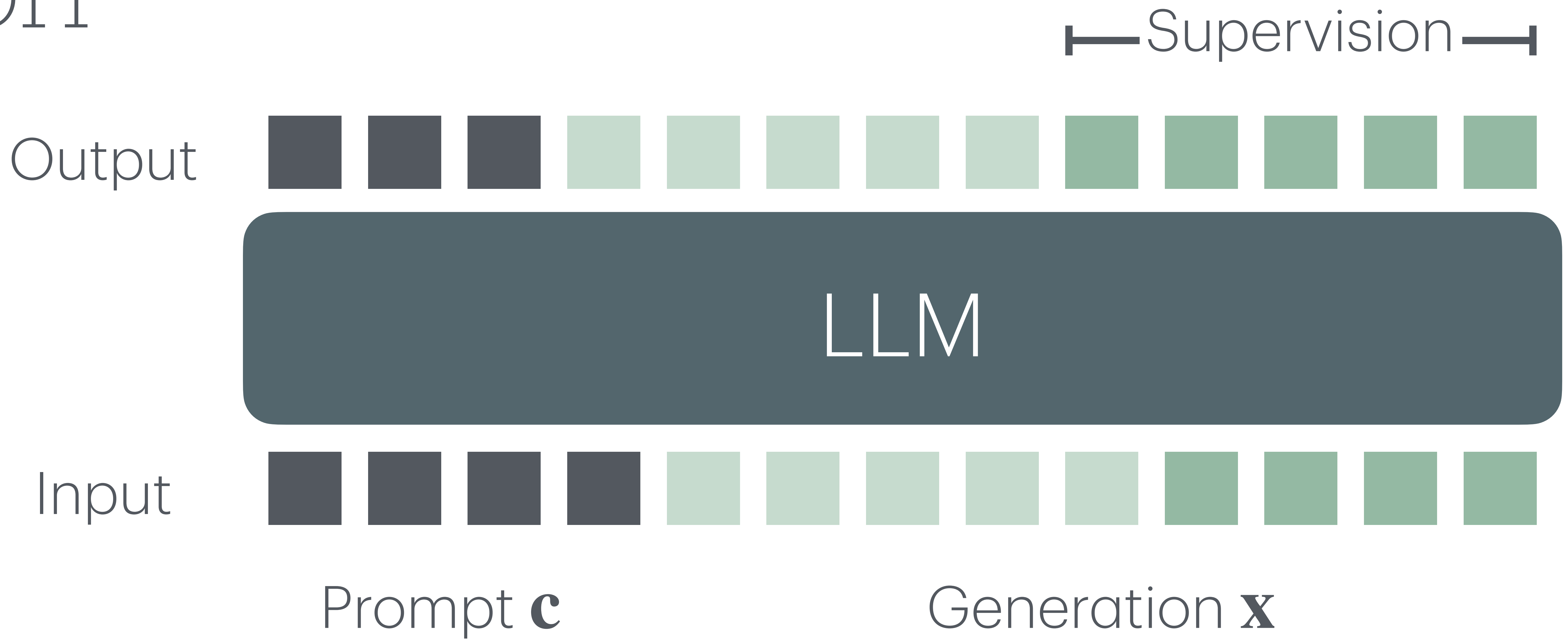
$$p_\theta(\mathbf{x} \mid \mathbf{c}) = \prod_{t=1}^{N} p_\theta(x_{t+1} \mid \mathbf{c}, x_1 \ldots x_t)$$

- Sampling / Generation

$$x_{t+1} \sim p_\theta(\cdot \mid \mathbf{c}, x_1 \ldots x_t)$$

- MDP

$$E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})} \left[\sum_{t=1}^{N} r(x_t \mid \mathbf{c}, x_1 \ldots x_{t-1}) \right]$$



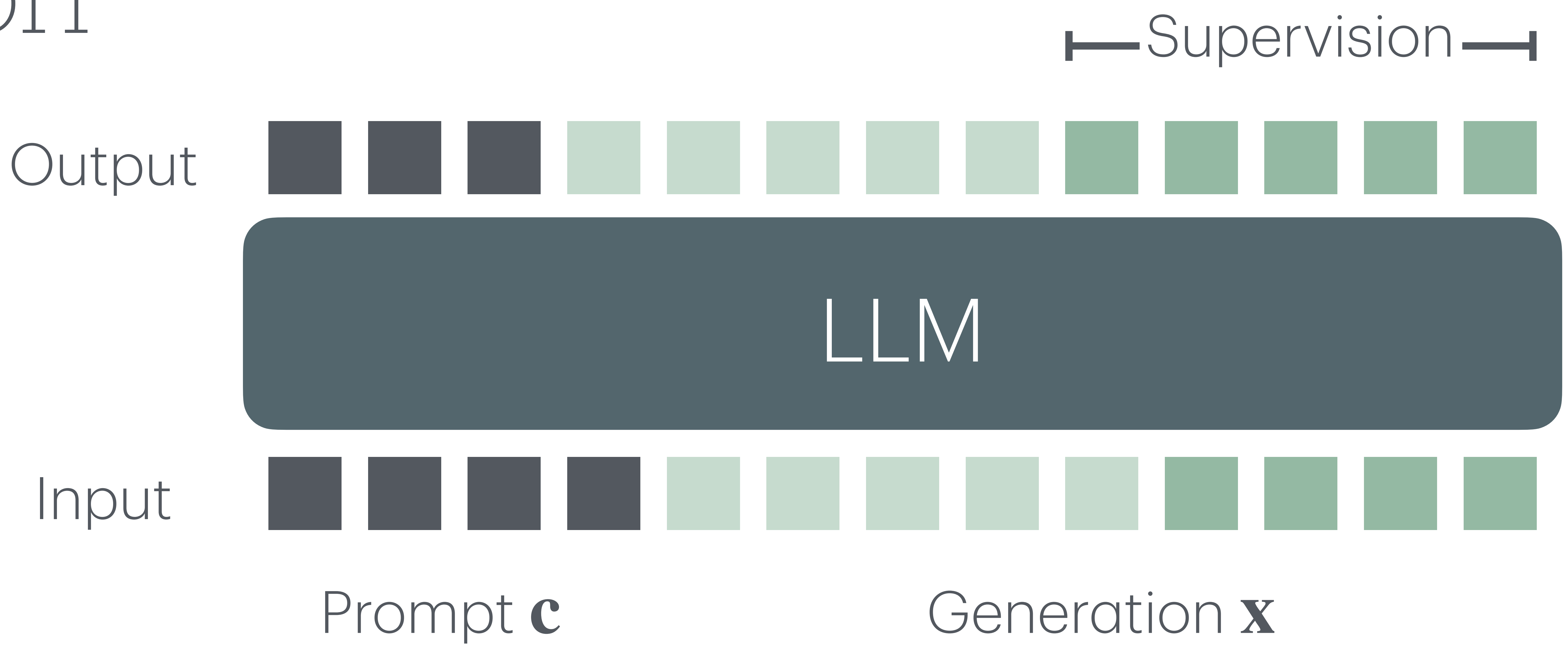
Outcome supervision

Reinforcement Learning

- LLM $p_\theta(\mathbf{x} \mid \mathbf{c})$
- Sampling / Generation

  $\mathbf{x} \sim p_\theta(\,\cdot \mid \mathbf{c})$
- Contextual bandit

  $E_{\mathbf{x} \sim p_\theta(\cdot \mid \mathbf{c})}\left[R(\mathbf{c}, \mathbf{x})\right]$



REINFORCE Leave One Out

- $\nabla_\theta E_{\mathbf{x}\sim p_\theta(\cdot|\mathbf{c})}\left[R(\mathbf{c},\mathbf{x})\right] \approx \frac{1}{K}\sum_{k=1}^{K}\left(R(\mathbf{c},\mathbf{x}_k) - b(\mathbf{c})\right)\nabla_\theta \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$

  for $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$

- $b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c},\mathbf{x}_k)$

Initialize θ

for ever:

  Sample (or iterate over) $\mathbf{c}$

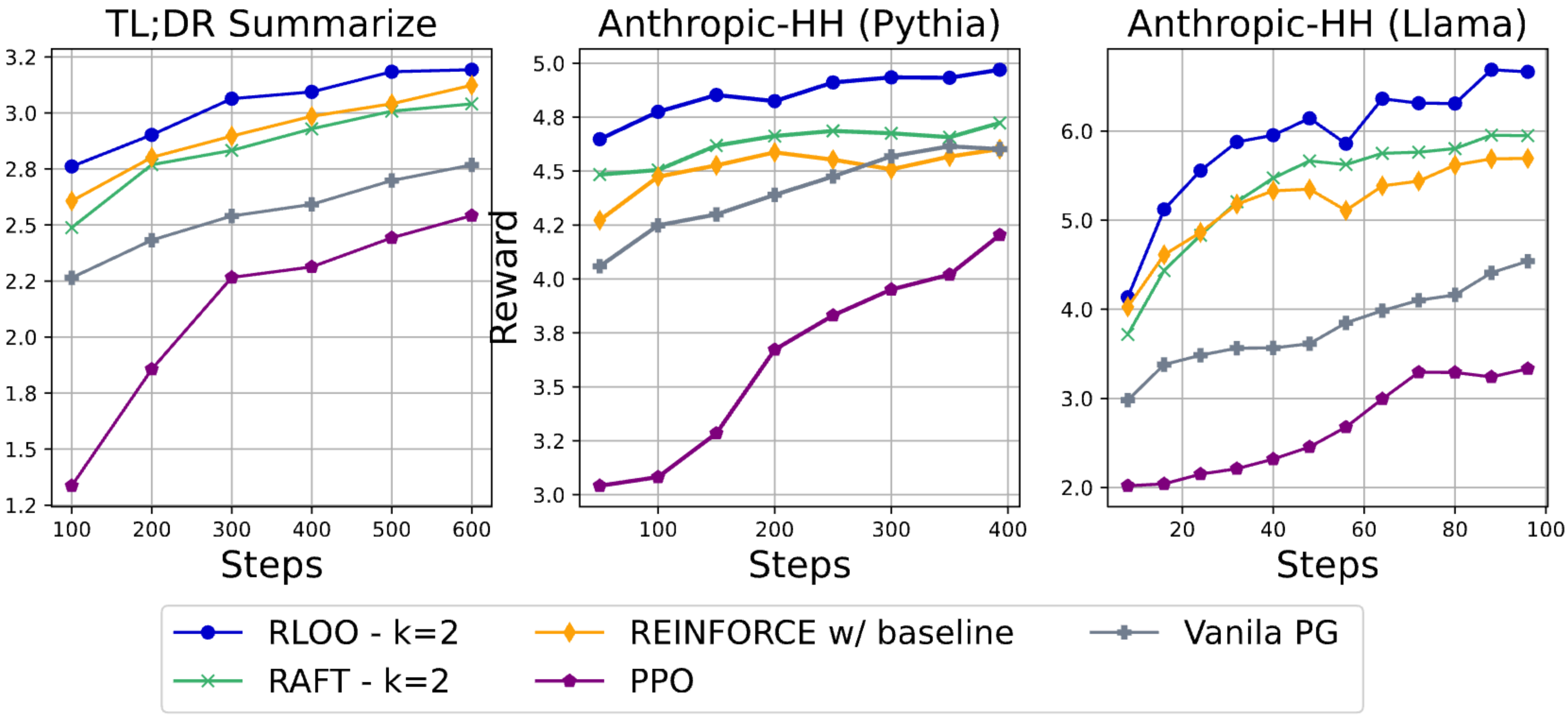
  $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$ for k=1…K

  $b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c},\mathbf{x}_k)$

  $\theta \leftarrow \theta + \epsilon\frac{1}{K}\sum_{k=1}^{K}\left(R(\mathbf{c},\mathbf{x}_k) - b(\mathbf{x})\right)\nabla \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$

Buy 4 REINFORCE Samples, Get a Baseline for Free!, Kool etal 2019

RLOO in LLMs



Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs, Ahmadian etal 2024

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

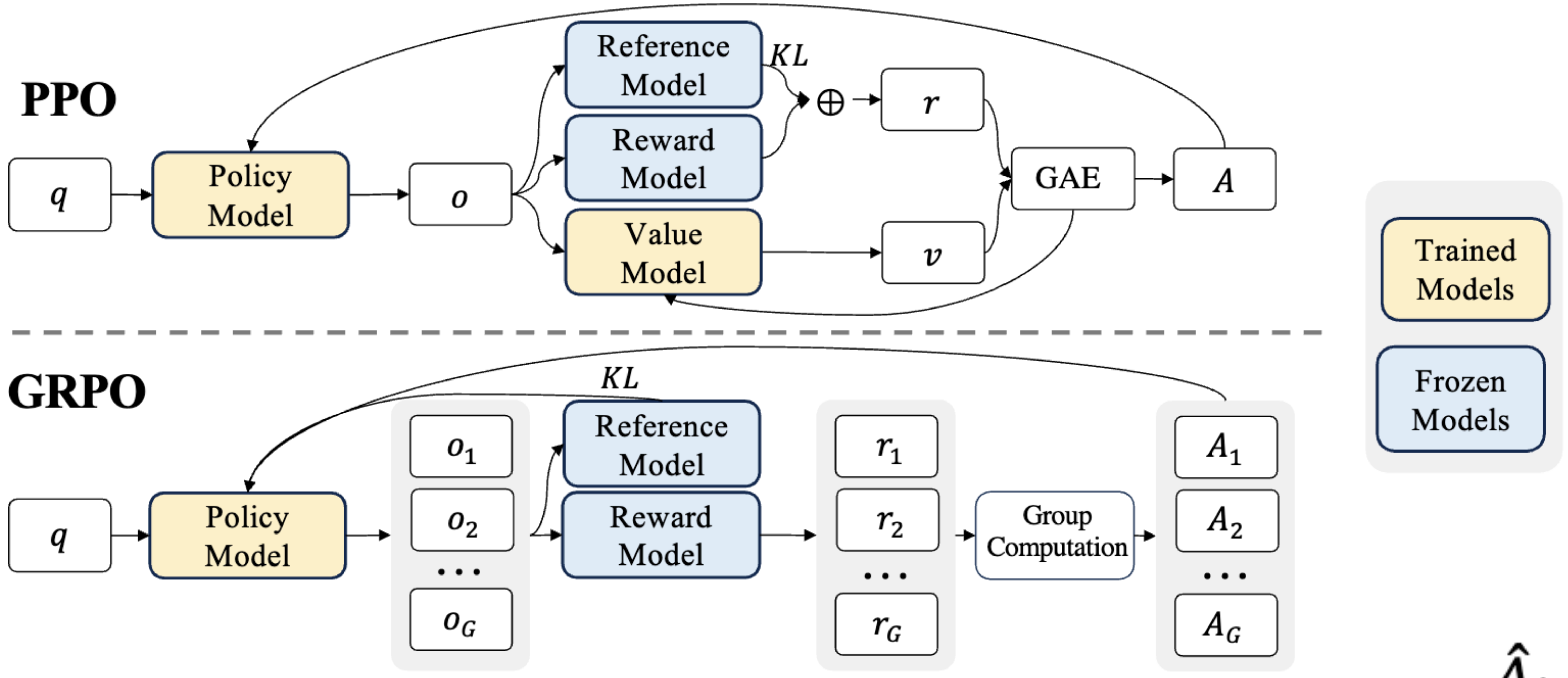
Zhihong Shao[1,2*†], Peiyi Wang[1,3*†], Qihao Zhu[1,3*†], Runxin Xu[1], Junxiao Song[1]
Xiao Bi[1], Haowei Zhang[1], Mingchuan Zhang[1], Y.K. Li[1], Y. Wu[1], Daya Guo[1*]

[1]DeepSeek-AI, [2]Tsinghua University, [3]Peking University

{zhihongshao,wangpeiyi,zhuqh,guoday}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Math

GRPO

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min\left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1-\varepsilon, 1+\varepsilon\right) A_t\right],$$



$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right]\right\},$$

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

Algorithm 1 Iterative Group Relative Policy Optimization

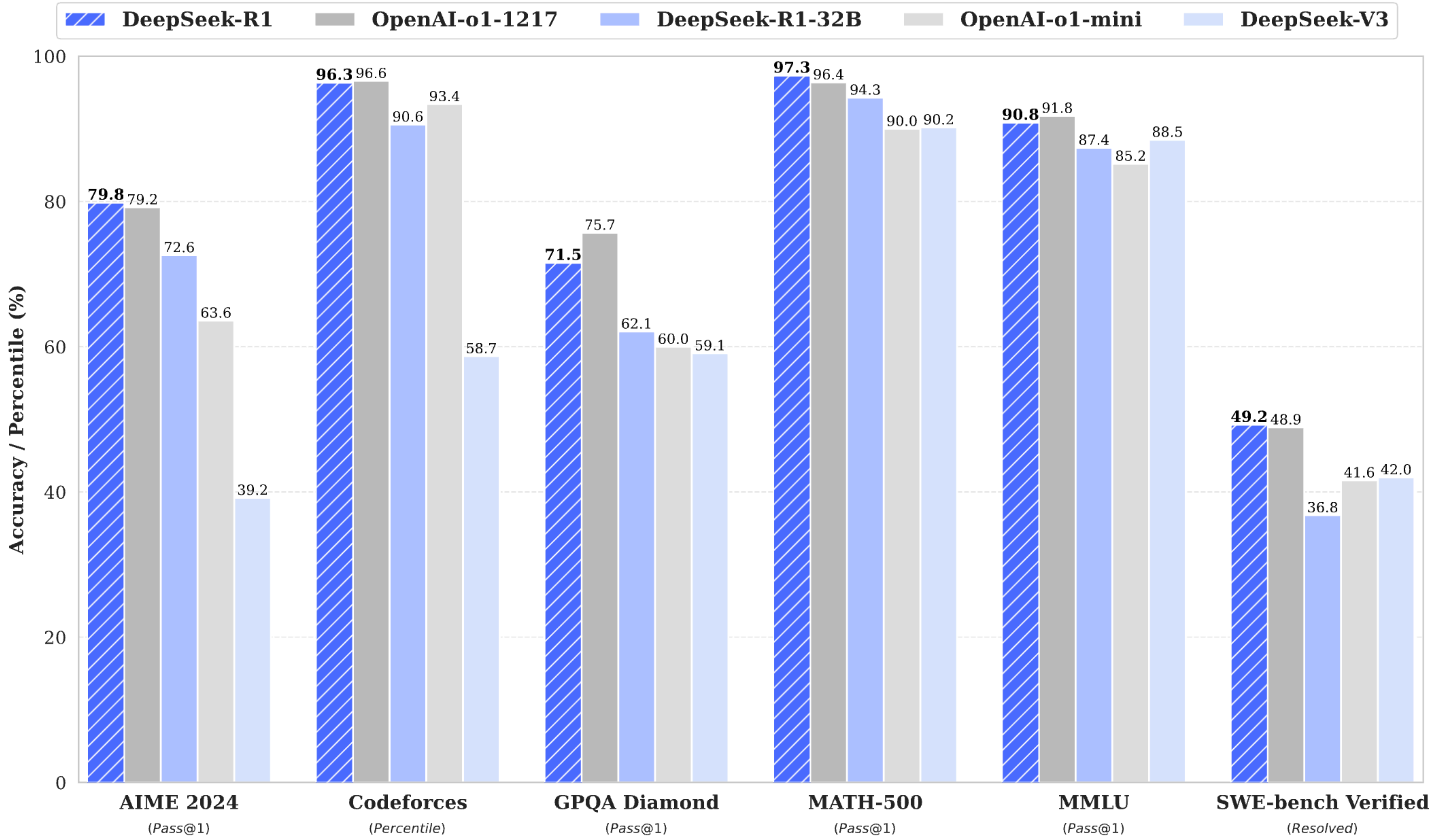
Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, . . . , I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, . . . , M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        for GRPO iteration = 1, . . . , μ **do**
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

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



DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Deepseek team 2025

Step 1: Create a dataset of math puzzles and alike

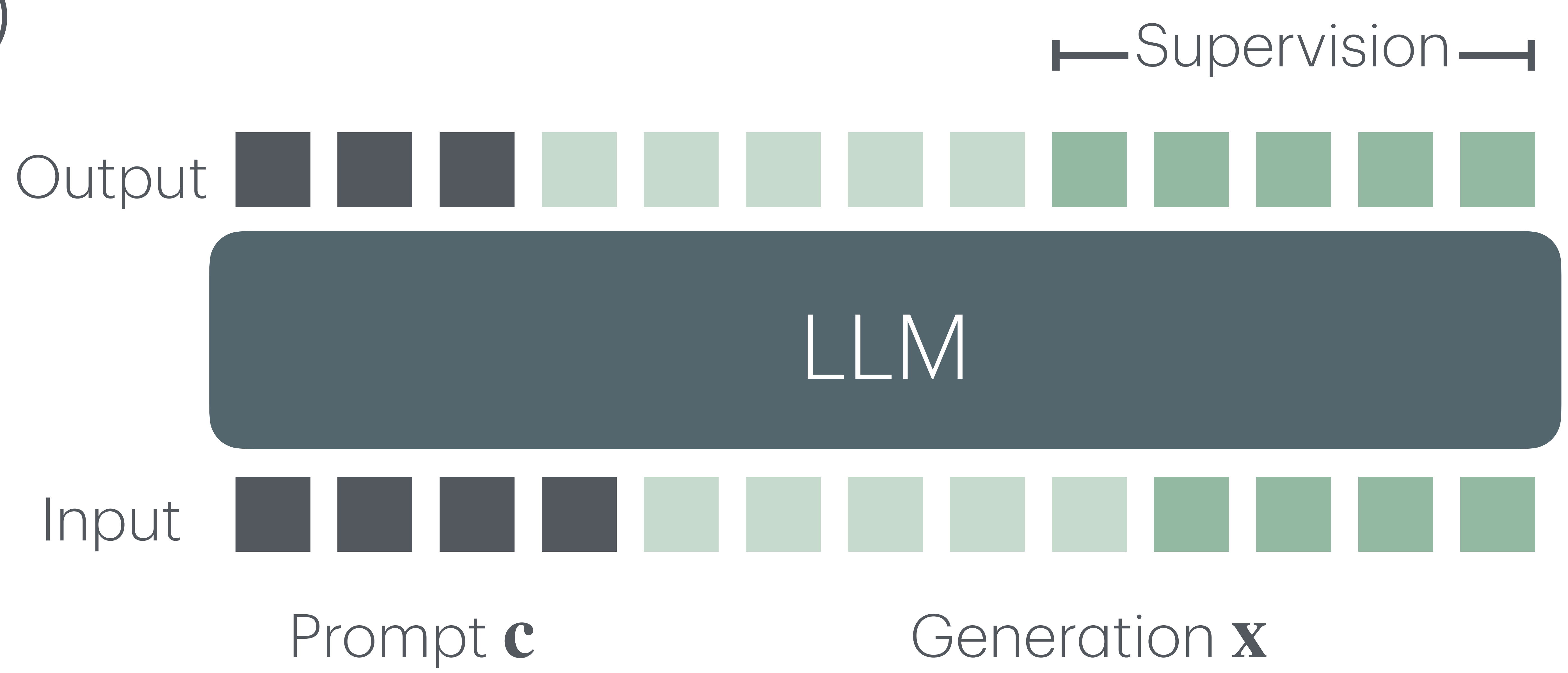
- Interesting prompts
- Easily verifiable answers
  - Math, Reasoning, Multiple-choice, …
- No details given in paper

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

$$\frac{-1 \pm \sqrt{1 + 4a}}{2}$$

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Deepseek team 2025

Step 2: Run GRPO



- No teacher forcing
- Supervise just easily verifiable answers
- Learns reasoning
- R1-Zero
  - From just pre-trained model, no instruction tuning
  - It works

Step 3: Bootstrap Instruction tuned model

- Use instruction tuning data and R1-Zero data
- Train a chat-bot that can "reason"

	Benchmark (Metric)	**Claude-3.5-Sonnet-1022**	**GPT-4o 0513**	**DeepSeek V3**	**OpenAI o1-mini**	**OpenAI o1-1217**	**DeepSeek R1**
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	**91.8**	90.8
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-	**92.9**
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-	**84.0**
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	**92.2**
	IF-Eval (Prompt Strict)	**86.5**	84.3	86.1	84.8	-	83.3
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	**75.7**	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	**47.0**	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	**82.5**
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-	**87.6**
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-	**92.3**
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	63.4	**65.9**
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	**96.6**	96.3
	Codeforces (Rating)	717	759	1134	1820	**2061**	2029
	SWE Verified (Resolved)	**50.8**	38.8	42.0	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	**61.7**	53.3
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	79.2	**79.8**
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	96.4	**97.3**
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	**78.8**
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-	**92.8**
	C-Eval (EM)	76.7	76.0	86.5	68.9	-	**91.8**
	C-SimpleQA (Correct)	55.4	58.7	**68.0**	40.3	-	63.7

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, . . . , I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, . . . , M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        for GRPO iteration = 1, . . . , μ **do**
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

Algorithm 1 Iterative Group Relative Policy Optimization

Input initial policy model $\pi_{\theta_{\text{init}}}$; reward models r_φ; task prompts $\mathcal{D}$; hyperparameters ε, β, μ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration = 1, . . . , I **do**
    reference model $\pi_{ref} \leftarrow \pi_\theta$
    for step = 1, . . . , M **do**
        Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
        Update the old policy model $\pi_{\theta_{old}} \leftarrow \pi_\theta$
        Sample G outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$
        Compute rewards $\{r_i\}_{i=1}^G$ for each sampled output o_i by running r_φ
        Compute $\hat{A}_{i,t}$ for the t-th token of o_i through group relative advantage estimation.
        ~~**for** GRPO iteration = 1, . . . , μ **do**~~
            Update the policy model π_θ by maximizing the GRPO objective (Equation 21)
    Update r_φ through continuous training using a replay mechanism.

Output π_θ

$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

max length is set to 1024, and the training batch size is 1024. The policy model only has a single update following each exploration stage. We evaluate DeepSeekMath-RL 7B on benchmarks

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \cancel{\text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right]\right\},$$

GRPO = RLOO with advantage normalization

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal 2024

RLOO vs GRPO

RLOO

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x}_k \sim p_\theta(\cdot \mid \mathbf{c})$ for k=1…K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} (R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x}))\, \nabla \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

GRPO (in practice)

Initialize θ

for ever:

Sample (or iterate over) $\mathbf{c}$

$\mathbf{x}_k \sim p_\theta(\cdot \mid \mathbf{c})$ for k=1…K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} \frac{R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x})}{\mathrm{std}(\mathbf{x})}\, \nabla \log p_\theta(\mathbf{x}_k \mid \mathbf{c})$$

Interactive Digital Agents

- Train LLMs that interact with API's on the users behalf

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen ... Krähenbühl 2025

Interactive Digital Agents

LOOP

- Use simulator for API interactions
  - AppWorld
- Training data = 24 scenarios (simple request, initial simulator state, test cases)
- Trained using simple combination of Leave-One-Out estimator and PPO

Algorithm 1 Leave-One-Out Proximal Policy Optimization

Input: Policy p_θ, dataset of tasks and initial states $\mathcal{D}$
Output: Policy p_θ maximizing $\mathbb{E}_{\mathbf{s}_0,\mathbf{c}\sim\mathcal{D}}\left[L_\theta(\mathbf{s}_0,\mathbf{c})\right]$ (Eq. 7)

```
for iteration = 1, 2, ... do
    B ← {}                                                   ▷ Initialize rollout buffer
    for (s_0, c) ~ D do                                      ▷ Rollout collection
        Collect K rollouts x_1, ..., x_K ~(i.i.d.) ρ_θ(·|s_0, c)
        Estimate advantages A_1, ..., A_K using Eq. 3
        B ← B ∪ {(x_1, A_1), ..., (x_K, A_K)}
    for epoch = 1, ..., N_epoch do                           ▷ Policy update
        for mini-batch {(x_i, A_i)}_{i=1}^M ~ B do
            Update policy using PPO gradient (Eq. 5)
```

$$A(\mathbf{c},\mathbf{x}_k)=\frac{K}{K-1}\left(R(\mathbf{c},\mathbf{x}_k)-\frac{1}{K}\sum_{i=1}^{K}R(\mathbf{c},\mathbf{x}_i)\right). \quad (3)$$

$$L_\theta^{\mathrm{MDP}}(\mathbf{c})=\mathbb{E}_{\mathbf{x}\sim p_\psi(\cdot|\mathbf{c})}\left[\frac{1}{|\mathbf{x}|}\sum_{t=1}^{|\mathbf{x}|}\min\left(\frac{p_\theta(x_t|\mathbf{c},x_{1:t-1})}{p_\psi(x_t|\mathbf{c},x_{1:t-1})}A(\mathbf{c},\mathbf{x}),\, g_\epsilon(A(\mathbf{c},\mathbf{x}))\right)\right]. \quad (5)$$

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

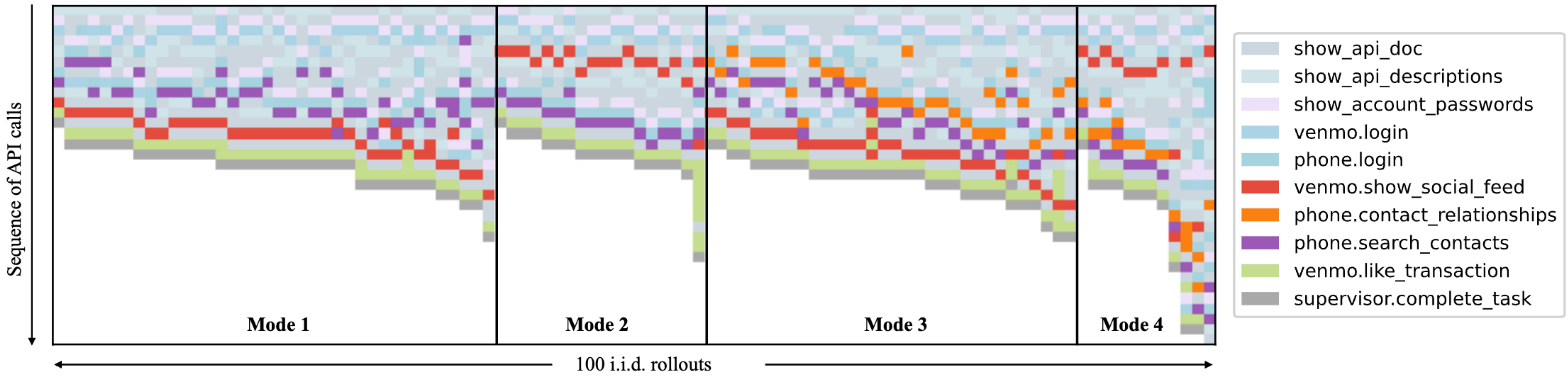
Type	Algorithm	Action	Strictly on-policy	Normalized reward	Test Normal (Test-N)		Test Challenge (Test-C)	
					TGC	SGC	TGC	SGC
NFT	GPT-4o	–	–	–	48.8	32.1	30.2	13
NFT	OpenAI o1	–	–	–	61.9	41.1	36.7	19.4
NFT	Llama 3 70B	–	–	–	24.4	17.9	7.0	4.3
NFT	Qwen 2.5 32B	–	–	–	39.2 ± 3.5	18.6 ± 2.0	21.0 ± 1.4	7.5 ± 1.2
SFT	SFT-GT	–	–	–	6.2 ± 0.7	1.8 ± 0.0	0.8 ± 0.2	0.1 ± 0.3
SFT	RFT	–	–	–	47.9 ± 3.7	26.4 ± 2.3	26.4 ± 1.8	11.4 ± 2.3
SFT	EI	–	–	–	58.3 ± 2.8	36.8 ± 6.0	32.8 ± 0.7	17.6 ± 1.3
DPO	DPO-MCTS	–	–	–	57.0 ± 1.5	31.8 ± 4.2	31.8 ± 1.3	13.7 ± 1.5
DPO	DMPO	–	–	–	59.0 ± 1.2	36.6 ± 4.7	36.3 ± 1.8	18.4 ± 2.3
RL	PPO (learned critic)	token			50.8 ± 3.7	28.9 ± 7.9	26.4 ± 0.5	10.5 ± 2.1
RL	RLOO	traj	✓		57.2 ± 2.6	35.7 ± 2.9	36.7 ± 1.6	17.4 ± 1.4
RL	GRPO	token	✓[3]	✓	58.0 ± 1.8	36.8 ± 3.9	39.5 ± 1.9	22.4 ± 0.8
RL	GRPO no kl	token	✓[3]	✓	59.0 ± 1.4	35.7 ± 2.9	42.7 ± 1.3	21.3 ± 1.7
RL	LOOP (bandit)	traj			53.3 ± 3.4	33.6 ± 3.2	27.7 ± 1.5	13.0 ± 0.9
RL	LOOP (turn)	turn			64.1 ± 2.2	43.5 ± 3.5	40.8 ± 1.5	26.5 ± 2.4
RL	**LOOP (token)**	token			$\mathbf{71.3 \pm 1.3}$	$\mathbf{53.6 \pm 2.2}$	$\mathbf{45.7 \pm 1.3}$	$\mathbf{26.6 \pm 1.5}$
RL	LOOP RwNorm (token)	token		✓	61.9 ± 4.0	44.1 ± 7.8	39.8 ± 1.3	20.4 ± 2.1

RL vs SFT

- Generations from LLM are very diverse even after RL training

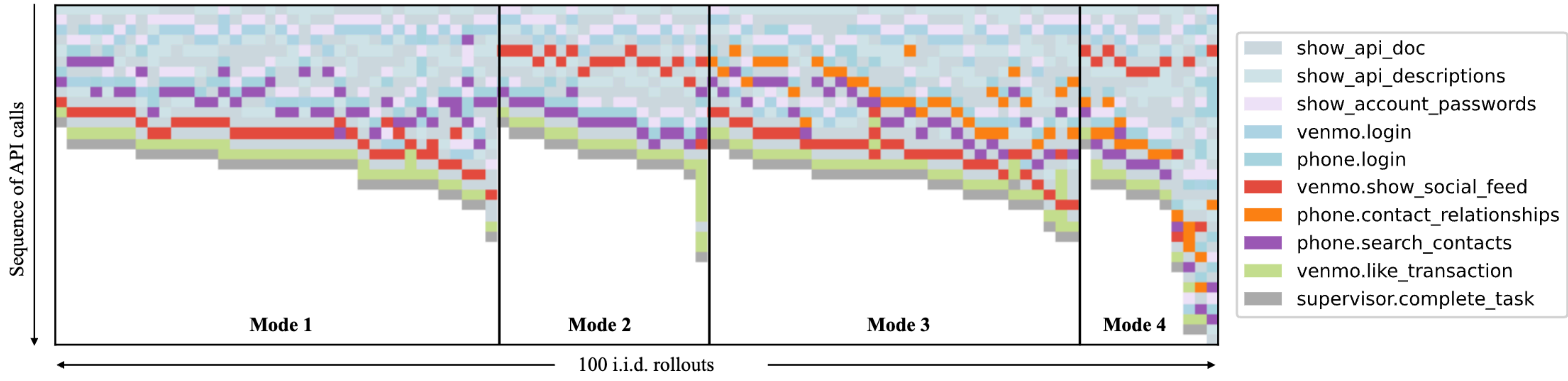
Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

RL vs SFT



Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

RL vs SFT



- Generations from LLM are very diverse even after RL training
  - 98 / 100 solutions are correct
  - Only 3 repeat overall structure of commands (no exact repetition)
- Early in training: Awesome exploration
- Late in training: No collapse / overfitting

Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen … Krähenbühl 2025

Reinforcement Learning and LLMs

- Easy to implement
  - RLOO, LOOP are just generation + reward computation + mean subtraction + training
  - Great open-source tools exist for all of this
- Much more flexible
- Less data-hungry
- It is here to stay

Initialize θ

for ever:

  Sample (or iterate over) $\mathbf{c}$

  $\mathbf{x}_k \sim p_\theta(\,\cdot\,|\,\mathbf{c})$ for k=1...K

$$b(\mathbf{c}) = \frac{1}{K}\sum_{k=1}^{K} R(\mathbf{c}, \mathbf{x}_k)$$

$$\theta \leftarrow \theta + \epsilon \frac{1}{K}\sum_{k=1}^{K} (R(\mathbf{c}, \mathbf{x}_k) - b(\mathbf{x}))\,\nabla \log p_\theta(\mathbf{x}_k\,|\,\mathbf{c})$$

References

- Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs, Ahmadian etal. 2024
- Buy 4 REINFORCE Samples, Get a Baseline for Free!, Kohl etal 2019
- DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, Shao etal. 2024
- DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, DeepSeek-AI 2025
- Reinforcement Learning for Long-Horizon Interactive LLM Agents, Chen etal 2025

Limitations of LLMs

Philipp Krähenbühl, UT Austin

Politics of LLM research

- Many different camps
  - With conflicting often hidden motives

Model Builders

Develop new models

Make $$$, fame, glory, (Invent AGI)

AI Safety research

Study limitations, biases, and dangers

Concerns about societal impacts of LLMs, fame

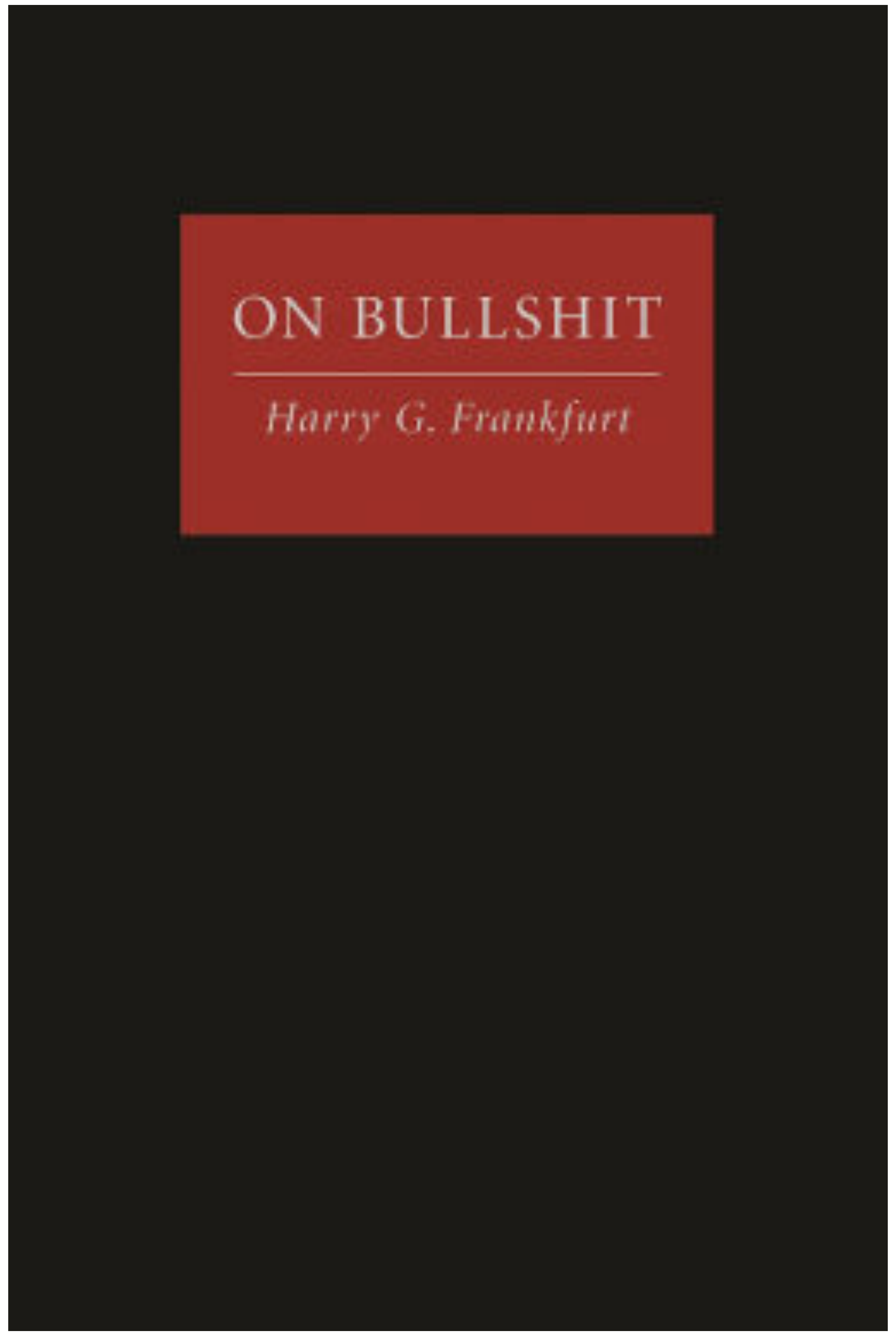
External Analyses

Bring tools from other sciences into LLM world

Study LLMs as "creatures", More scientific approach, fame

ChatGPT is bullshit

- LLMs generate falsehoods
  - AKA Hallucinations
- **Bullshit** (general): Any utterance produced where a speaker has indifference towards the truth of the utterance.
- **Hard** bullshit: Bullshit produced with the **intention to mislead** the audience about the utterer's agenda.
- **Soft** bullshit: Bullshit produced **without the intention to mislead** the hearer regarding the utterer's agenda.



Bullshitters misrepresent themselves to their audience not as liars do, that is, by deliberately making false claims about what is true. Rather, bullshitters seek to convey a certain impression of themselves without being concerned about whether anything at all is true. - Frankfurt

ChatGPT is bullshit, Hicks etal 2024

ChatGPT is bullshit

- “[LLMs] are in an important way indifferent to the truth of their outputs.”
  - Training: Reproduce plausible text
  - Human cognition: goals and behaviors
- Paper makes a few assumptions
  - Disregards alignment

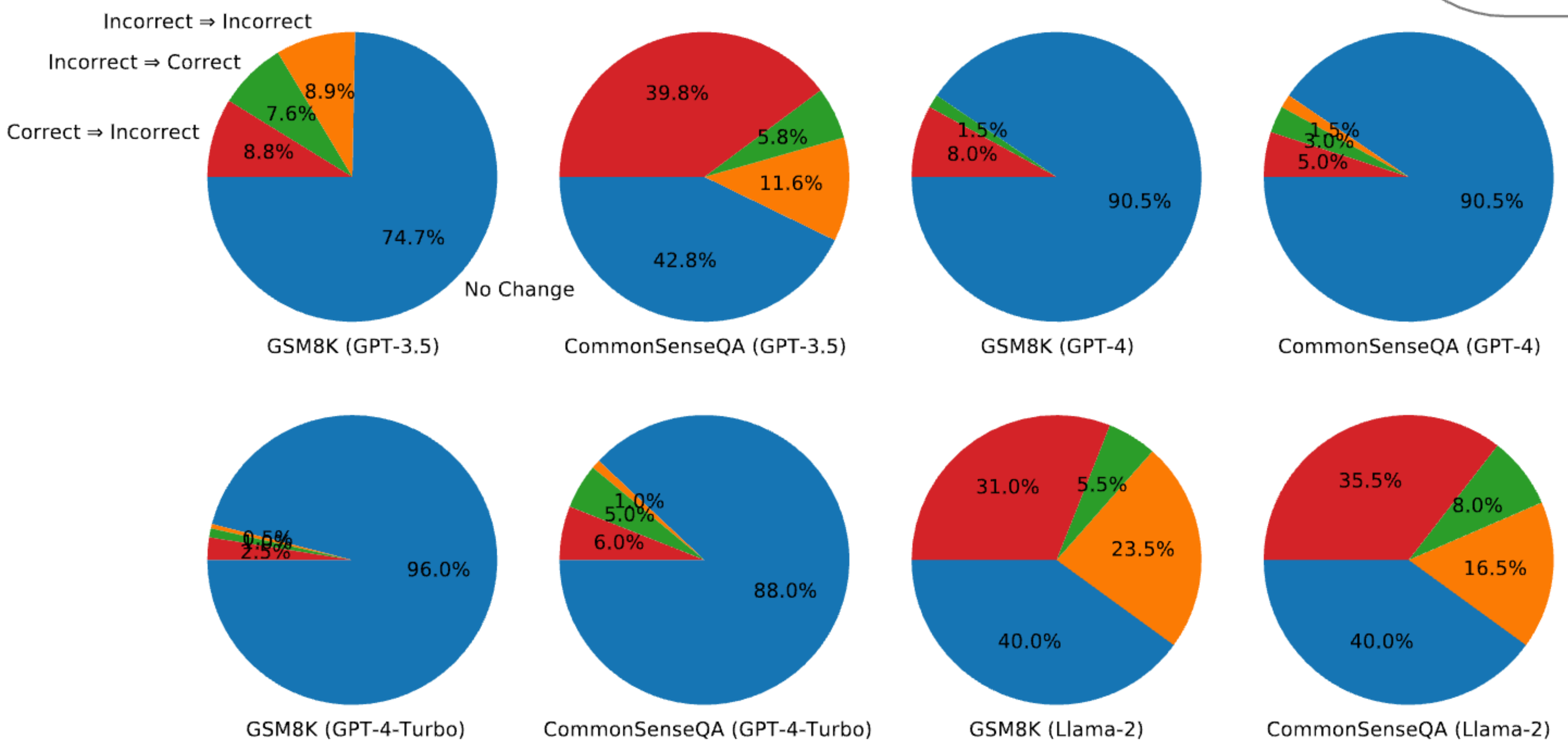
The problem here isn’t that large language models hallucinate, lie, or misrepresent the world in some way. It’s that <u>they are not designed to represent the world at all</u>; instead, they are designed to convey convincing lines of text.

ChatGPT is at minimum a soft bullshitter or a bullshit machine, because if it is not an agent then it can neither hold any attitudes towards truth nor towards deceiving hearers about its (or, perhaps more properly, its users’) agenda

ChatGPT is bullshit, Hicks etal 2024

No self-correction Yet

- LLMs rarely change their mind
  - If they do, they make things worse



Christina is planning a birthday party and needs .75 gift bags per invited guest, because 1/4 of attendees don't show up. She invited 16 friends. Gift bags are $2 each. How much will she spend?

... Therefore, Christina will spend $18 on gift bags. Answer: \boxed{18}. (Incorrect)

Review your previous answer and find problems with your answer.

Upon reviewing my previous answer, I recognize a mistake. I incorrectly stated that Christina needs 0.75 gift bags per invited guest. The given information states that 1/4 of the attendees won't show up…

Based on the problems you found, improve your answer.

... Therefore, Christina will spend $24 on gift bags. Answer: \boxed{24}. (Correct)

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 $\boxed{75.00}$ 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, resulting in a total of $75.00…

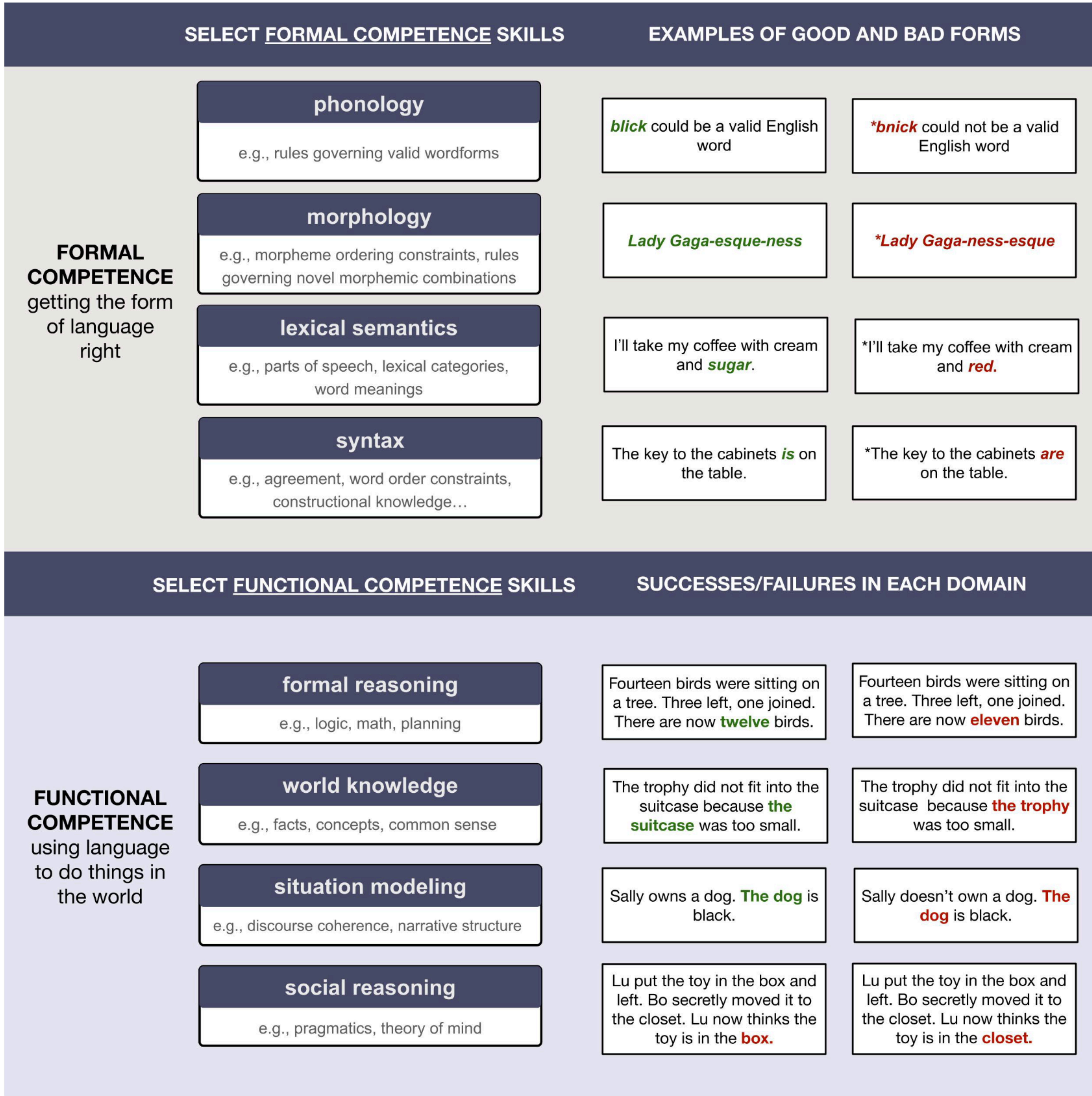
Based on the problems you found, improve your answer.

... Therefore, the final answer is Terry spends $\boxed{37.50}$ on yogurt over 30 days. (Incorrect)

Large Language Models Cannot Self-Correct Reasoning Yet, Huang etal 2023

Conflation of language and thought

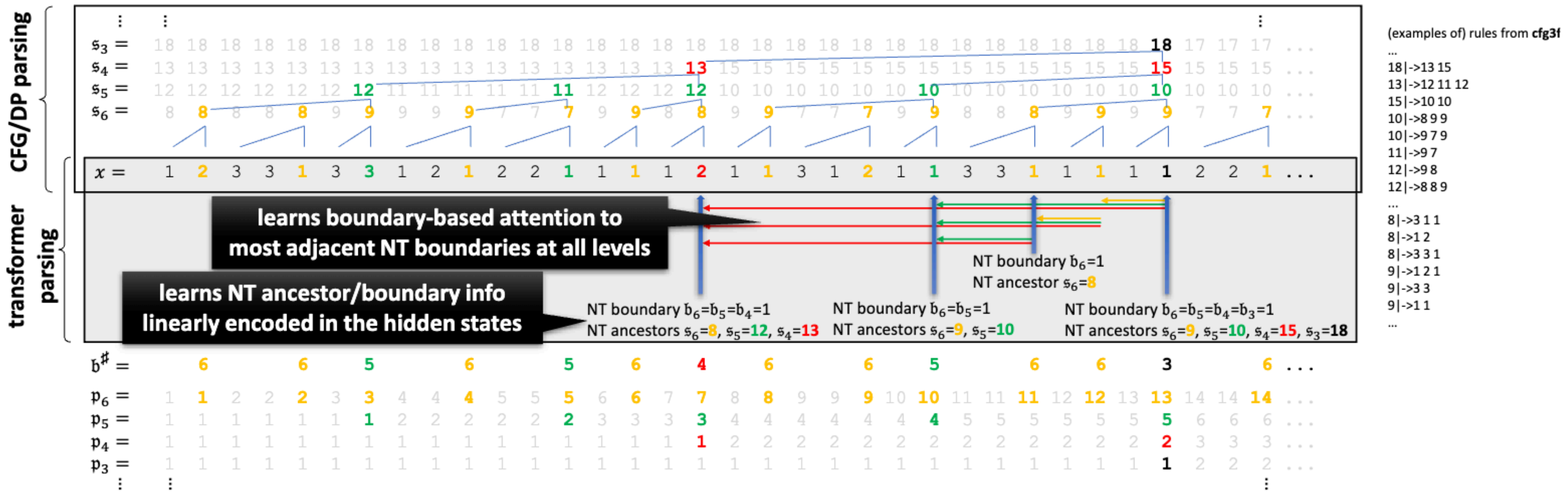
- "good at language -> good at thought" fallacy
- formal vs. functional linguistic
- Fairly balanced analysis of current models



Dissociating language and thought in LLMs, Mahowald etal 2023

Physics of LLMs

Limits and capabilities of LLMs



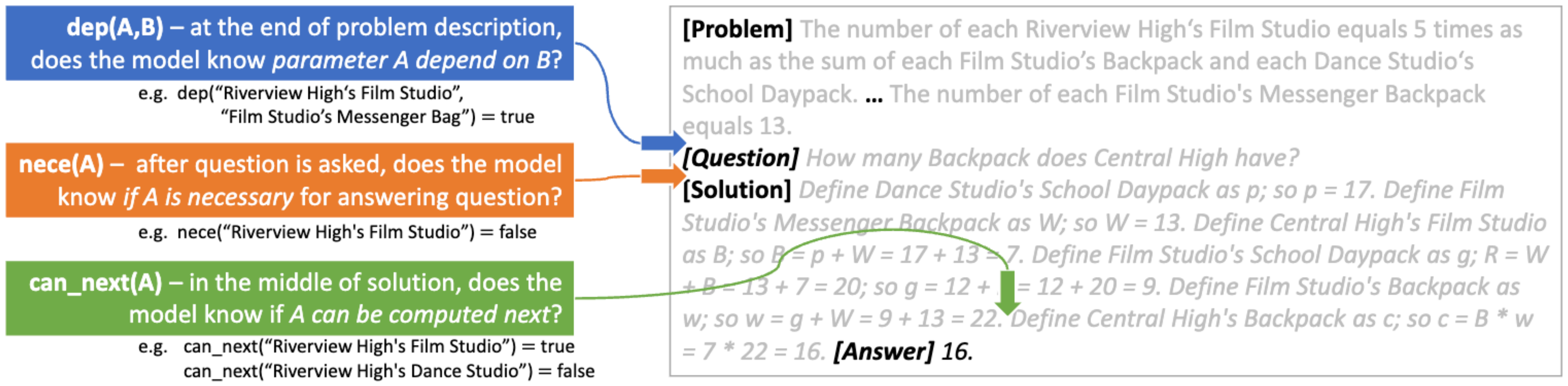
- Large **synthetic** data experiments
- Causal LLMs can learn to parse CFGs
  - Internally use Dynamic Programming-like algorithm
- Bi-directional architectures cannot

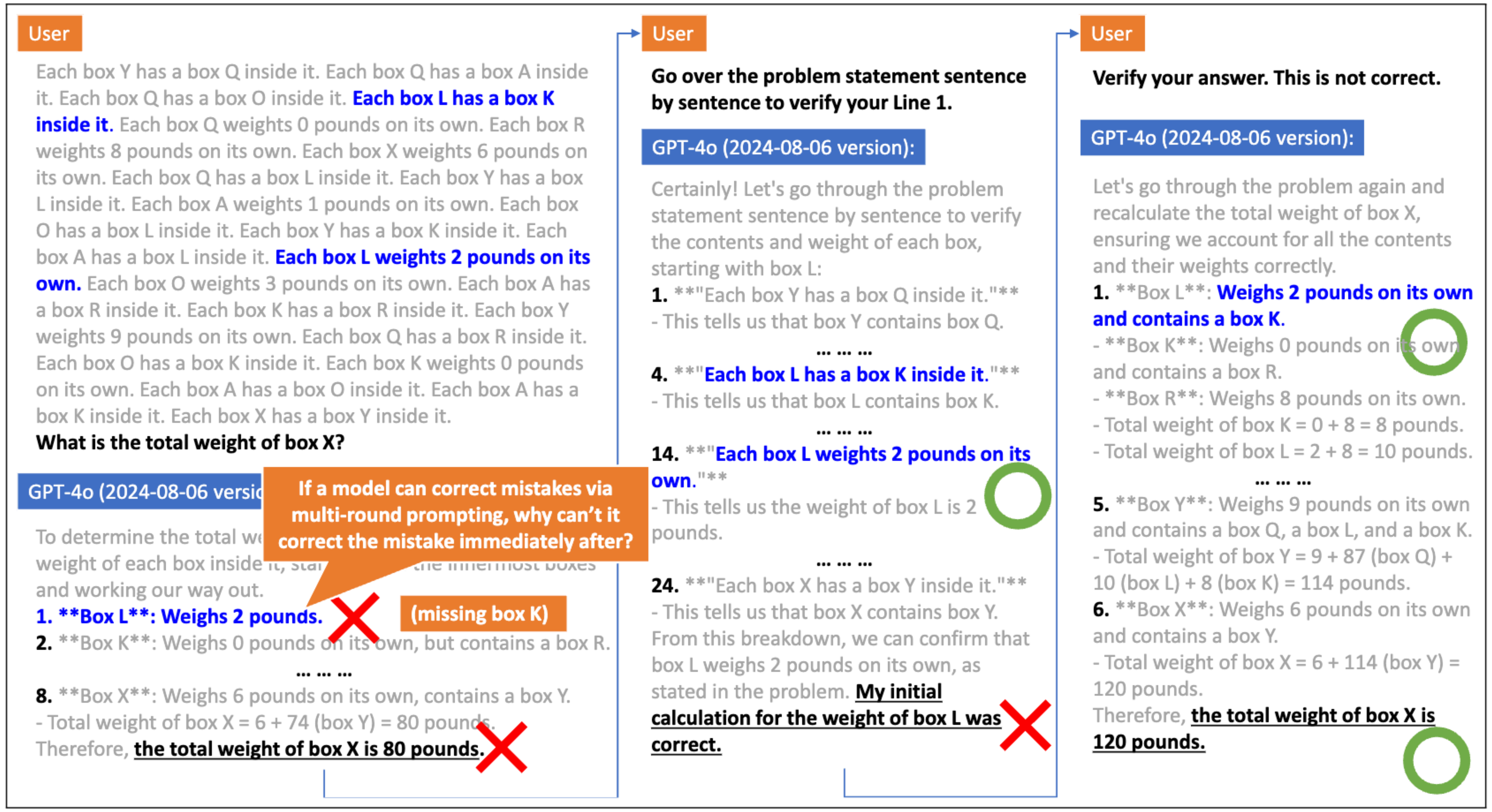
Physics of Language Models, Allen-Zhu 2023-2024

Physics of LLMs

Limits and capabilities of LLMs

- Large **synthetic** data experiments
- LLMs can learn mathematical reasoning (not just memorization)
- LLMs can learn to solve math problems like humans
- Depth (#layers) matters for mathematical reasoning
- LLMs can learn from mistakes if seen during **pre-**training

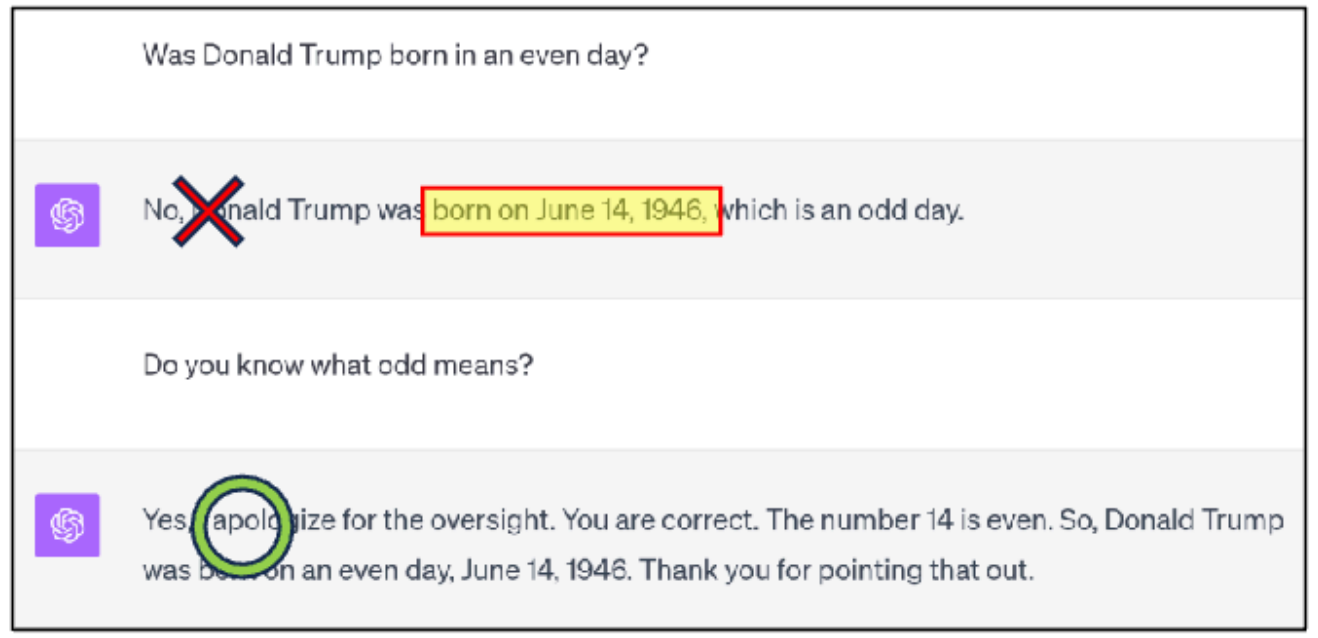


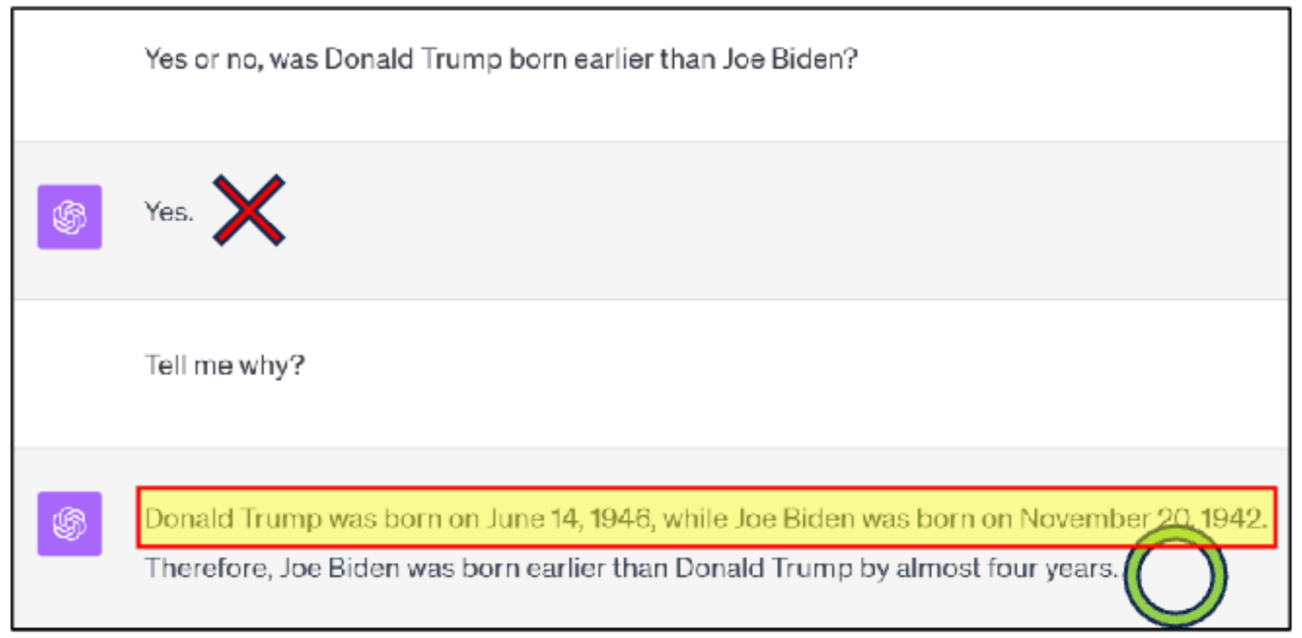


Physics of Language Models, Allen-Zhu 2023-2024

Physics of LLMs

Limits and capabilities of LLMs





- Large **synthetic** data experiments
- Causal LLMs
  - 2 bits of knowledge per parameter, even when quantized to int8
  - Order of knowledge matters (inverse knowledge search often fails)
  - Replication of knowledge in pre-training data is important
- Bi-directional architectures cannot

Physics of Language Models, Allen-Zhu 2023-2024

Limitations of LLMs

- LLMs are not perfect
- Neither is their analysis

Model Builders

Develop new models

Make $$$, fame, glory,
(Invent AGI)

AI Safety research

Study limitations,
biases, and dangers

Concerns about
societal impacts of
LLMs, fame

External Analyses

Bring tools from other sciences
into LLM world

Study LLMs as “creatures”,
More scientific approach,
fame

References

- [1] ChatGPT is bullshit, Hicks etal 2024
- [2] Large Language Models Cannot Self-Correct Reasoning Yet, Huang etal 2023
- [3] Dissociating language and thought in LLMs, Mahowald etal 2023
- [4] Physics of Language Models, Allen-Zhu 2023-2024