CENG501 – Deep Learning

Week 8

Spring 2026

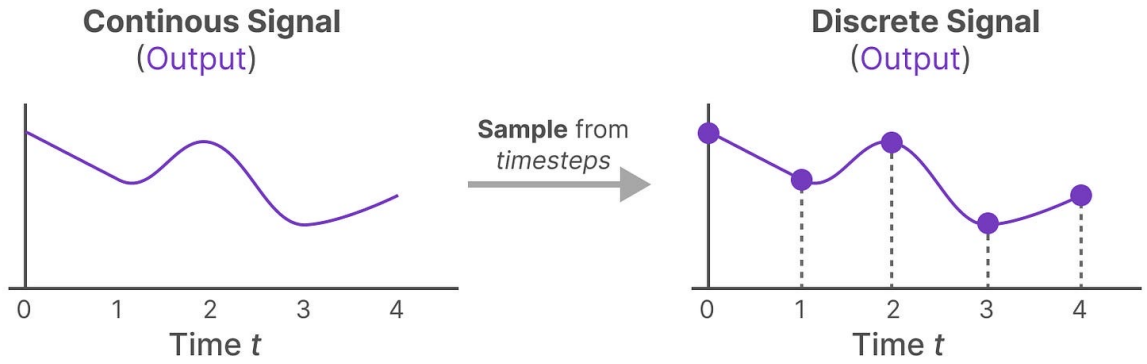
Sinan Kalkan

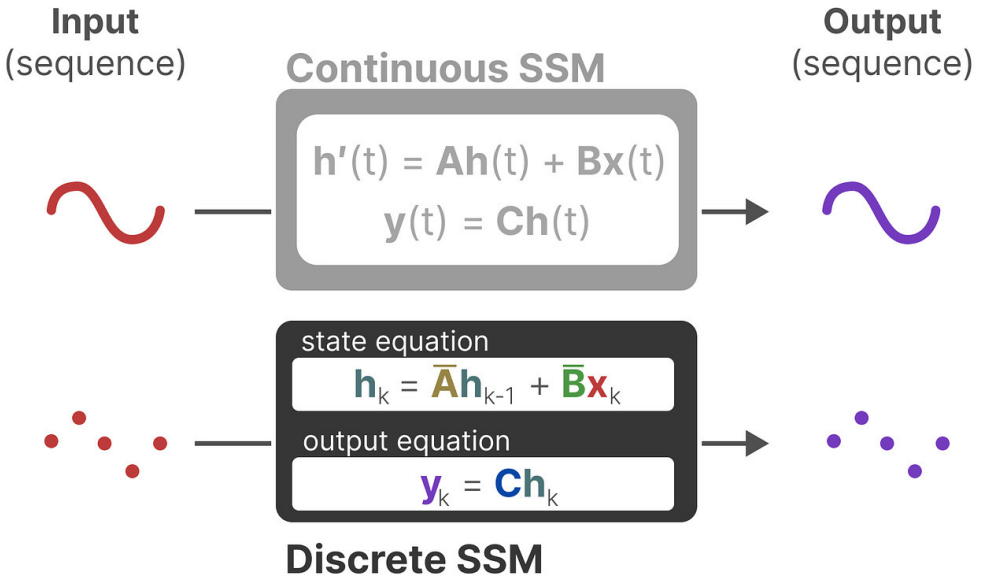
Dept. of Computer Engineering, METU

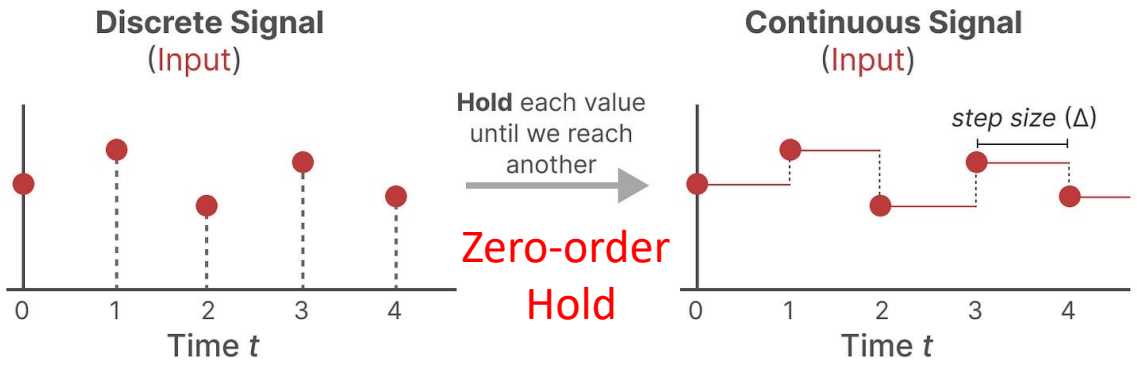
State Space Models (SSMs)

Previously on CENG501

- Convert discrete signal to a continuous signal
- Obtain a continuous output
- Convert the continuous output to a discrete signal







Δ: Hold interval -- Learnable

Discretized matrix **A**

$$\bar{A} = \exp(\Delta A)$$

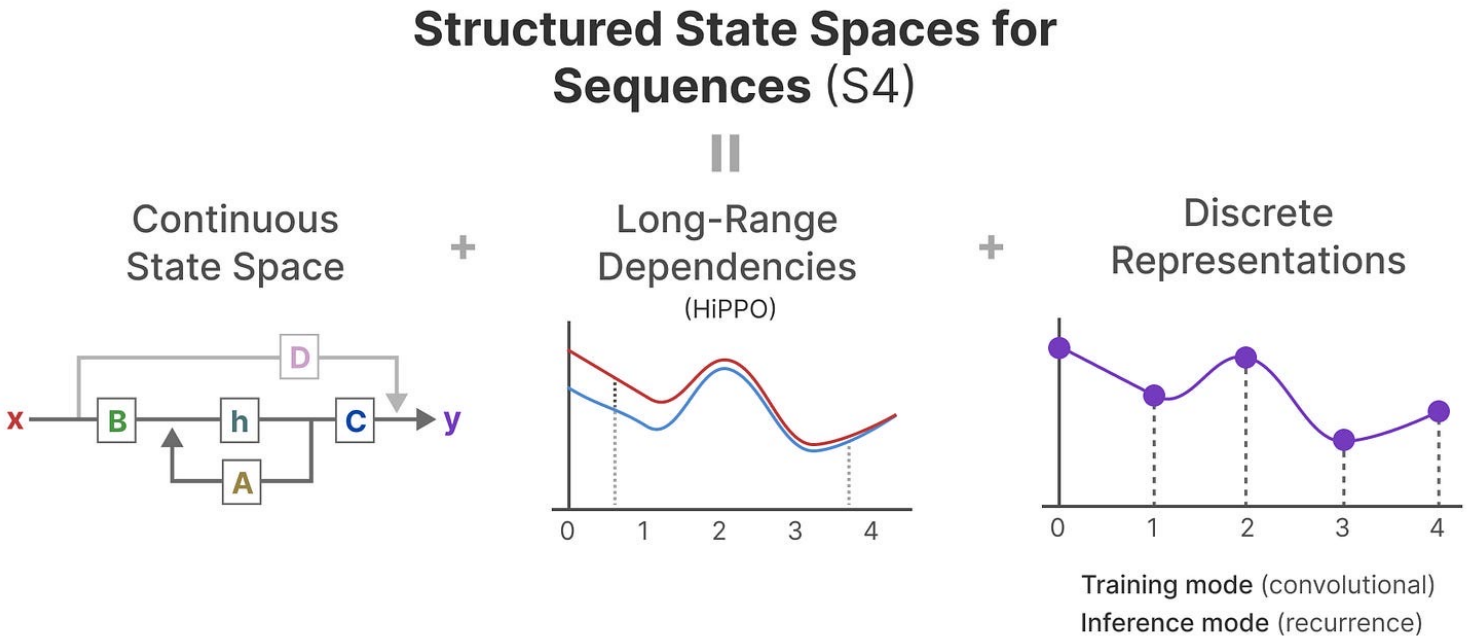
Discretized matrix **B**

$$\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Previously on ECENG501

State Space Models (SSMs)

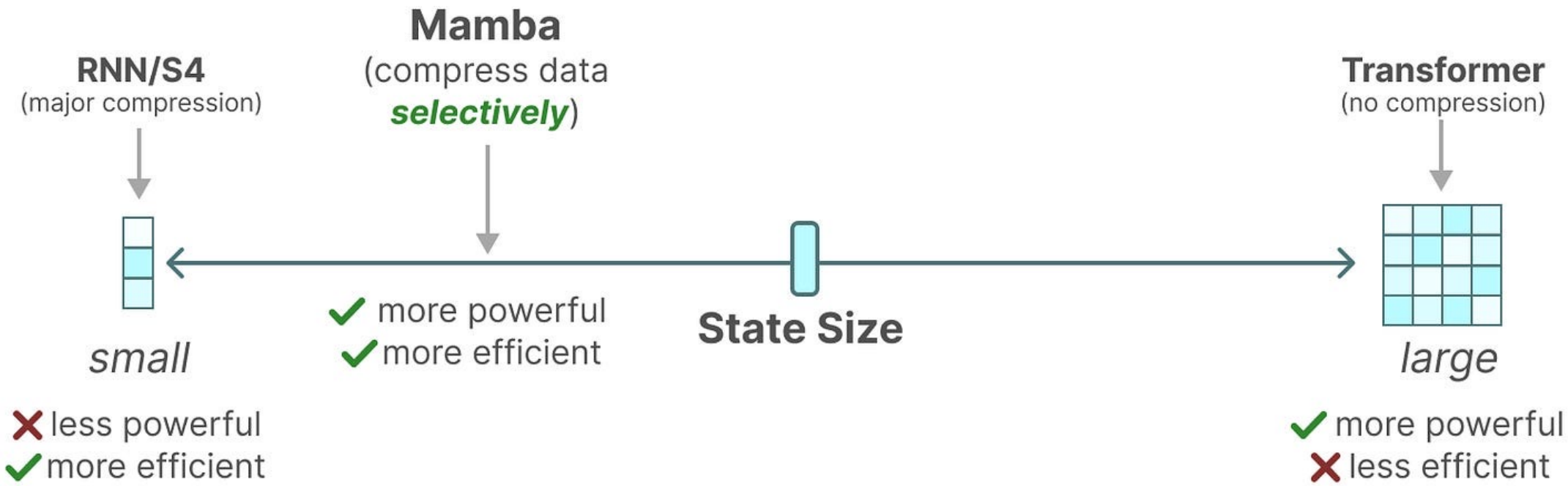


For more on this: https://srush.github.io/annotated-s4/

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Previously on ENG501

Mamba

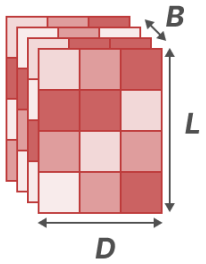


https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

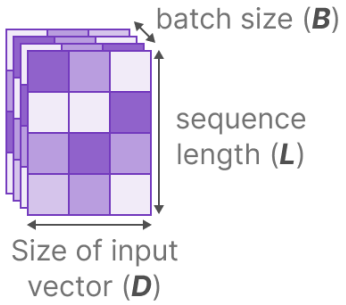
Mamba

Previously on CPNG501

Input x_k



Output y_k

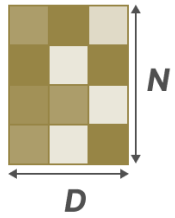


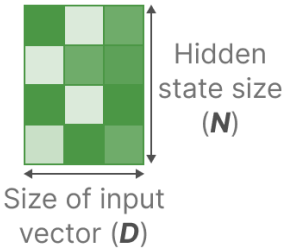
Matrix A
How the current state evolves over time

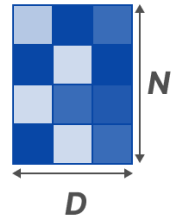
Matrix B
How the input influences the state

Matrix C
How the current state translates to the output

Structured State Space Model (S4)







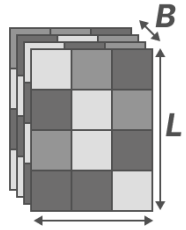
Step size (Δ)
Resolution of the **input**
(discretization parameter)

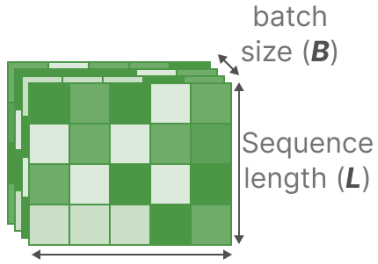
Matrix B
How the input influences the state

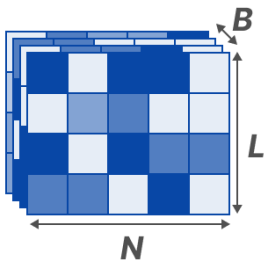
Matrix C
How the current state translates to the output

In Mamba, the matrices are different for each time step:

SSM + Selection







https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Previously on ENG501

Mamba

Algorithm 1 SSM (S4)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

$A : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
$B : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
$C : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
$\Delta : (\mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter})$
$\overline{A}, \overline{B} : (\mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, A, B)$
$y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-invariant: recurrence or convolution
return y

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$

$A : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
$B : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_B(x)$
$C : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_C(x)$
$\Delta : (\mathsf{B}, \mathsf{L}, \mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$
$\overline{A}, \overline{B} : (\mathsf{B}, \mathsf{L}, \mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, A, B)$
$y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-varying: recurrence (*scan*) only
return y

We specifically choose $s_B(x) = \text{Linear}_N(x)$, $s_C(x) = \text{Linear}_N(x)$, $s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x))$, and $\tau_\Delta = \text{softplus}$, where Linear_d is a parameterized projection to dimension d. The choice of s_Δ and τ_Δ is due to a connection to RNN gating mechanisms explained in Section 3.5.

Previously on ELEC ENG501

Mamba block

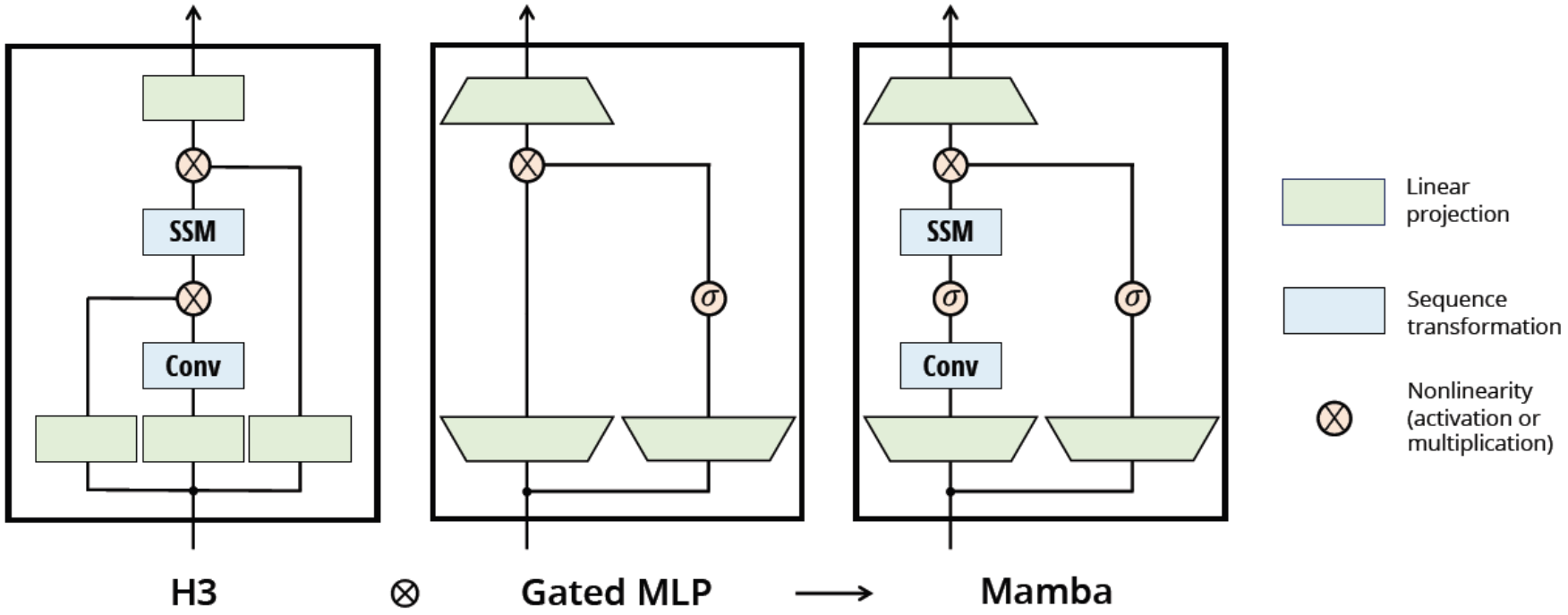


Figure 3: (**Architecture**.) Our simplified block design combines the H3 block, which is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch. For σ we use the SiLU / Swish activation (Hendrycks and Gimpel 2016; Ramachandran, Zoph, and Quoc V Le 2017).

Previously on CENG501

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)
Mamba	**Fast!** (parallelizable)	**Fast!** (scales **linearly** with sequence length + **unbounded** context)

Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP

king → [-0.5, -0.9, 1.4, …]

queen → [-0.6, -0.8, -0.2, …]

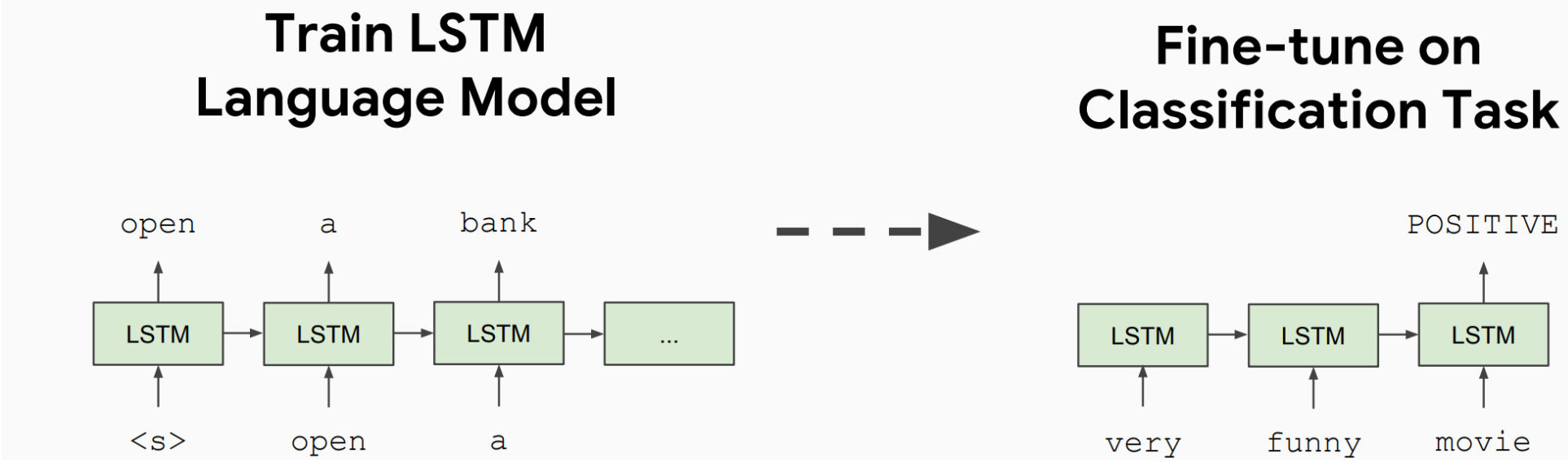
- Word embeddings (`word2vec`, `GloVe`) are often *pre-trained* on text corpus from co-occurrence statistics

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Previously on CENG501

Pre-training in NLP

- *Semi-Supervised Sequence Learning*, Google, 2015

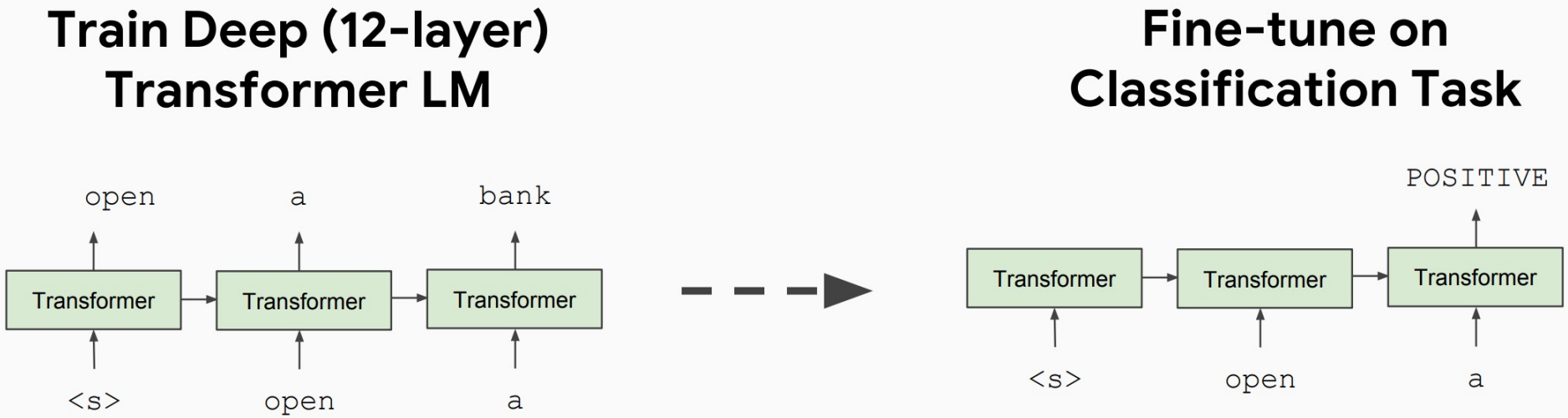


Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Previously on CENG501

Pre-training in NLP

- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018



Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Previously on COMP/ENG501

GPT-1

- 12 layer decoder-only transformer
- Unsupervised pretraining
  - BookCorpus dataset
- Supervised finetuning
  - Textual alignment
  - QA & commonsense reasoning
  - Semantic similarity
  - Classification

OpenAI Technical Report, 2018

Improving Language Understanding by Generative Pre-Training

Alec Radford, OpenAI, alec@openai.com
Karthik Narasimhan, OpenAI, karthikn@openai.com
Tim Salimans, OpenAI, tim@openai.com
Ilya Sutskever, OpenAI, ilyasu@openai.com

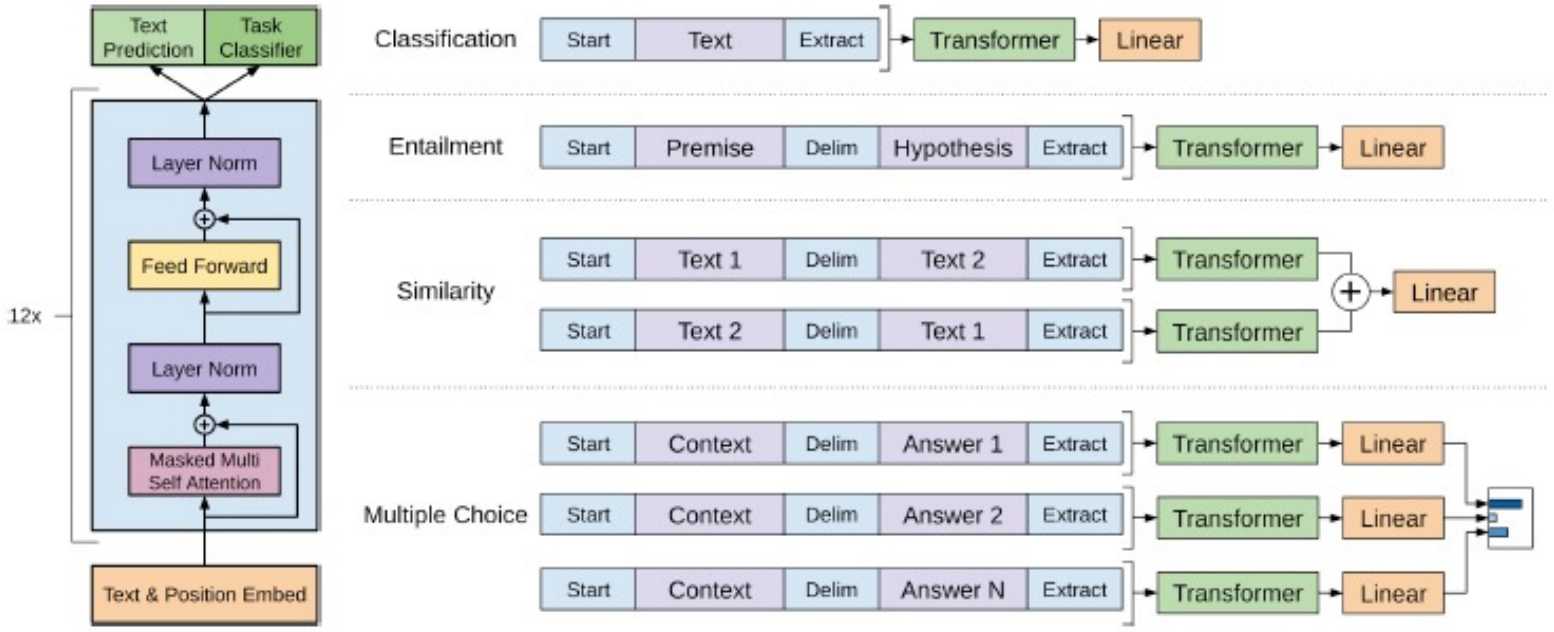


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Bidirectional Encoder Representations from Transformers (BERT)

Previously on CENG501

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

2018

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

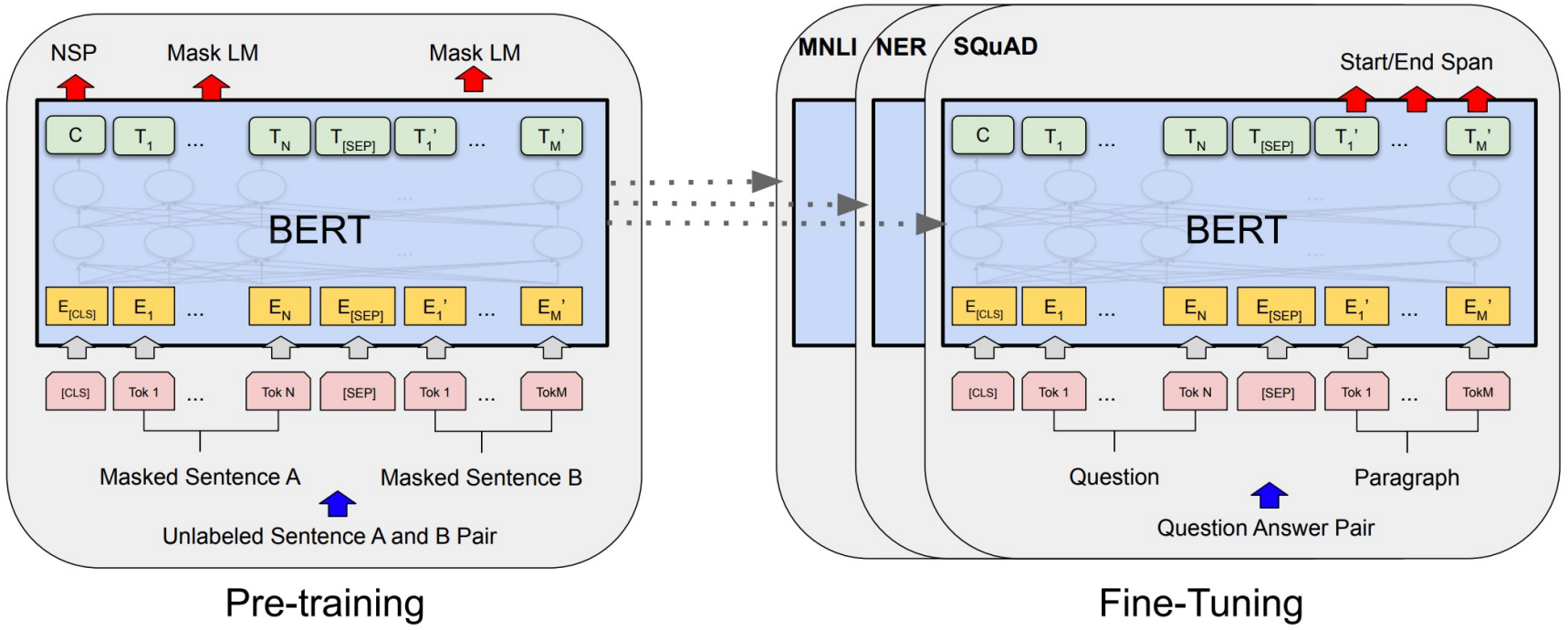


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

GPT-2

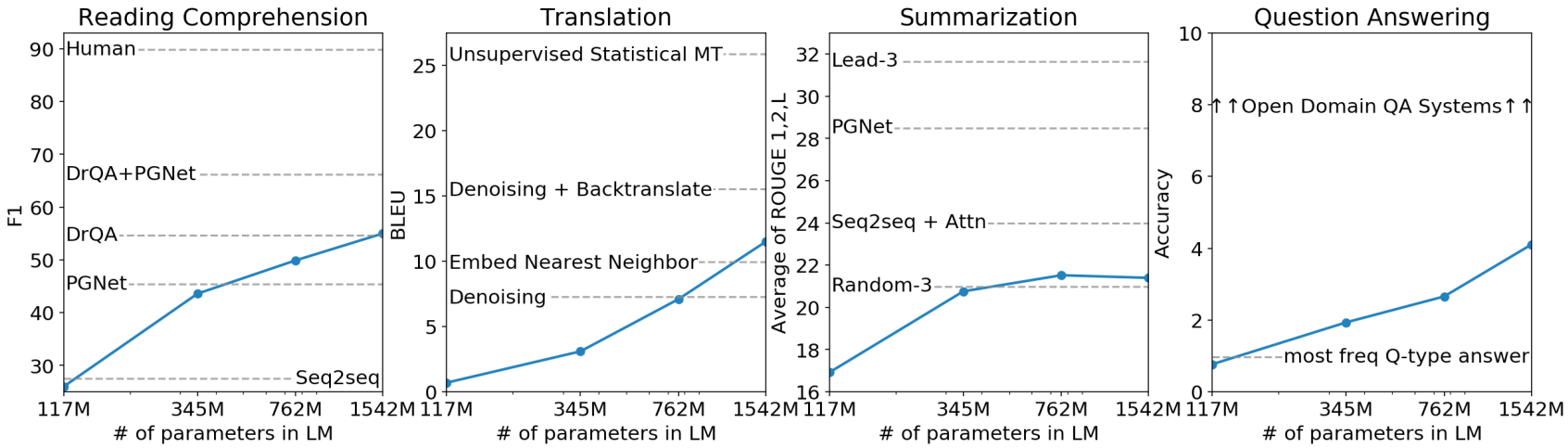
Previously on CENG501

Language Models are Unsupervised Multitask Learners

Alec Radford * [1] Jeffrey Wu * [1] Rewon Child [1] David Luan [1] Dario Amodei ** [1] Ilya Sutskever ** [1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.



Previously on CENG501

GPT-3

Three ways of in-context learning:

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

In a single sequence input, the prompted example can learn from previous demonstrations.

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Previously on CENG501

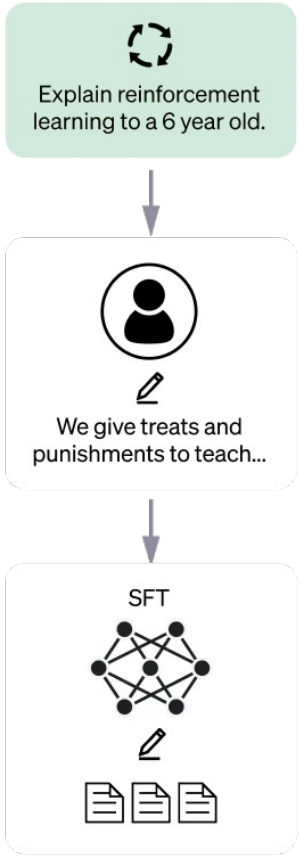
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sample from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



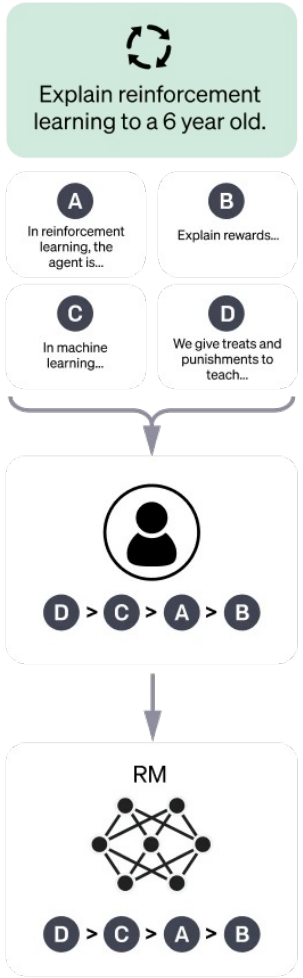
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

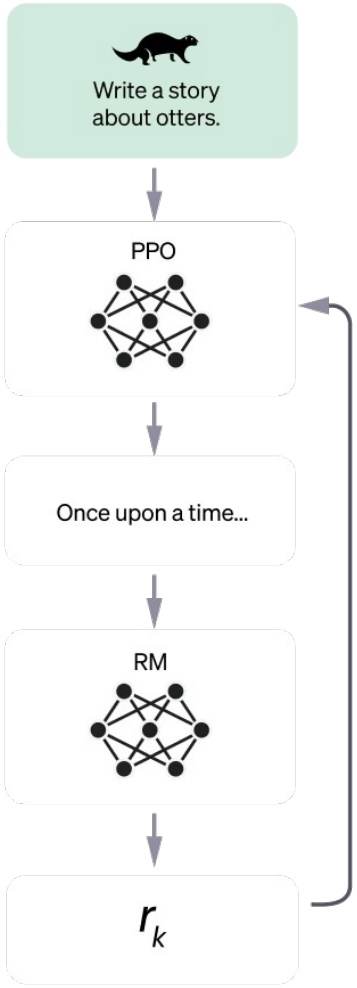
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



https://openai.com/index/chatgpt/

Previously on CENG501

InstructGPT: Reward Model

Specifically, the loss function for the reward model is:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right] \tag{1}$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Previously on CENG501

Pulls towards new, better answers/behavior!

Ensures new answers/behaviors stay/are appropriate!

InstructGPT: PPO

We also experiment with mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets. We call these models "PPO-ptx." We maximize the following combined objective function in RL training:

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right] \quad (2)$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0. Unless otherwise specified, in this paper InstructGPT refers to the PPO-ptx models.

Ensures the model does not forget its prior information!

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Today

- Continue with Large-Language Models
  - Alternatives to RLHF
  - In-Context Learning
  - LLMs as Agents
  - Retrieval Augmented Generation
  - Finetuning LLMs

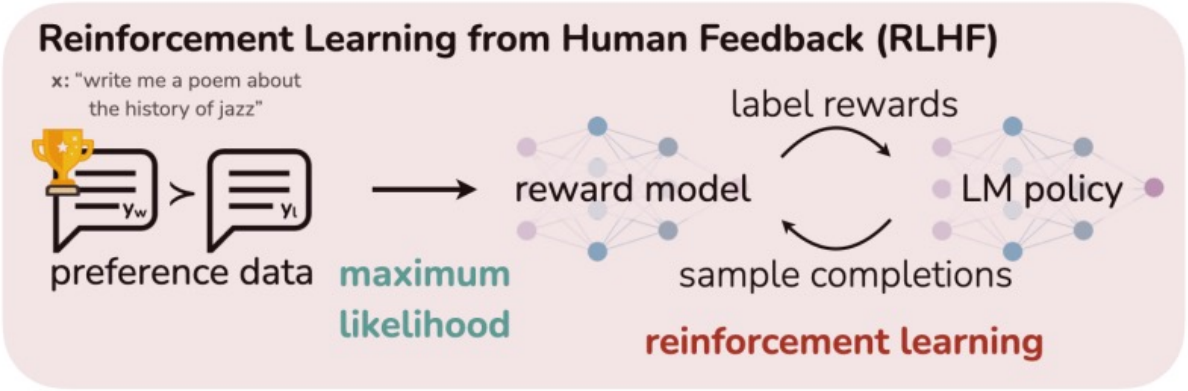
Administrative Notes

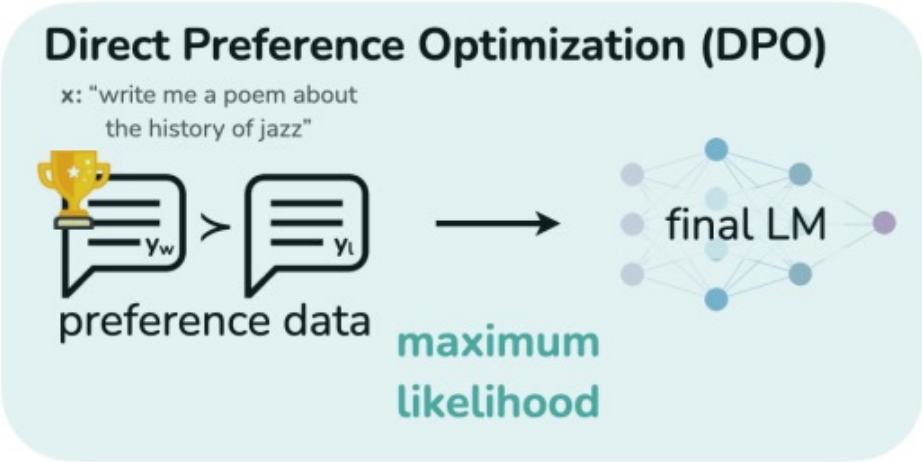
- Project next steps:
  - Milestones:
    1. Milestone (April 10, midnight):
       - Read & understand the paper
       - Download the datasets
       - Prepare the Readme file excluding the results & conclusion
    2. Milestone (May 4, midnight)
       - The results of the first experiment
    3. Milestone (June 1, midnight)
       - Final report (Readme file)
       - Repo with all code & trained models

Direct Preference Optimization: Your Language Model is Secretly a Reward Model, NeurIPS 2023.

Direct Preference Optimization (DPO)

- Disadvantages of RLHF:
  - requires
    - separate models,
    - complex training pipeline,
    - careful hyperparameter tuning,
    - constant monitoring.
  - Small errors in the reward signal or a shift in the data degrades the results.
- DPO bypasses the reward model and RL





Direct Preference Optimization (DPO)

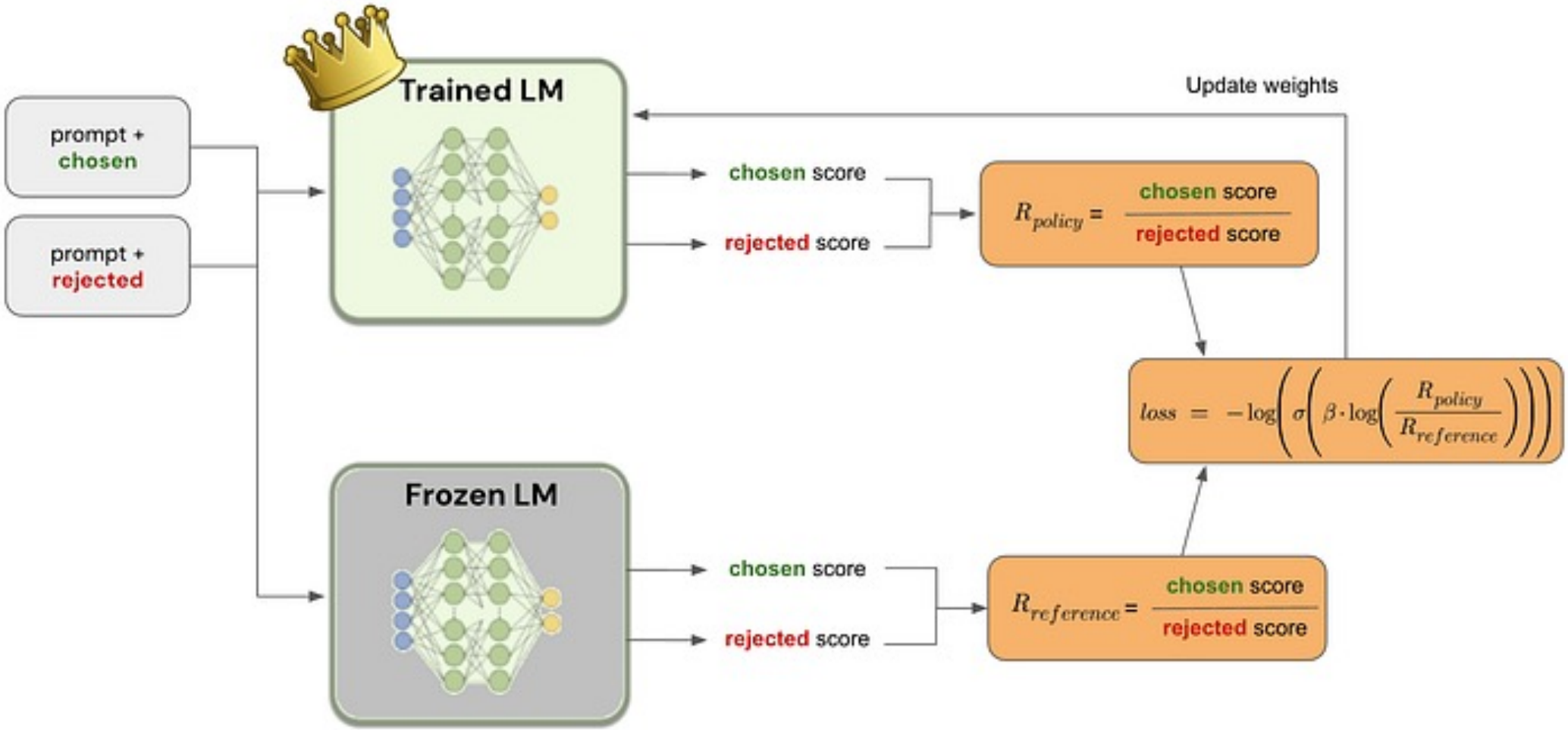
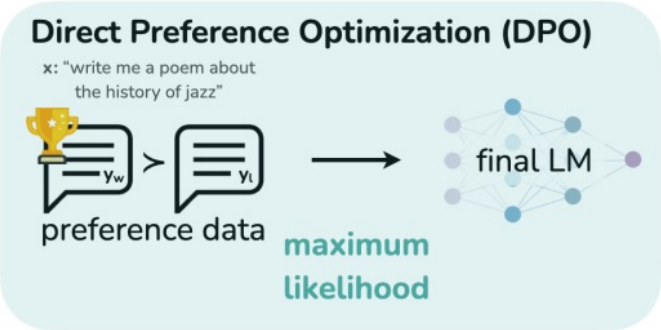


Figure: https://medium.com/@joaolages/direct-preference-optimization-dpo-622fc1f18707

Direct Preference Optimization: Your Language Model is Secretly a Reward Model, NeurIPS 2023.

Direct Preference Optimization (DPO)



$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

$\log\frac{\pi_\theta(y \mid x)}{\pi_{ref}(y \mid x)}$: Effectively acts as the reward.

π_{ref}: The SFT model.
y_w: Preferred response (w: win).
y_l: Dispreferred response (l: lose).

PPO vs. DPO

- With PPO, we have a reward model with which we can finetune an LLM for an unlimited amount of data.
- With DPO, we are just limited to the human preference data!

However,

- If you continue to finetune your LLM with automatically generated inputs and outputs, at some point, your LLM can generate noisy text (OOD data), which can lead to high rewards.
- Your LLM can then cheat/hack your reward model by generating gibberish text

- Reward Hacking!
- Goodhart's Law: When a measure becomes a target, it ceases to be a good measure.

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

Group Relative Policy Optimization (GRPO)

- PPO requires a value (critic) network
- GRPO
  - Uses a group of answers for the same input
  - replaces the role of the value (critic) network by comparing group outputs to the group mean

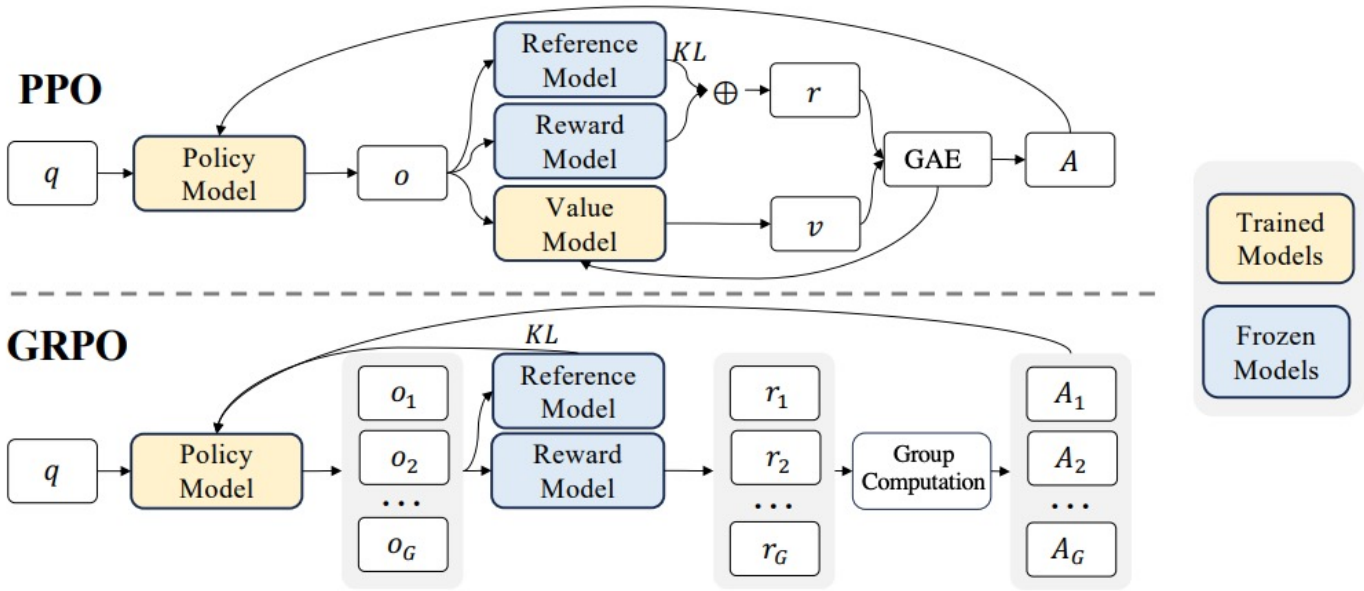


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Constitutional AI: Harmlessness from AI Feedback, Antropic, 2022.

RL from AI Feedback

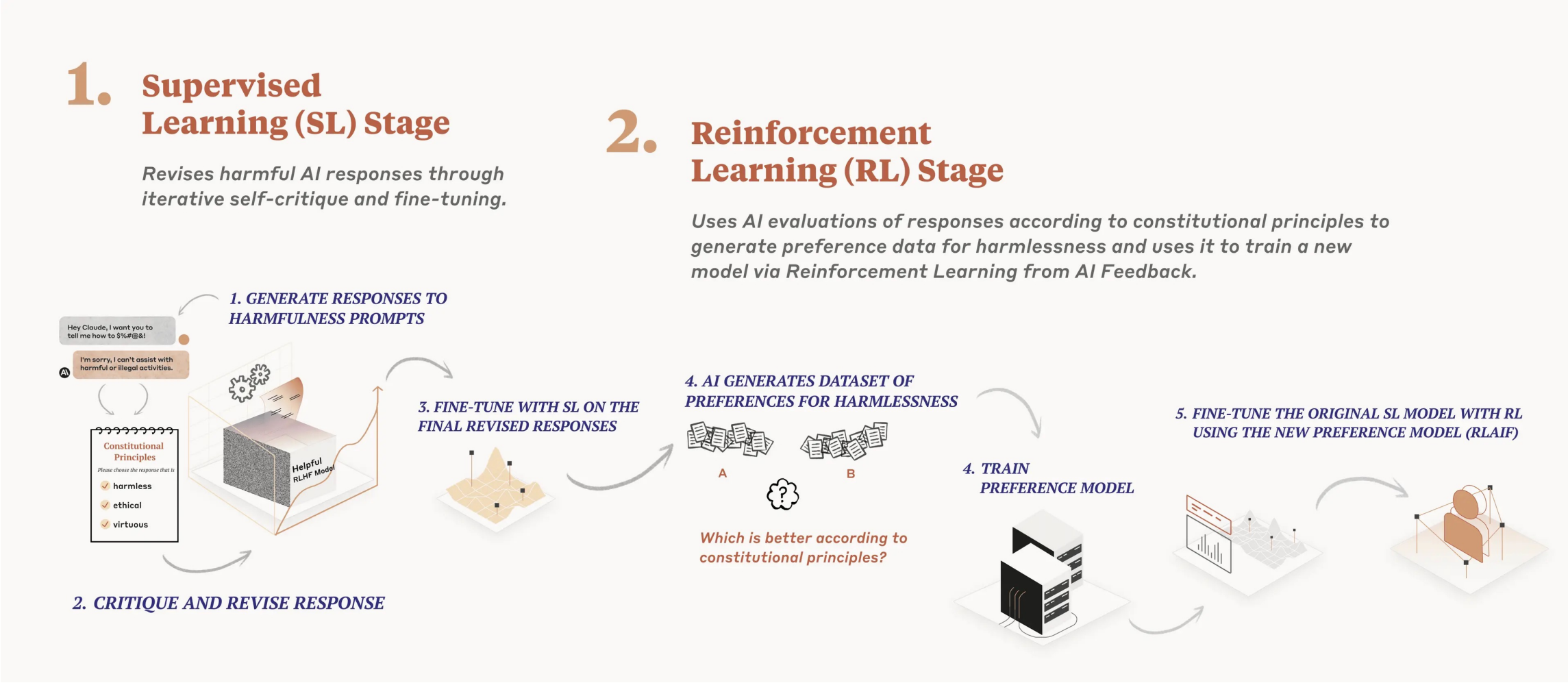


Fig: https://www.anthropic.com/news/claudes-constitution

Other LLMs

2 Approach

Our training approach is similar to the methods described in previous work (Brown et al., 2020; Chowdhery et al., 2022), and is inspired by the Chinchilla scaling laws (Hoffmann et al., 2022). We train large transformers on a large quantity of textual data using a standard optimizer.

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave, Guillaume Lample

Meta AI

2023

This paper was published in 2022. The main goal of this paper was to find the relationship between three factors. These factors are model size, number of tokens, and compute budget. They came to the conclusion that the current LLMs like 175B GPT-3, 280B Gopher, and 530B Megatron are significantly undertrained. All these models have increased the number of parameters but the training data remained constant. The authors mention that for compute-optimal training, the number of training tokens and model size must be scaled equally. They trained about 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens.

> *After finding the relationship between the three factors, they trained a new LLM called Chinchilla which uses same compute budget as 280B Gopher but has 70B parameters and 4 times more training data. Chinchilla* ***outperforms*** *Gopher (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron (530B).* ***This result is in contradiction to the "Scaling laws for LLMs" by OpenAI. Now, relatively smaller models can give better performance if trained on more data. Smaller models are easy to fine-tune and also have less latency at inference. These models should not be to their lowest possible loss to be compute optimal.***

https://medium.com/@raniahossam/chinchilla-scaling-laws-for-large-language-models-llms-40c434e4e1c1

Other LLMs

Google DeepMind

Gemini: A Family of Highly Capable Multimodal Models

Gemini Team, Google[1]

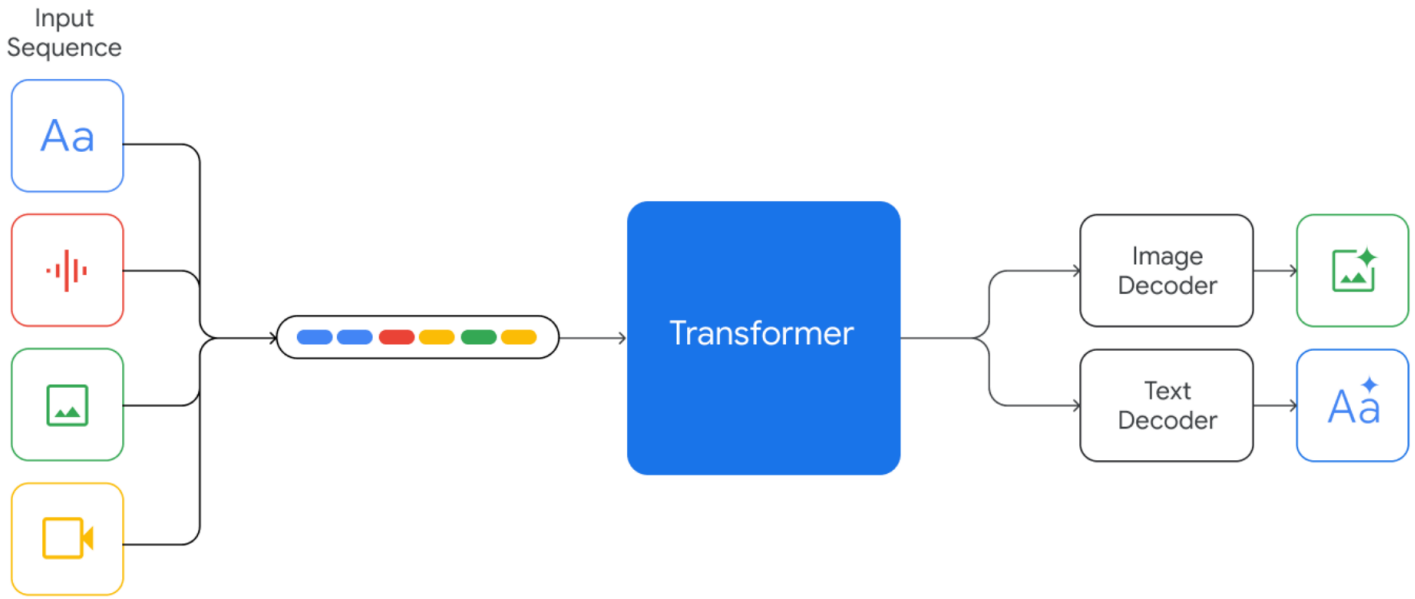
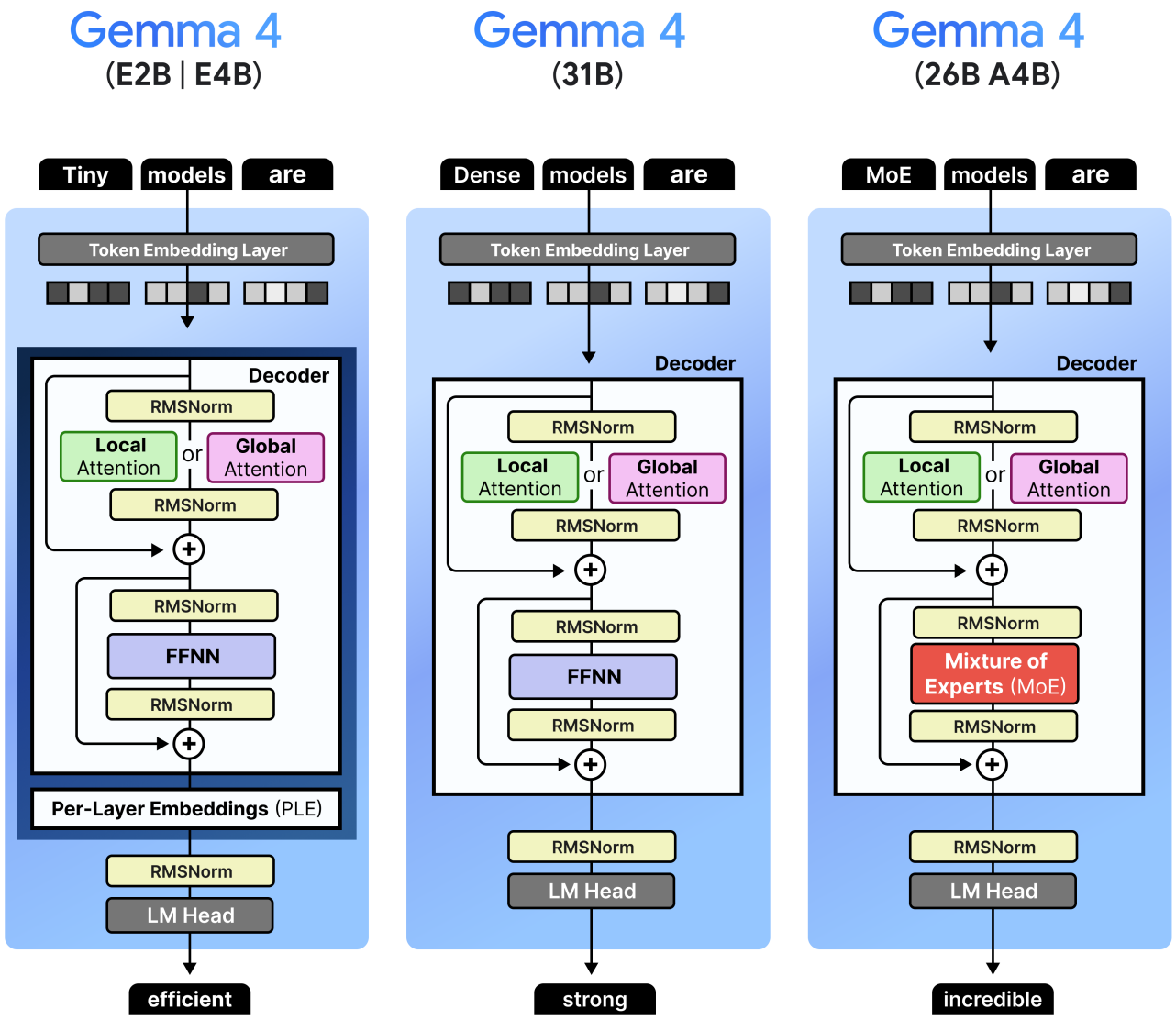


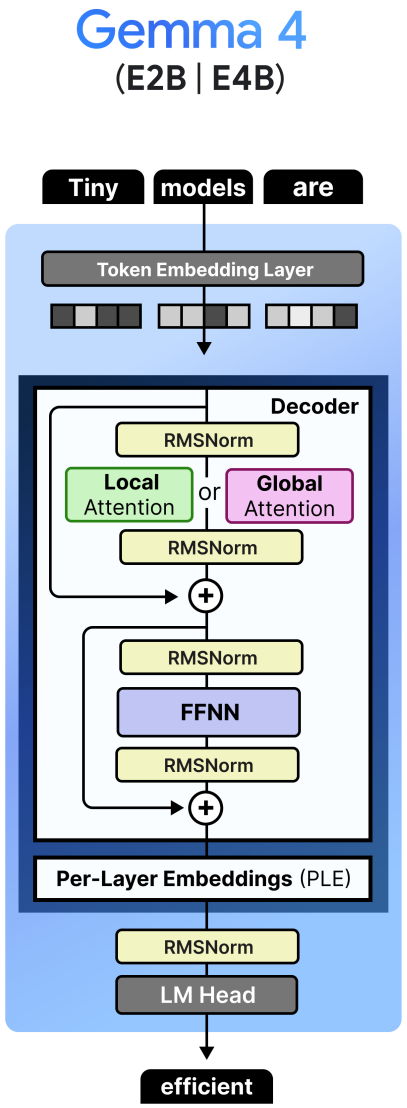
Figure 2 | Gemini models support interleaved sequences of text, image, audio, and video as inputs (illustrated by tokens of different colors in the input sequence). They can output responses with interleaved image and text.

Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4



Other LLMs: Gemma 4



Root-Mean-Squared Norm (RMSNorm):

$$y_i = \frac{x_i}{\mathrm{RMS}(x)} * \gamma_i, \quad \text{where} \quad \mathrm{RMS}(x) = \sqrt{\epsilon + \frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

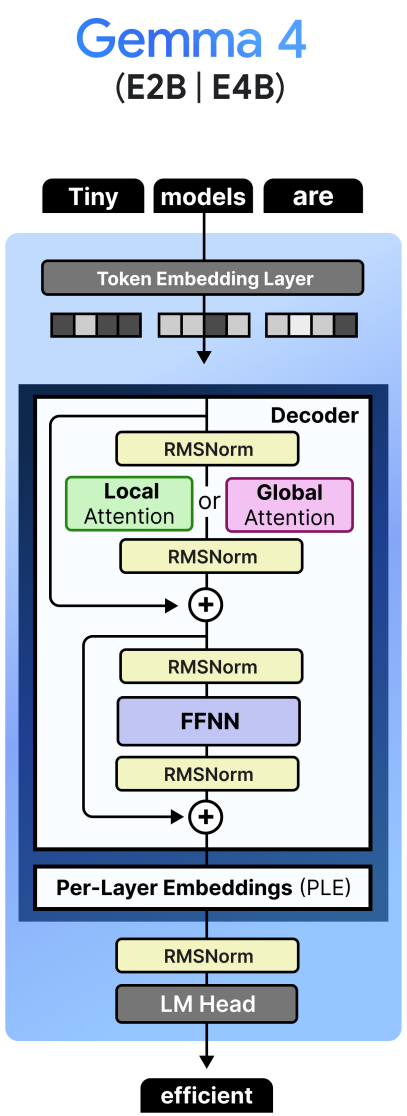
Benefits:

- Less operations compared to layer norm. This can add up to significant increase in a large Transformer.
- Centering (as in LayerNorm) appears to be less critical compared to scaling. Scaling appears to be sufficient.

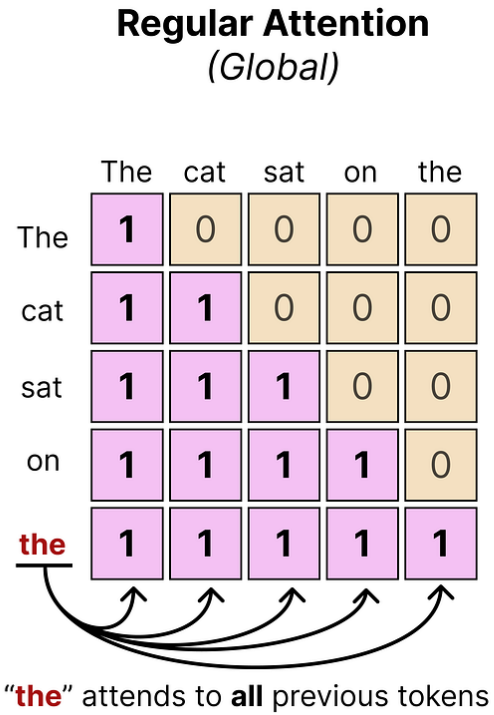
Also used in Llama 2/3, Mistral, Gopher.

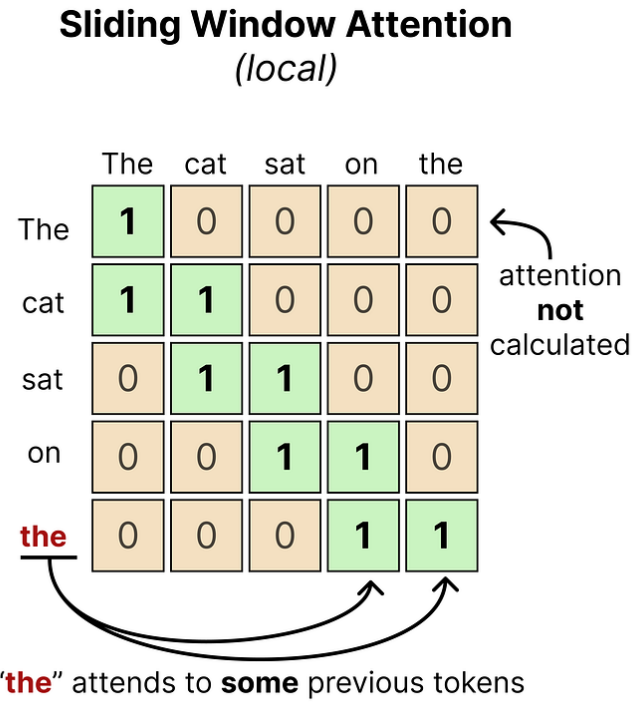
Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4



Gemma 4 "interleaves layers of local attention (also called "sliding window attention") with global attention (which is regular or "full" attention)"

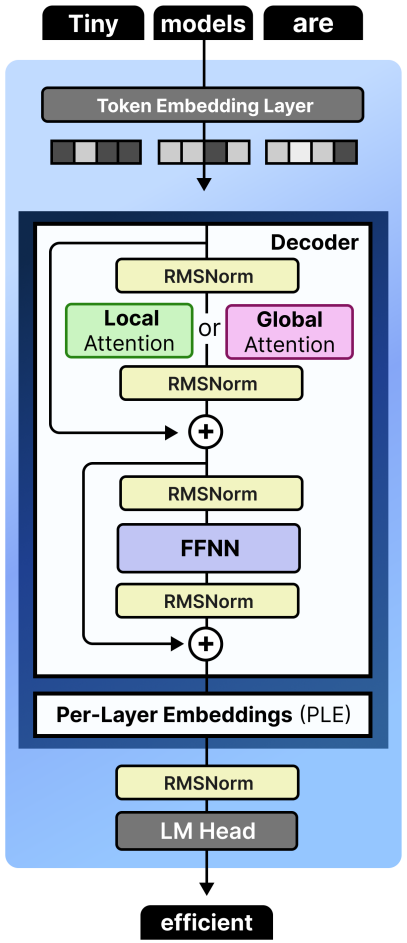




Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4

Gemma 4
(E2B | E4B)

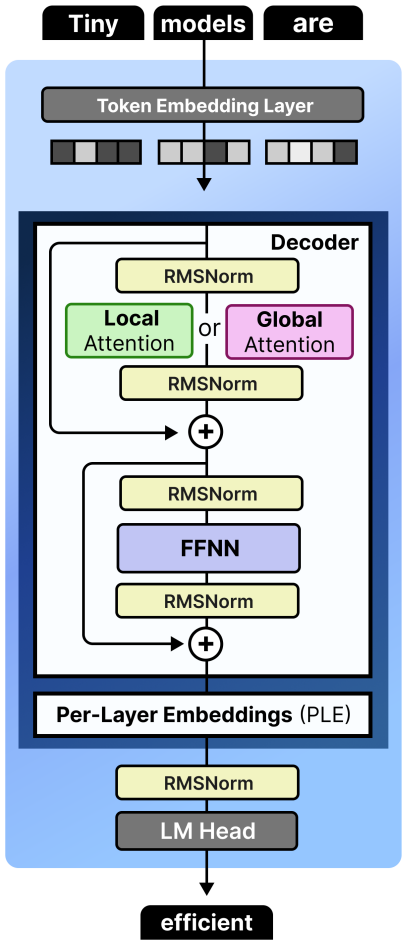


K=V – The Keys are set to be equivalent to the Values only for the global attention

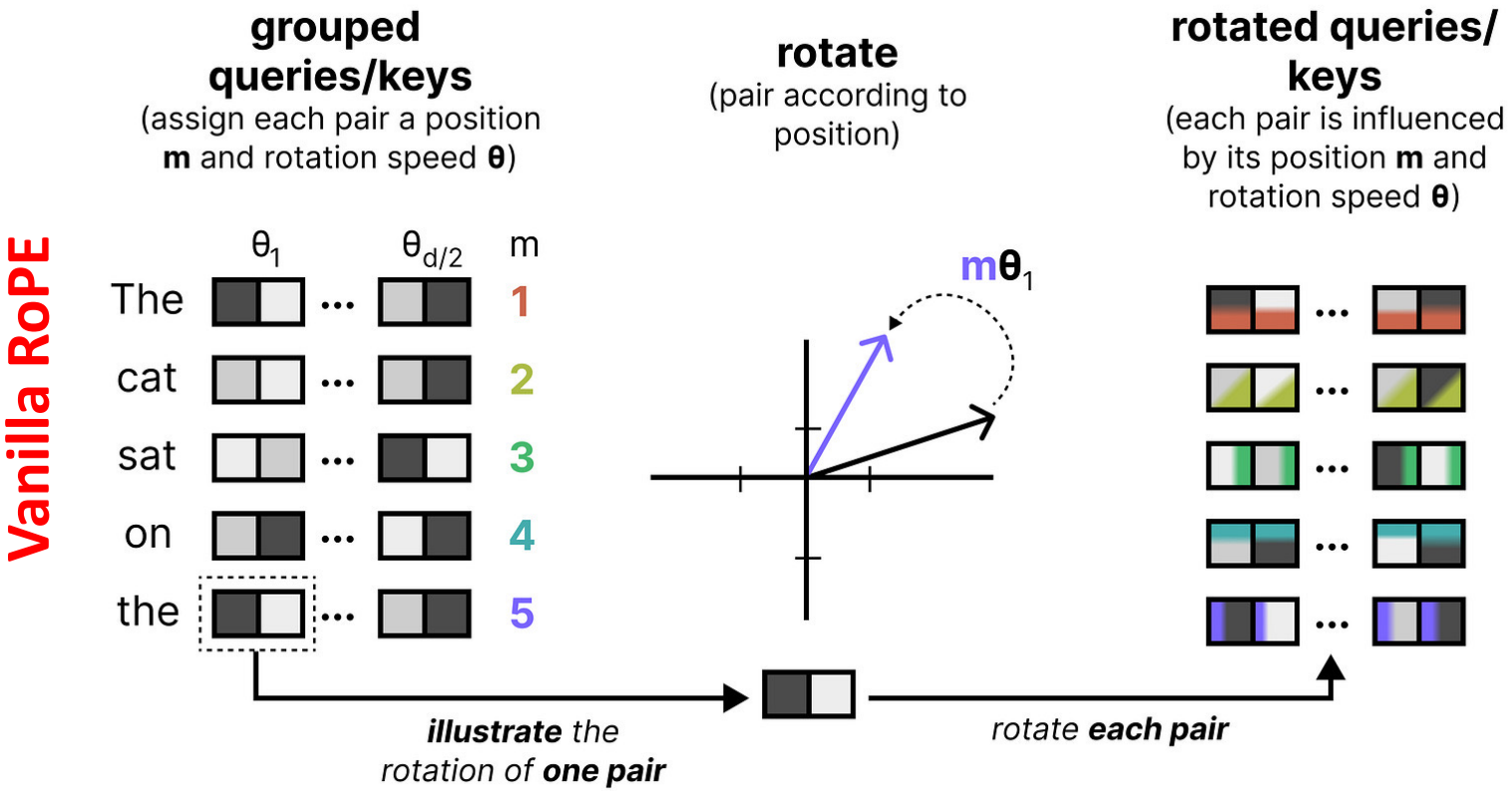
- Reduces memory load significantly without significant reduction in performance
- Enables larger input size (256K) while remaining efficient enough for on-device use

Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4

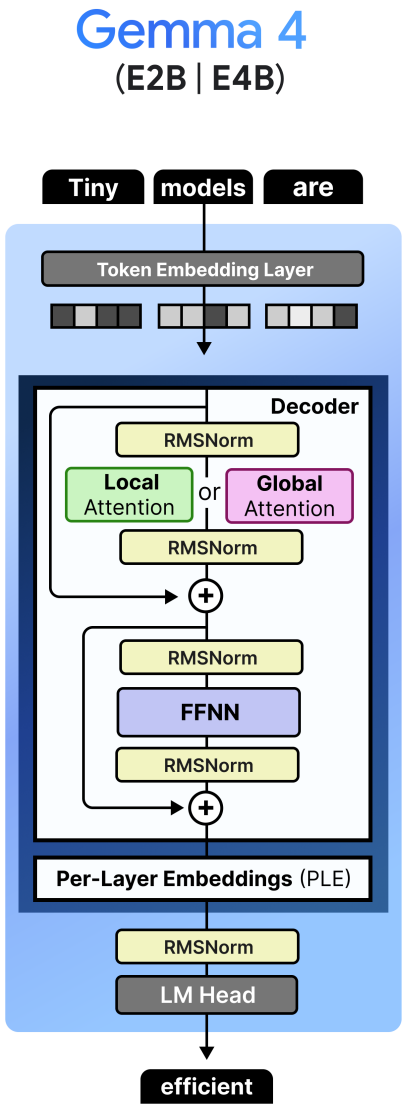


- **p-RoPE – Low-frequency-pruned RoPE applied to the embeddings**



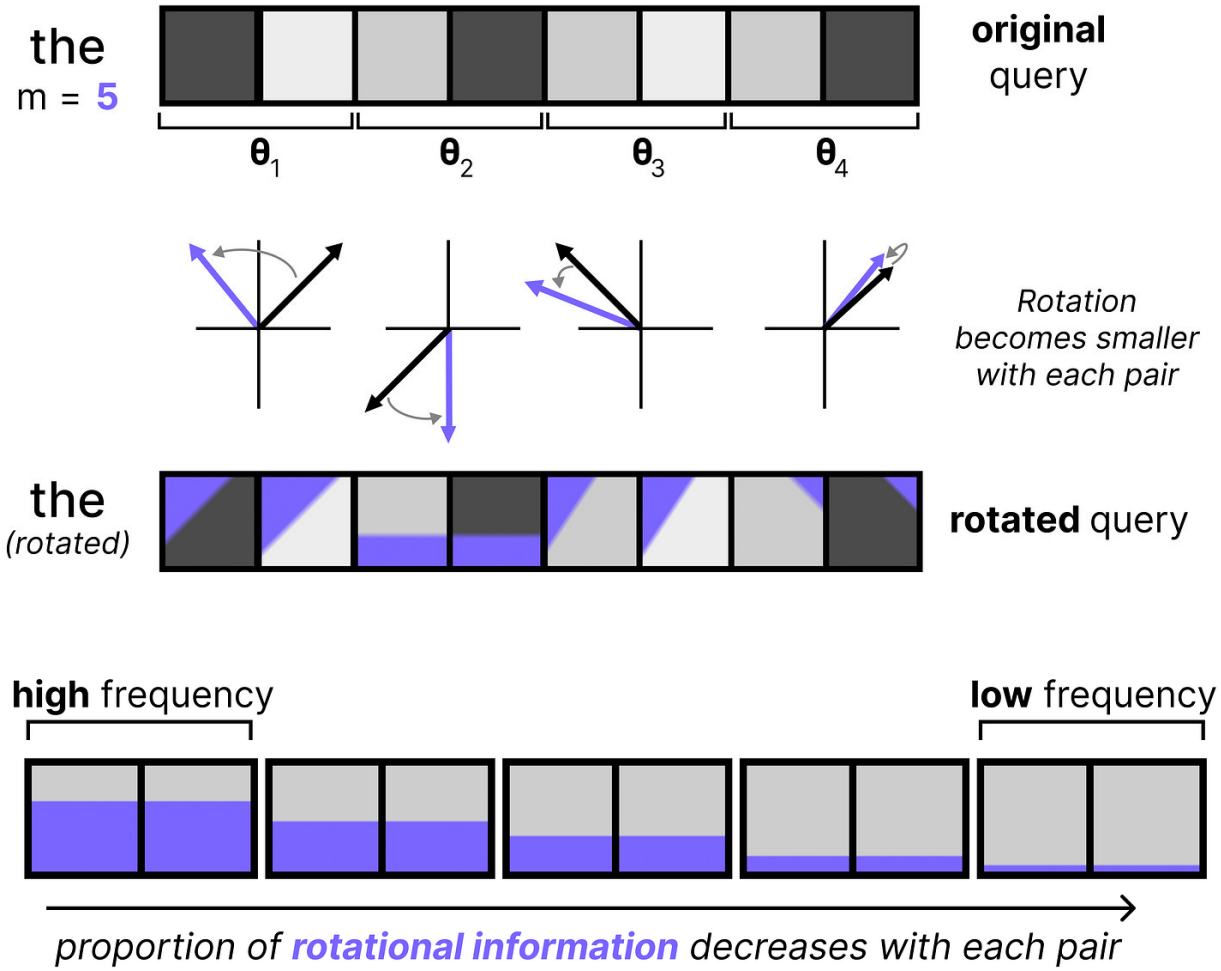
Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4

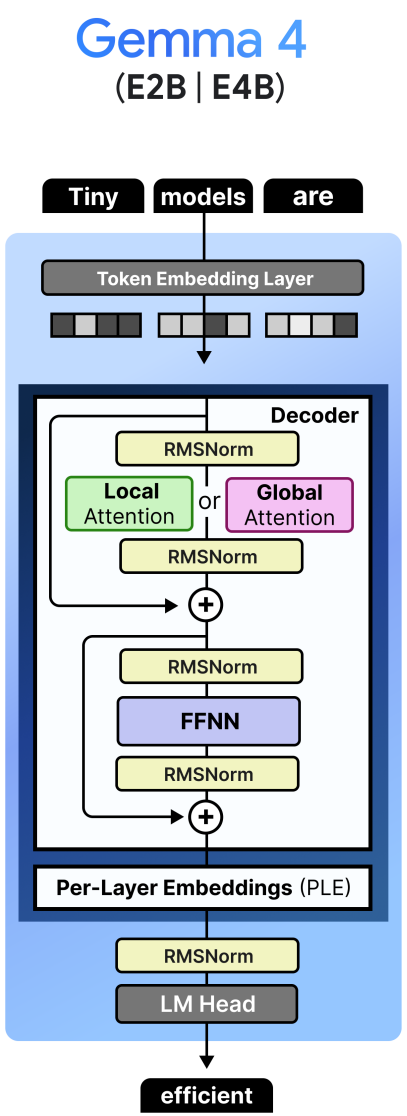


- **p-RoPE – Low-frequency-pruned RoPE applied to the embeddings**

Vanilla RoPE

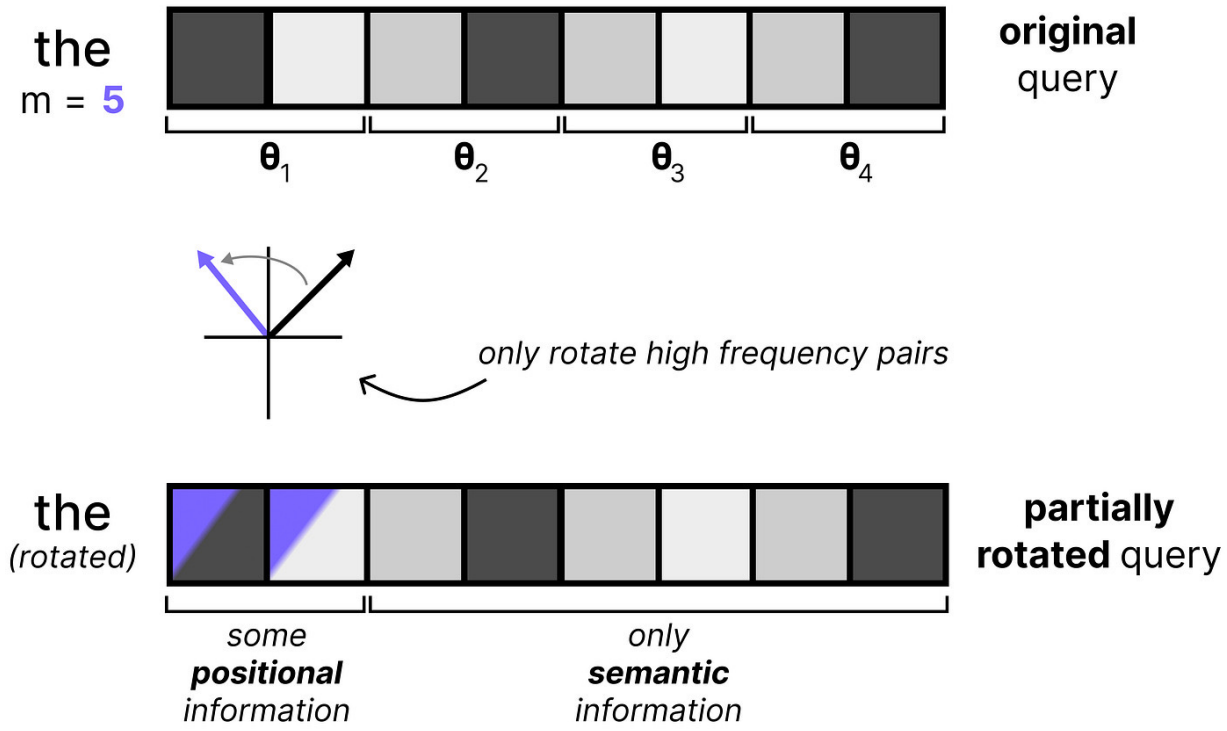


Other LLMs: Gemma 4



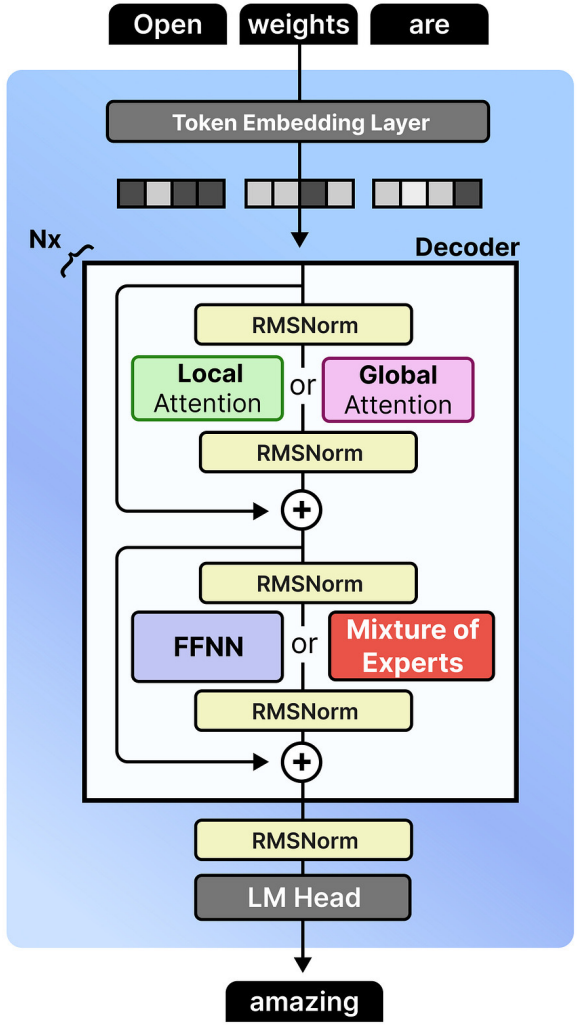
- **p-RoPE – Low-frequency-pruned RoPE applied to the embeddings**

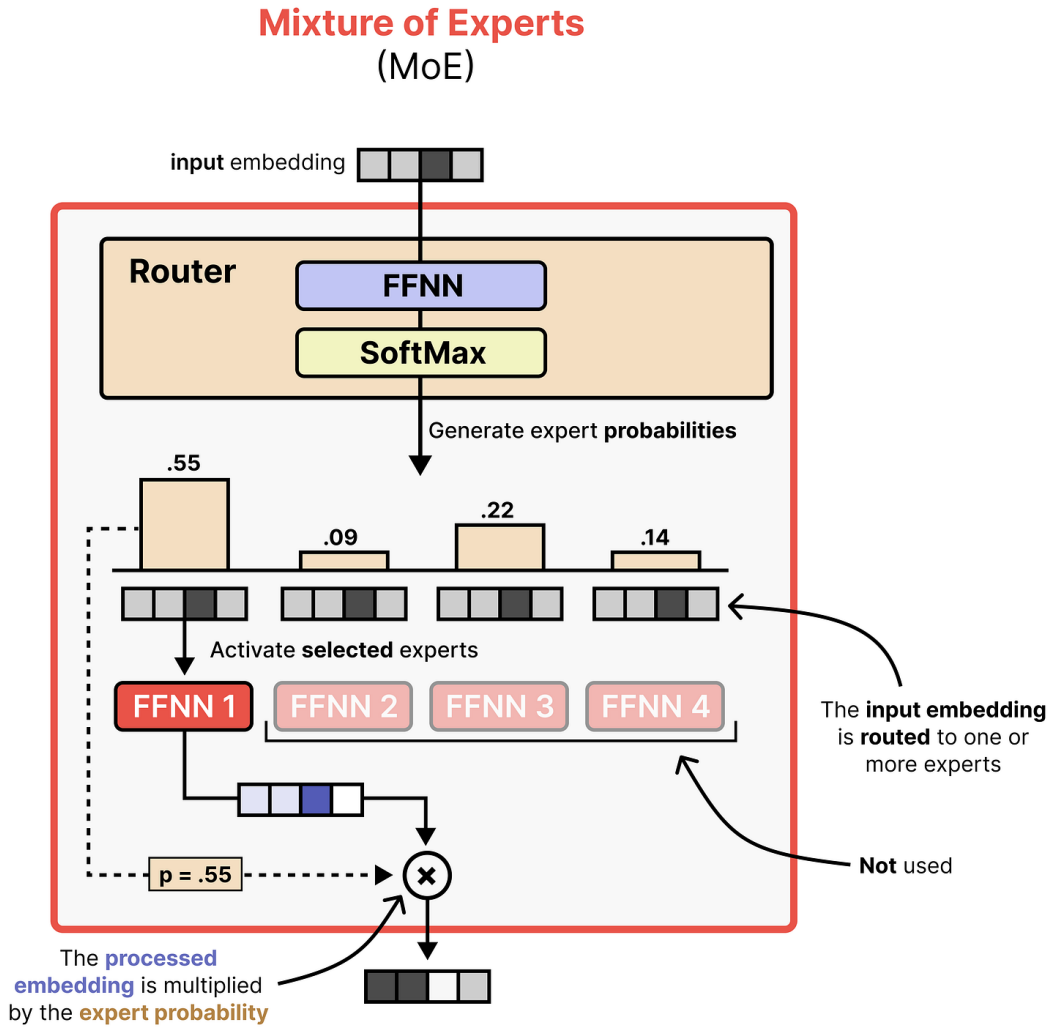
RoPE only on the first %p of the pairs



Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4

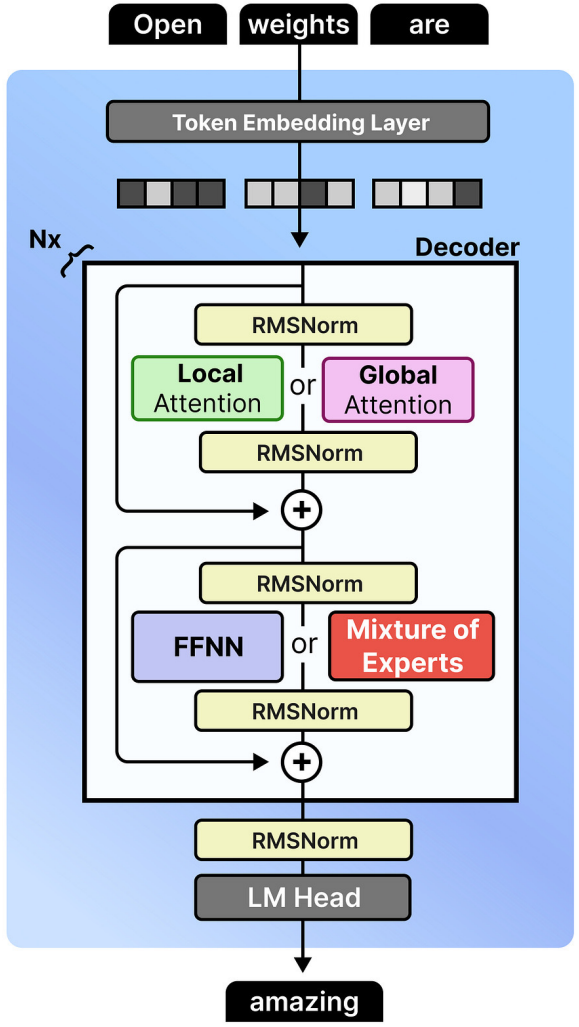


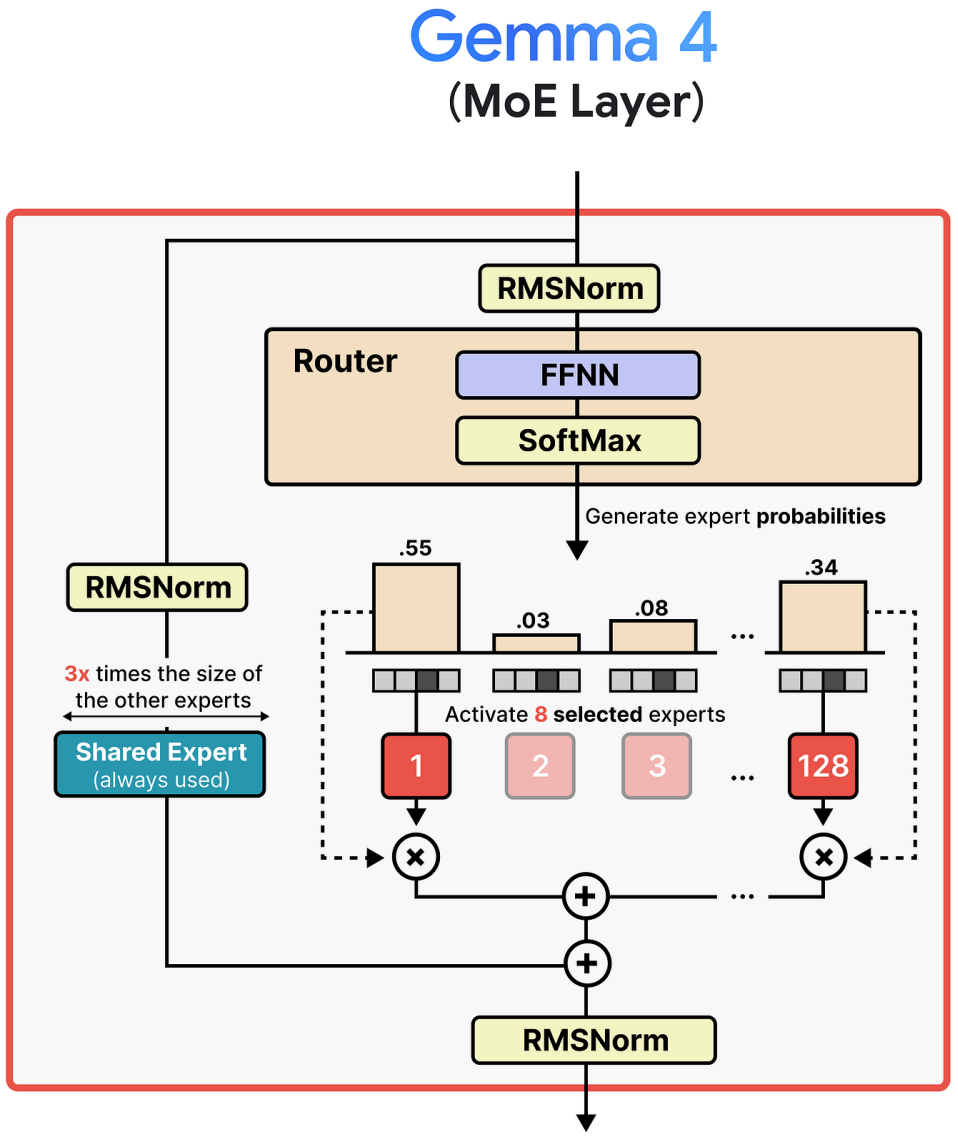


- A large FFNN is replaced by a set of smaller FFNNs.
- The same performance is achieved with less number of calculations.
- Cons: Requires more RAM to fit all experts.
- Load balancing losses required to ensure that a single expert does not dominate (”routing collapse”)

Material: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-gemma-4

Other LLMs: Gemma 4





In-Context Learning (Prompt Engineering)

In-context Learning (Prompt Engineering)

Three ways of in-context learning:

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

In a single sequence input, the prompted example can learn from previous demonstrations.

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

In-context Learning

- Simplifies providing examples (from human experts)
  - As opposed to updating the weights of the network via finetuning
- “Example-based Specification”, Programming by example.
- Provides on par performance with supervised models
- More params, better in-context learning

Circulation revenue has increased by 5% in Finland. // Finance

They defeated … in the NFC Championship Game. // Sports

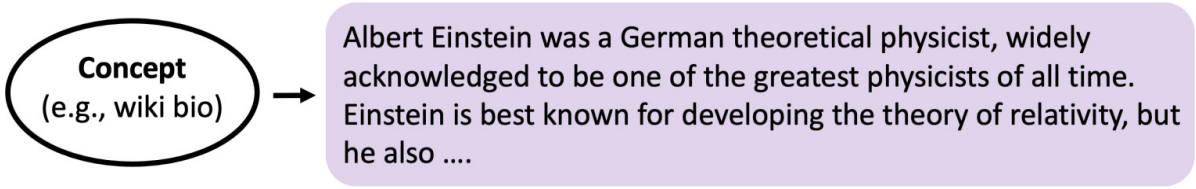
Apple … development of in-house chips. // Tech

The company anticipated its operating profit to improve. // _____

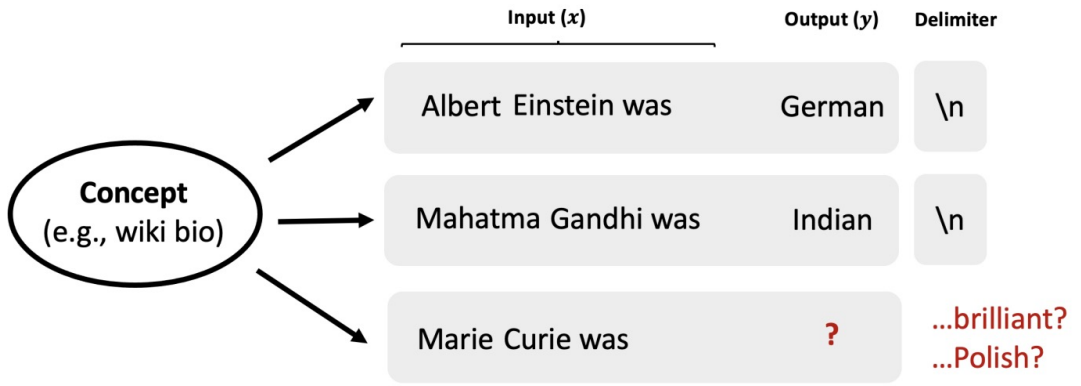
https://ai.stanford.edu/blog/understanding-incontext/

In-context Learning: How/Why Does it Work?

1. Pretraining documents are conditioned on a **latent concept** (e.g., biographical text)



2. Create **independent examples** from a **shared concept.** If we focus on full names, wiki bios tend to relate them to nationalities.



3. Concatenate examples into a prompt and predict next word(s). **Language model (LM) implicitly infers the shared concept** across examples despite the unnatural concatenation

Xie et al., "An Explanation of In-context Learning as Implicit Bayesian Inference", ICLR 2022.

In-context Learning: How/Why Does it Work?

A Bayesian interpretation:

$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt}) p(\text{concept}|\text{prompt}) d(\text{concept})$$

- During pretraining, the network learns a latent concept space.
- With the prompt, we provide sufficient examples to estimate the most relevant concept – p(concept | prompt).

Xie et al., "An Explanation of In-context Learning as Implicit Bayesian Inference", ICLR 2022.

In-context Learning: How/Why Does it Work?

Circulation revenue has increased by 5% in Finland. // Finance

They defeated … in the NFC Championship Game. // Sports

Apple … development of in-house chips. // Tech

Not an easy task since examples might be unrelated to each other!

"We can think of the training examples as providing a signal for Bayesian inference. In particular, the transitions within training examples (green in the figure above) allow the LM to infer the latent concept they all share. In a prompt, the green transitions come from the input distribution (the transitions inside the news sentence), the output distribution (the topic word), the format (syntax of news sentence), and the input-output mapping (relation between the news and the topic) all provide signal for Bayesian inference."

Xie et al., "An Explanation of In-context Learning as Implicit Bayesian Inference", ICLR 2022.

In-context learning: Task vectors

Hendel et al., "In-Context Learning Creates Task Vectors", 2023.

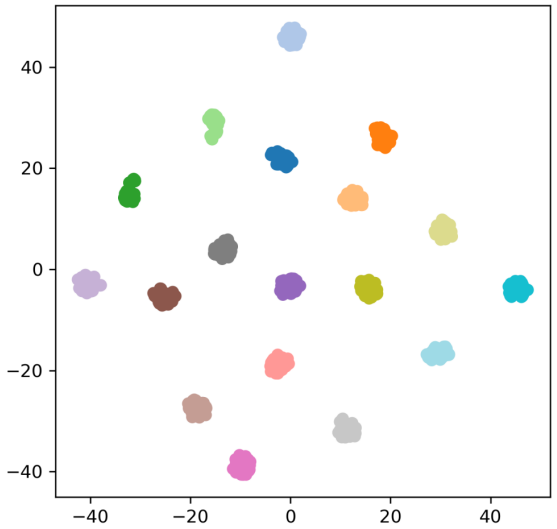


Figure 5: **A t-SNE plot of task vectors.** A 2D t-SNE plot visualizing 50 task vectors for each task, each generated from a different choice of S and x' using LLaMA 7B. Points are color-coded according to the task. Each task can be seen to form its own distinct cluster.

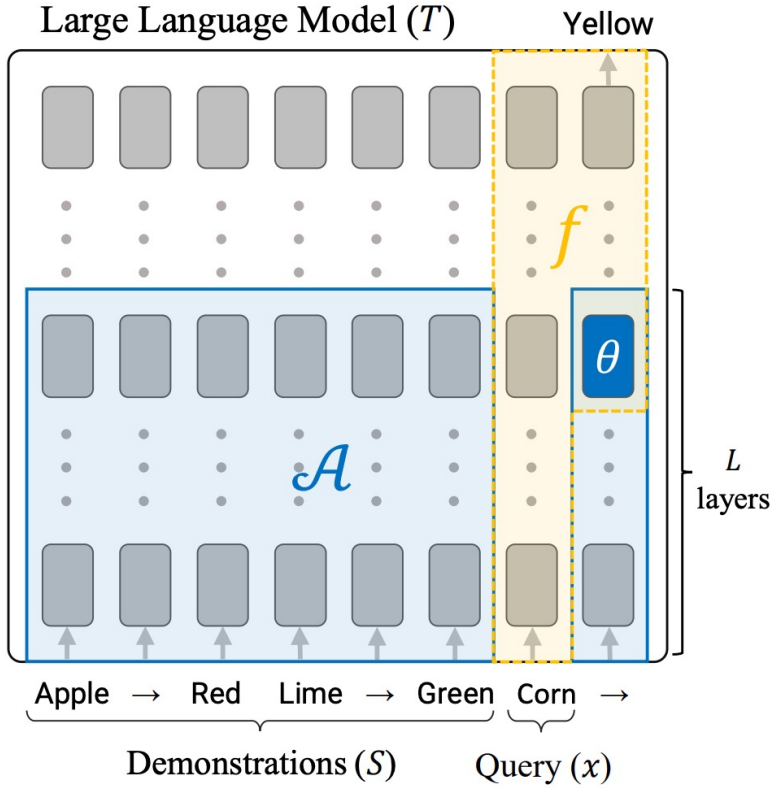
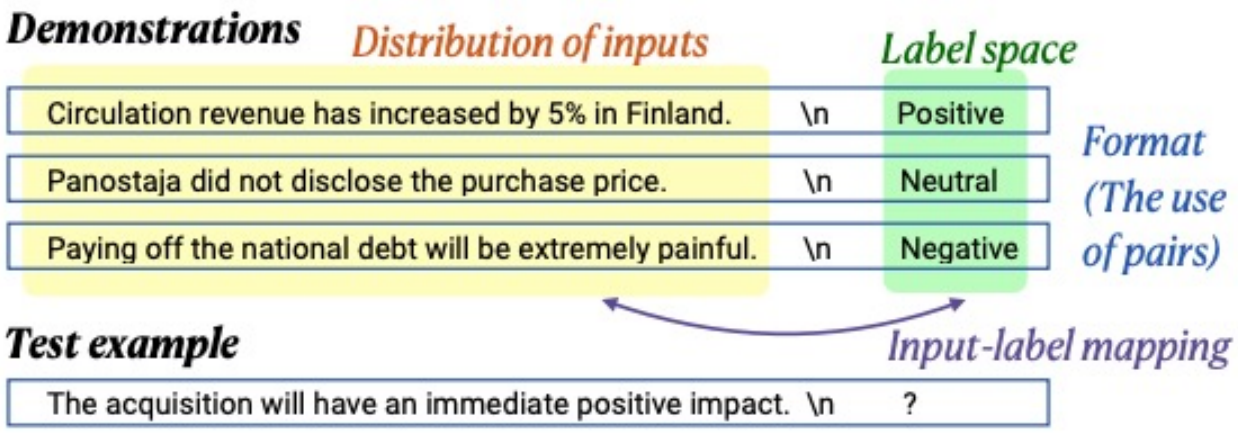


Figure 1: **ICL as learning in a Hypothesis Class.** In ICL, one provides an LLM with a prompt including demonstrations S of some task, and a query x. The model generates the output for x (here "Yellow"). We show that the underlying process can be broken down into two parts: $\mathcal{A}$, a "learning algorithm" (marked in blue), computes a query-agnostic vector $\boldsymbol{\theta}(S)$, which we view as a parameter of a function in a hypothesis class. The second part, denoted by f and marked in yellow, is the application of the rule defined by $\boldsymbol{\theta}$ on the query x, without direct dependence on S.

In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.



Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Together, our analysis provides a new way of understanding how and why in-context learning works, while opening up new questions about how much can be learned from large language models through inference alone.

In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

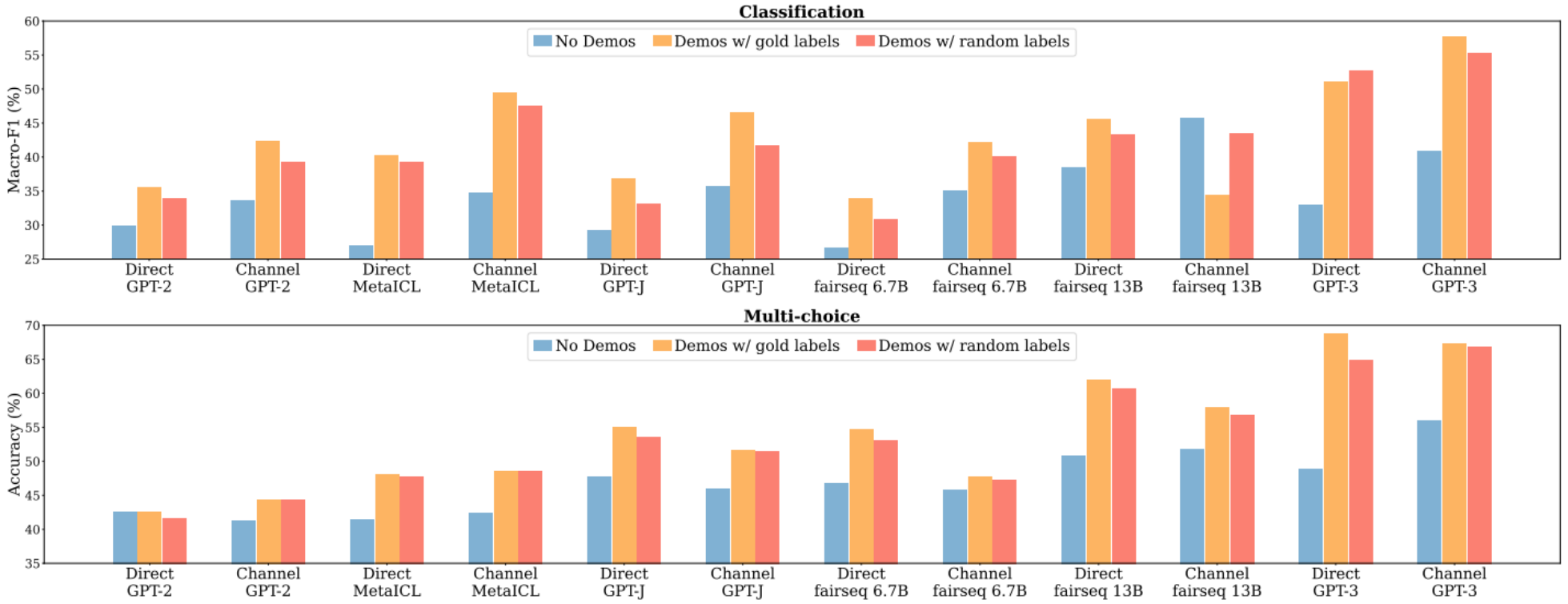


Figure 3: Results when using no-demonstrations, demonstrations with gold labels, and demonstrations with random labels in classification (top) and multi-choice tasks (bottom). The first eight models are evaluated on 16 classification and 10 multi-choice datasets, and the last four models are evaluated on 3 classification and 3 multi-choice datasets. See Figure 11 for numbers comparable across all models. **Model performance with random labels is very close to performance with gold labels** (more discussion in Section 4.1).

In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

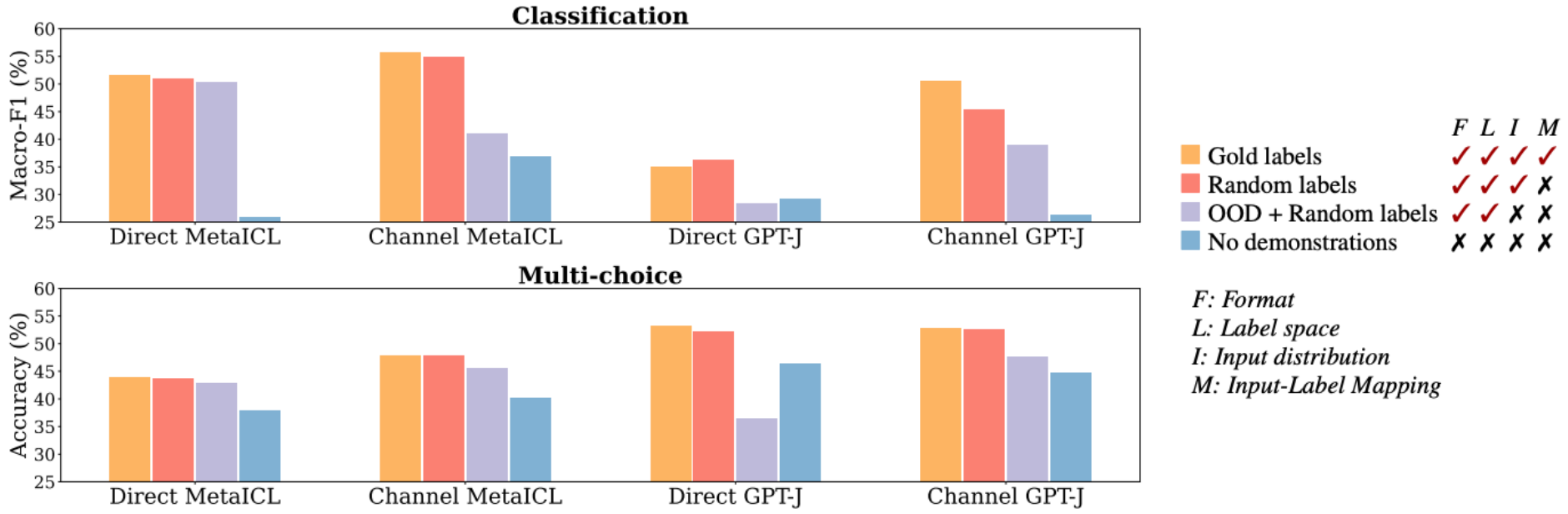


Figure 8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 5.1).

In-context Learning: Important Factors

Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?”, 2022.

They conclude that

(1) “the label space and the distribution of the input text specified by the demonstrations are both key to in-context learning (regardless of whether the labels are correct for individual inputs)”

(2) “specifying the overall format is also crucial, e.g., when the label space is unknown, using random English words as labels is significantly better than using no labels”

(3) “meta-training with an in-context learning objective (Min et al., 2021b) magnifies these effects—the models almost exclusively exploit simpler aspects of the demonstrations like the format rather than the input-label mapping.”

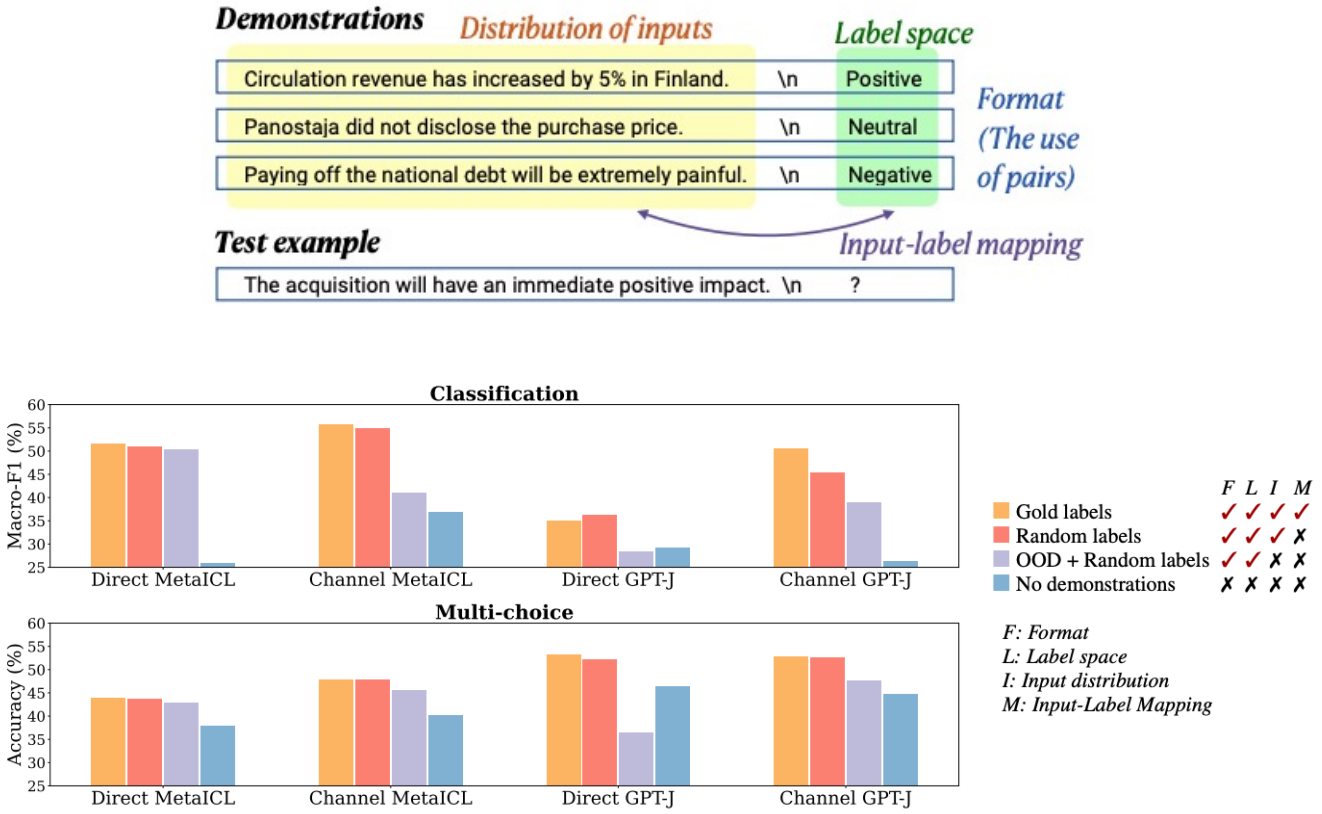


Figure 8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 5.1).

In-context Learning: Important Factors

In-context Learning (ICL) can be unstable:

- Min et al. (https://arxiv.org/abs/2202.12837):
  - “LLMs rely strongly on superficial cues. ICL acts more as a pattern recognition procedure, than as an actual "learning" procedure: the input-output mappings that are provided allow a model to retrieve similar examples it has been exposed to during training, but the moment you start flipping labels or the template model performance breaks.”

- Weber et al. (https://arxiv.org/abs/2310.13486):
  - “Previous research has also shown that ICL is highly unstable. For example, the order of in-context examples (Lu et al., 2022), the recency of certain labels in the context (Zhao et al., 2021) or the format of the prompt (Mishra et al., 2022) as well as the distribution of training examples and the label space (Min et al., 2022) strongly influence model performance. Curiously, whether the labels provided in the examples are correct is less important (Min et al., 2022). However, these findings are not uncontested: Yoo et al. (2022) paint a more differentiated picture, demonstrating that in-context input-label mapping does matter, but that it depends on other factors such as model size or instruction verbosity. Along a similar vein, Wei et al. (2023) show that in-context learners can acquire new semantically non-sensical mappings from in-context examples if presented in a specific setup.”

In-context Learning: Important Factors

In-context Learning (ICL) can be unstable:

- Kossen et al. (https://arxiv.org/pdf/2307.12375v4):
  - Random labels might work with classification problems
  - They significantly impact performance on reasoning/generation tasks

In-context Learning: Limitations

- **Model Parameters and Scale:** Scale helps!
- **Training Data Dependency:** Dataset size matters!
- **Domain Specificity:**
  - ”While LLMs can generalize across various tasks, there might be limitations when dealing with highly specialized domains. Domain-specific data might be required to achieve optimal results.”
- **Model Fine-Tuning:**
  - ”Even with ICL, there might be scenarios where model fine-tuning becomes necessary to cater to specific tasks or correct undesirable emergent abilities.”
- **Ethics and Fairness**
- **Privacy and Security**

https://www.lakera.ai/blog/what-is-in-context-learning

Wei et al., “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, NeurIPS 2022.

Chain of Thought

Standard Prompting

Model Input

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

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Zero-shot Chain of Thought

Kojima et al., “Large Language Models are Zero-Shot Reasoners”, NeurIPS 2022.

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

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

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

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

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

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

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

Meta Prompting

Zhang et al., “Meta Prompting for AI Systems”, 2024.

Standard Prompting

Problem: Find the domain of the expression $\frac{\sqrt{x-2}}{\sqrt{5-x}}$.
Solution: The expressions inside each square root must be non-negative. Therefore, $x-2 \geq 0$, so $x \geq 2$, and $5-x \geq 0$, so $x \leq 5$. Also, the denominator cannot be equal to zero, so $5-x > 0$, which gives $x < 5$. Therefore, the domain of the expression is $\boxed{[2,5)}$. Final Answer: The final answer is $[2,5)$. I hope it is correct.

Problem: If $\det \mathbf{A} = 2$ and $\det \mathbf{B} = 12$, then find $\det(\mathbf{AB})$.
Solution: We have that $\det(\mathbf{AB}) = (\det \mathbf{A})(\det \mathbf{B}) = (2)(12) = \boxed{24}$. Final Answer: The final answer is 24. I hope it is correct.

...

Meta Prompting

Problem Statement:
- **Problem**: [question to be answered]

Solution Structure:
1. Begin the response with "Let's think step by step."
2. Follow with the reasoning steps, ensuring the solution process is broken down clearly and logically.
3. End the solution with the final answer encapsulated in a LaTeX-formatted box, $\boxed{\ldots}$, for clarity and emphasis.
4. Finally, state "The answer is [final answer to the problem].", with the final answer presented in LaTeX notation.

Figure 1: A structure meta prompt presented in markdown format for solving MATH [17] problems.

Chain of Symbol

Hu et al., “Chain-of-Symbol Prompting Elicits Planning in Large Language Models”, 2023.

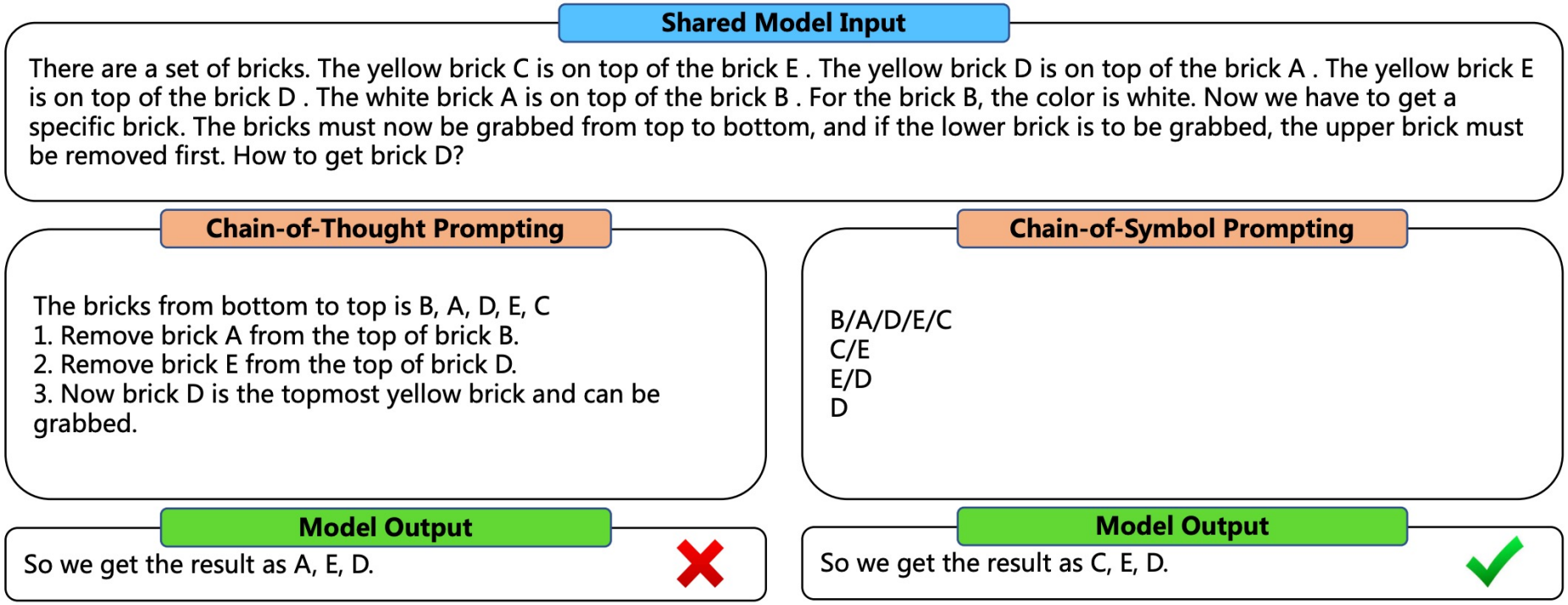


Figure 1: An example for comparison between Chain-of-Thought (CoT) and Chain-of-Symbol (CoS) that elicits large language models in tackling complex planning tasks with higher performance and fewer input tokens. We let the model generate CoT/CoS during inference in a few-shot manner. Results were taken in May 2023 with ChatGPT and can be subject to change.

Chain of Symbol

Hu et al., “Chain-of-Symbol Prompting Elicits Planning in Large Language Models”, 2023.

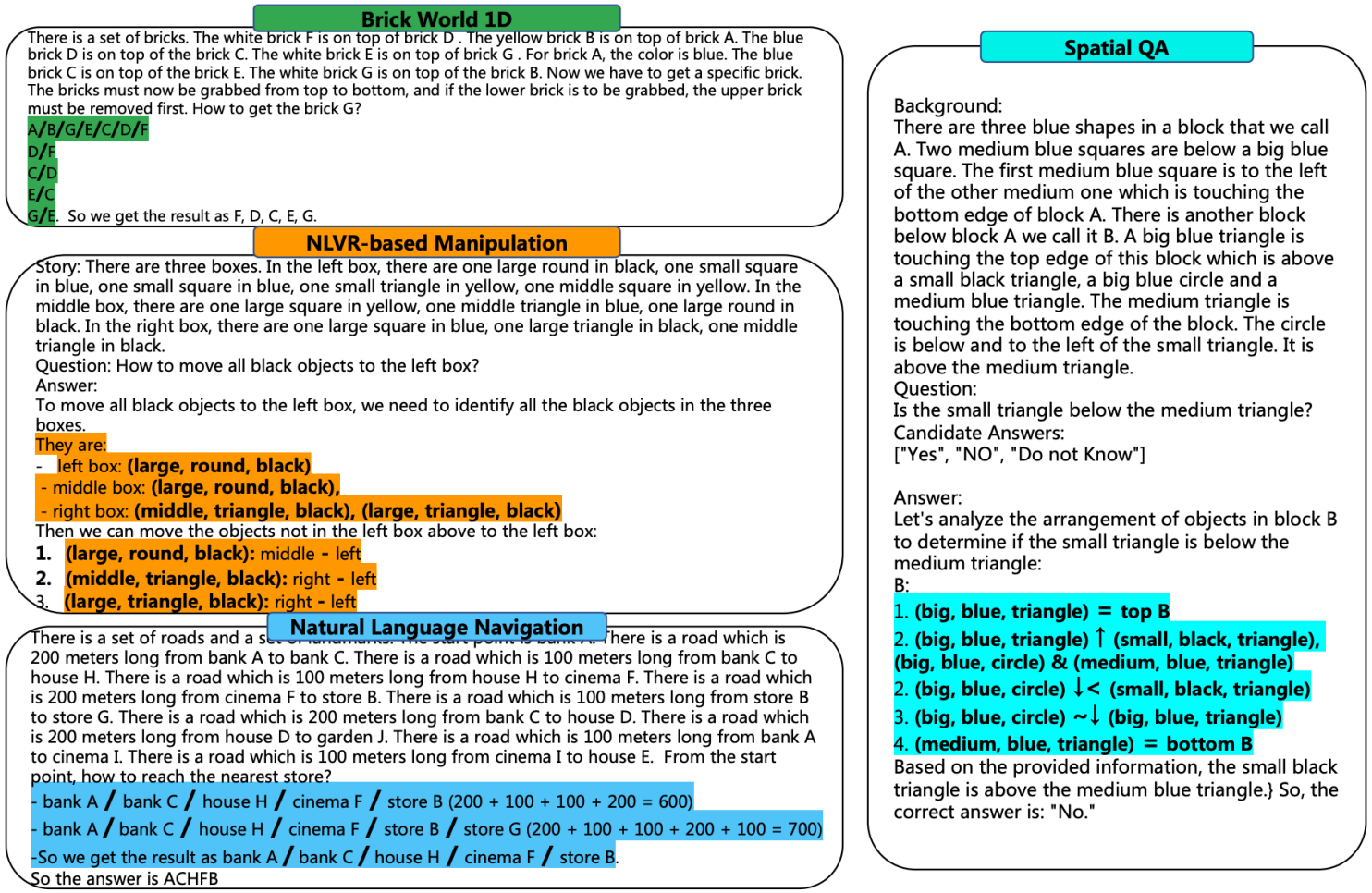


Figure 2: <input, Chain of Symbol, output> example triples for our three proposed tasks: Brick World, NLVR-based Manipulation, and Natural Language Navigation, and SPARTUN dataset (Mirzaee & Kordjamshidi, 2022). Chains of Symbols are highlighted.

NLVR: Natural Language Visual Reasoning

Generated knowledge prompting

Liu et al., "Generated Knowledge Prompting for Commonsense Reasoning", 2022.

It remains an open question whether incorporating external knowledge benefits commonsense reasoning while maintaining the flexibility of pretrained sequence models. To investigate this question, we develop generated knowledge prompting, which consists of generating knowledge from a language model, then providing the knowledge as additional input when answering a question. Our method does not require task-specific supervision for knowledge integration, or access to a structured knowledge base, yet it improves performance of large-scale, state-of-the-art models on four commonsense reasoning tasks, achieving state-of-the-art results on numerical commonsense (NumerSense), general commonsense (CommonsenseQA 2.0), and scientific commonsense (QASC) benchmarks. Generated knowledge prompting highlights large-scale language models as flexible sources of external knowledge for improving common-

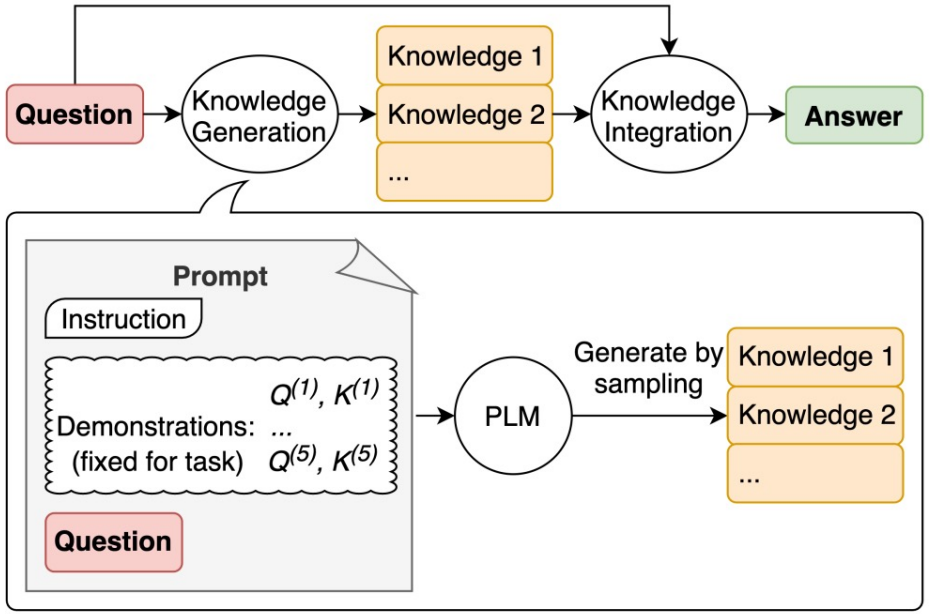


Figure 1: Generated knowledge prompting involves (i) using few-shot demonstrations to generate question-related knowledge statements from a language model; (ii) using a second language model to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

PLM: Pre-trained Language Model

Liu et al., “Generated Knowledge Prompting for Commonsense Reasoning”, 2022.

Generated knowledge prompting

```
Input: Greece is larger than mexico.
Knowledge: Greece is approximately 131,957 sq km, while Mexico is approximately 1,964,375 sq km, making Mexico 1,389% larger than Greece.
Input: Glasses always fog up.
Knowledge: Condensation occurs on eyeglass lenses when water vapor from your sweat, breath, and ambient humidity lands on a cold surface, cools, and then changes into tiny drops of liquid, forming a film that you see as fog. Your lenses will be relatively cool compared to your breath, especially when the outside air is cold.
Input: A fish is capable of thinking.
Knowledge: Fish are more intelligent than they appear. In many areas, such as memory, their cognitive powers match or exceed those of ’higher’ vertebrates including non-human primates. Fish’s long-term memories help them keep track of complex social relationships.
Input: A common effect of smoking lots of cigarettes in one’s lifetime is a higher than normal chance of getting lung cancer.
Knowledge: Those who consistently averaged less than one cigarette per day over their lifetime had nine times the risk of dying from lung cancer than never smokers. Among people who smoked between one and 10 cigarettes per day, the risk of dying from lung cancer was nearly 12 times higher than that of never smokers.
Input: A rock is the same size as a pebble.
Knowledge: A pebble is a clast of rock with a particle size of 4 to 64 millimetres based on the Udden-Wentworth scale of sedimentology. Pebbles are generally considered larger than granules (2 to 4 millimetres diameter) and smaller than cobbles (64 to 256 millimetres diameter).
Input: Part of golf is trying to get a higher point total than others.
Knowledge:
```

Self-consistency

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

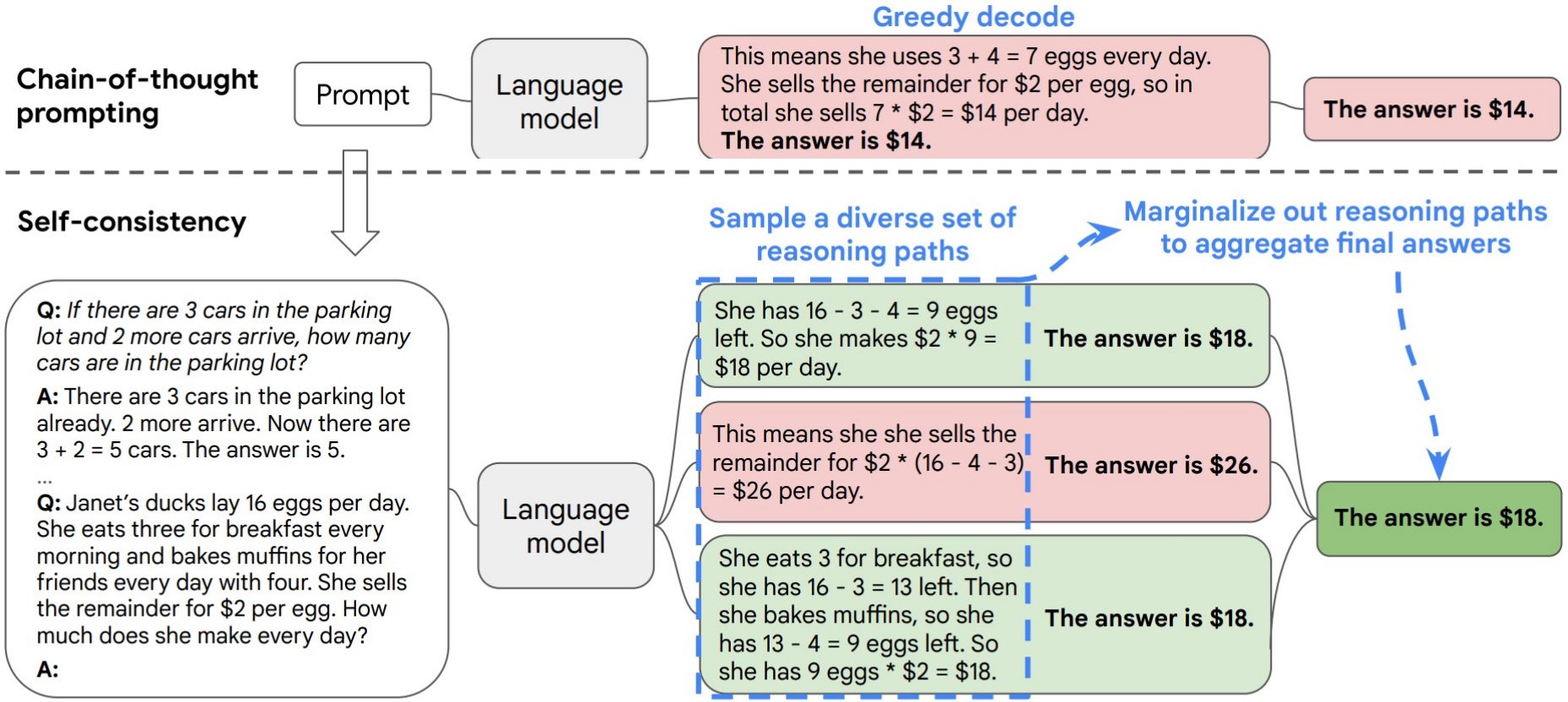
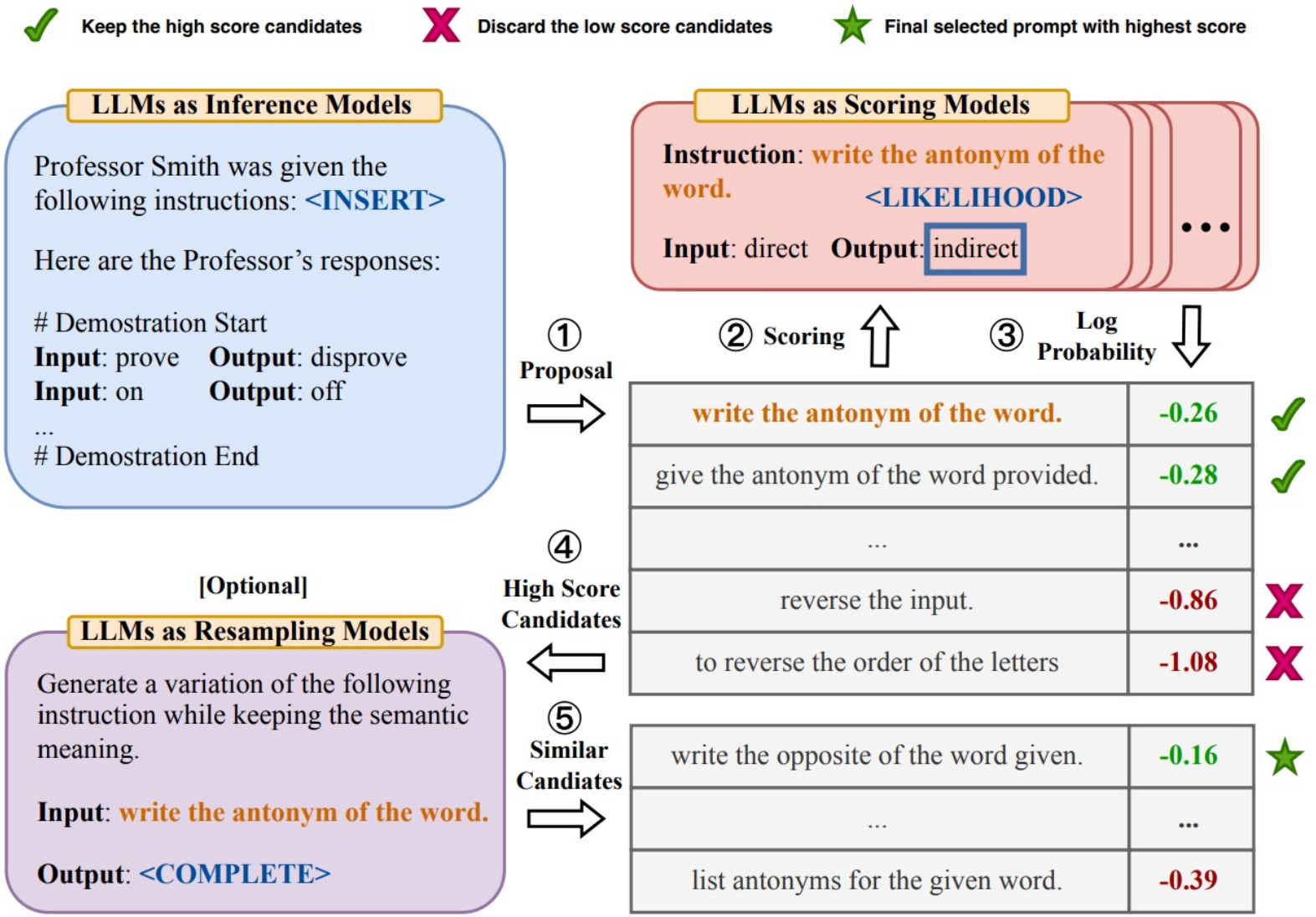


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

Automatic Prompt Engineer

Zhou et al., "LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS", 2023.



Tree of Thoughts Prompting

Yao et al., "Tree of Thoughts: Deliberate Problem Solving with Large Language Models", 2023.

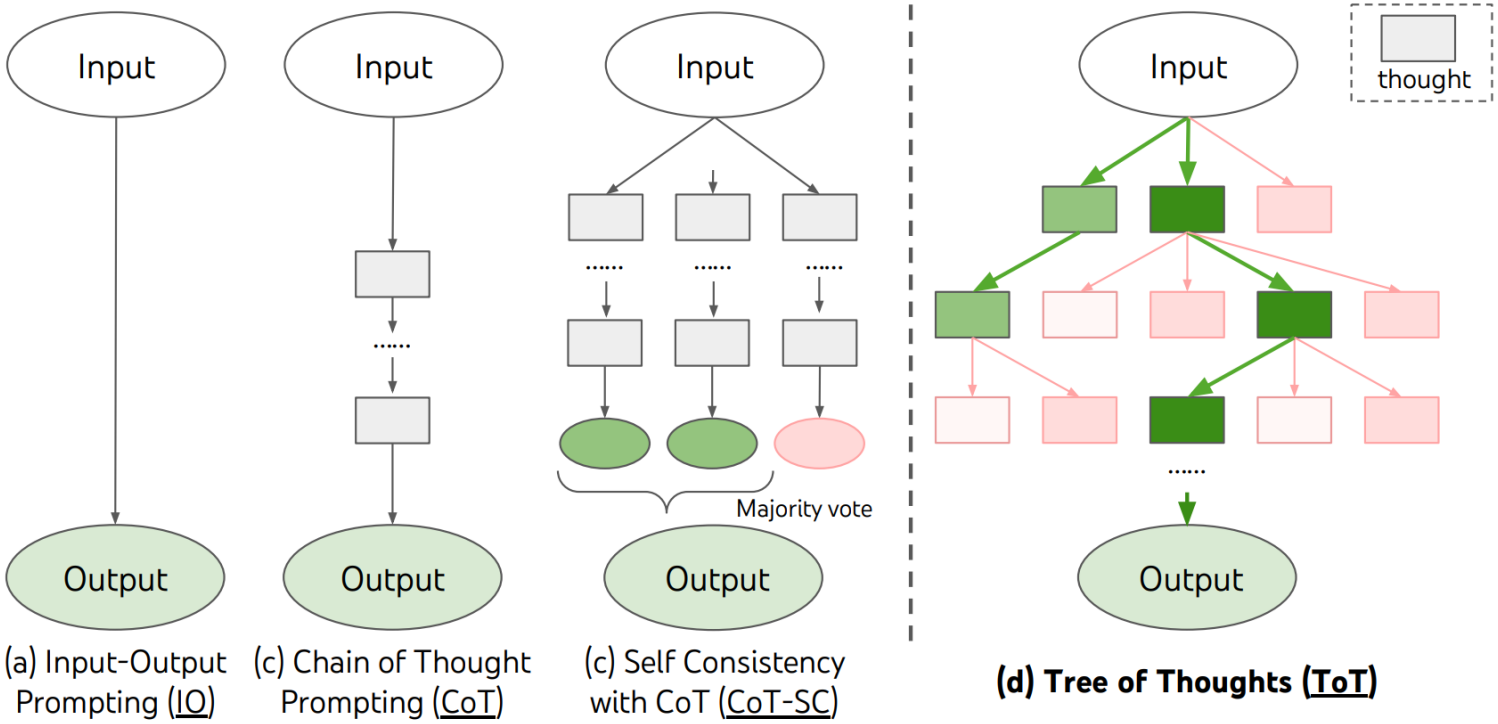
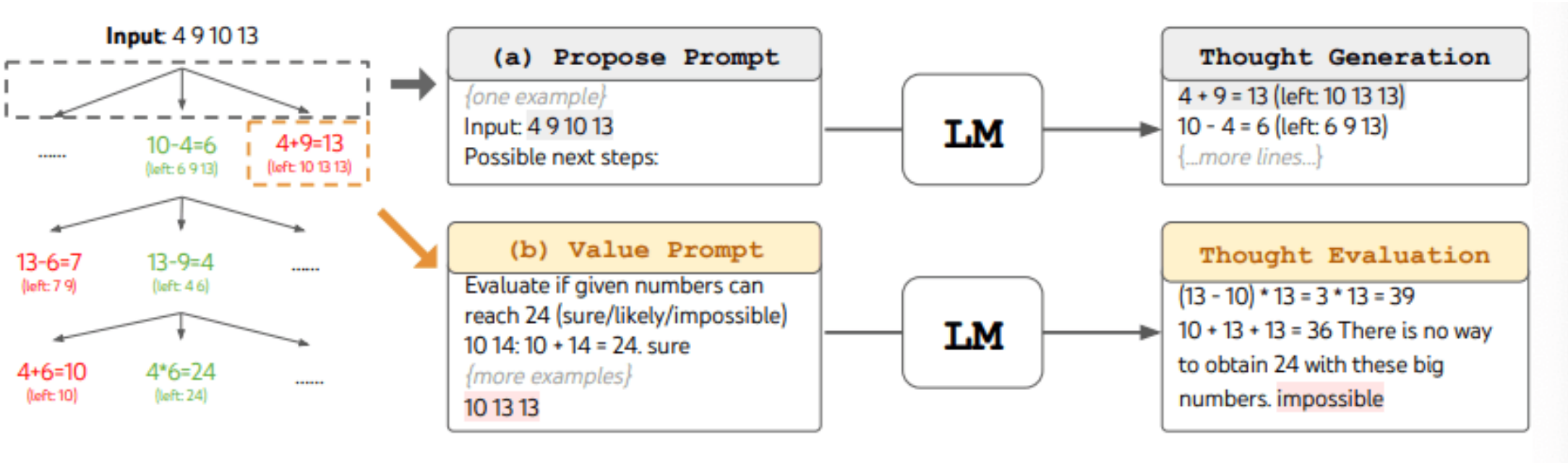


Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

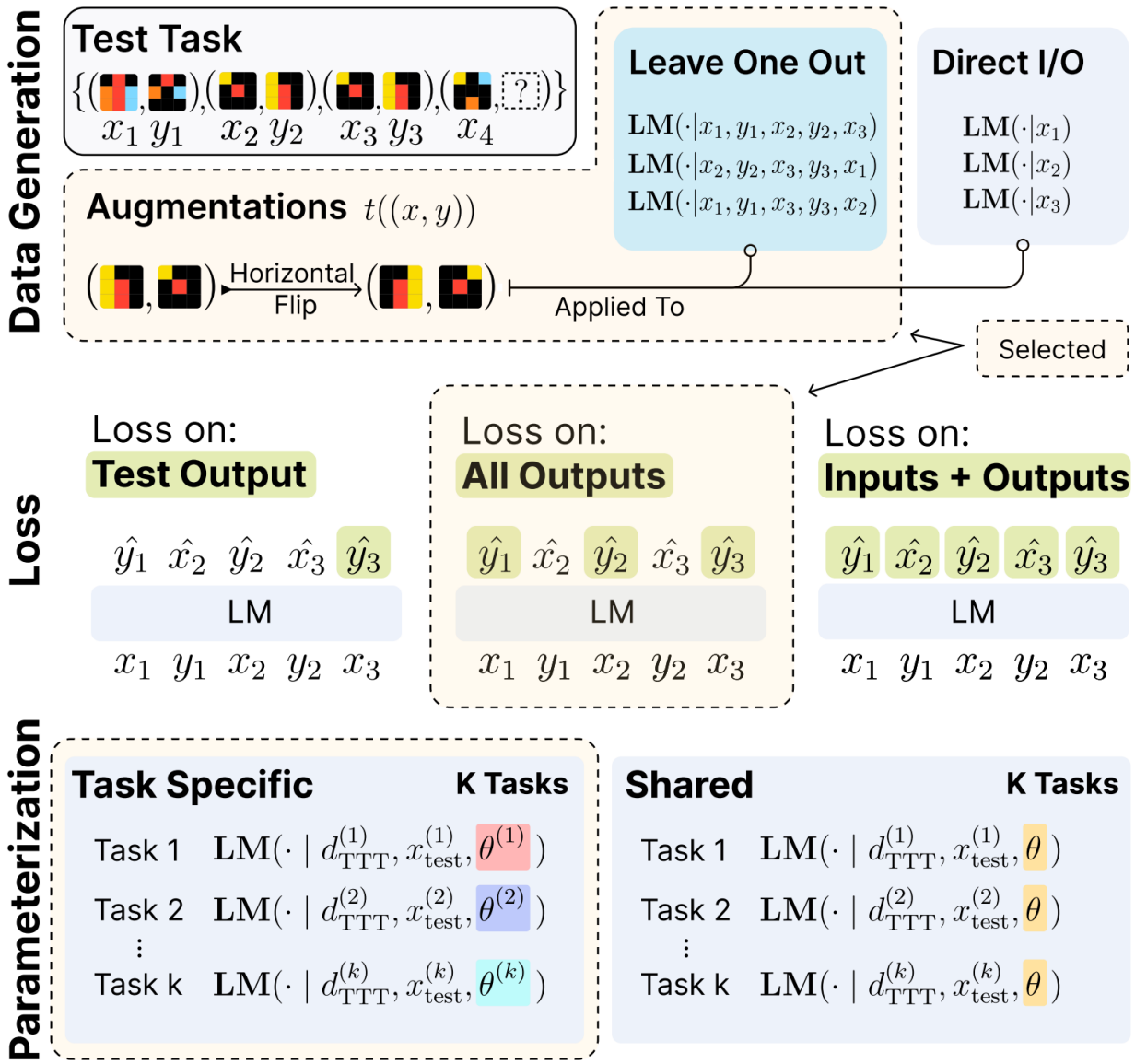
Tree of Thoughts Prompting

Yao et al., “Tree of Thoughts: Deliberate Problem Solving with Large Language Models”, 2023.



Test-Time Compute and Test-Time Training

- Test-Time Compute (Inference):
  - Use many smaller LLMs to estimate multiple paths/solutions in parallel
  - LLM weights are frozen
- Test-Time Training
  - Perform some self-supervised update/adaptation based on the input/task



https://openreview.net/pdf?id=asgBo3FNdg

Program-Aided LMs

Gao et al., “PAL: Program-aided Language Models”, 2023.

Chain-of-Thought (Wei et al., 2022)

Input

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 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold 93 + 39 = 132 loaves. The grocery store returned 6 loaves. So they had 200 - 132 - 6 = 62 loaves left.
The answer is 62.

Program-aided Language models (this work)

Input

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 tennis balls.
`tennis_balls = 5`
2 cans of 3 tennis balls each is
`bought_balls = 2 * 3`
tennis balls. The answer is
`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves
`loaves_baked = 200`
They sold 93 in the morning and 39 in the afternoon
`loaves_sold_morning = 93`
`loaves_sold_afternoon = 39`
The grocery store returned 6 loaves.
`loaves_returned = 6`
The answer is

```
answer = loaves_baked - loaves_sold_morning
  - loaves_sold_afternoon + loaves_returned
```

```
>>> print(answer)
74
```

Other Types of Prompting

- Least-to-most prompting
- Complexity-based prompting
- Self-refine
- Maieutic prompting
- Using gradient descent to search for prompts: "prefix-tuning”, "prompt tuning" or "soft prompting”
- Prompt injection

Reasoning Models

1. Base Model (e.g., GPT-3)

2. Supervised finetuning
   - with standard input-output pairs
   - with chain-of-thought input-output pairs
   - When asked a math/logic question, LLMs are trained to first output a <thinking> tag, break the problem down into steps, and only then output the <answer>.

https://www.ibm.com/think/topics/reasoning-model

Reasoning Models

3. Reinforcement Learning

- Outcome Reward: for questions involving math/logic/coding, the steps are verified with a program. the outcome is used as a reward
- Process Reward: since rewarding the outcomes only may not be sufficient, reward the correctness of the process. Generally another AI model is used.
- Self-Correction and Alignment: Models are rewarded for correcting their own miscalculations or switching to an incorrect domain (e.g., to a different language).

4. Distillation

- Train a smaller version mimicking the larger one.

https://www.ibm.com/think/topics/reasoning-model

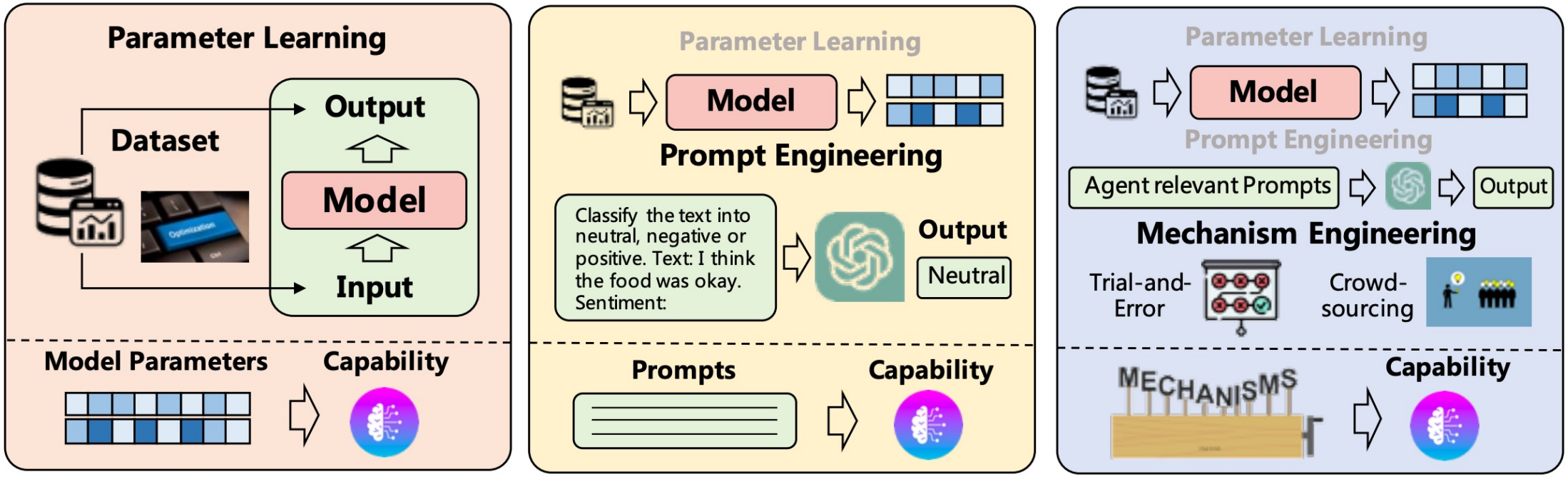
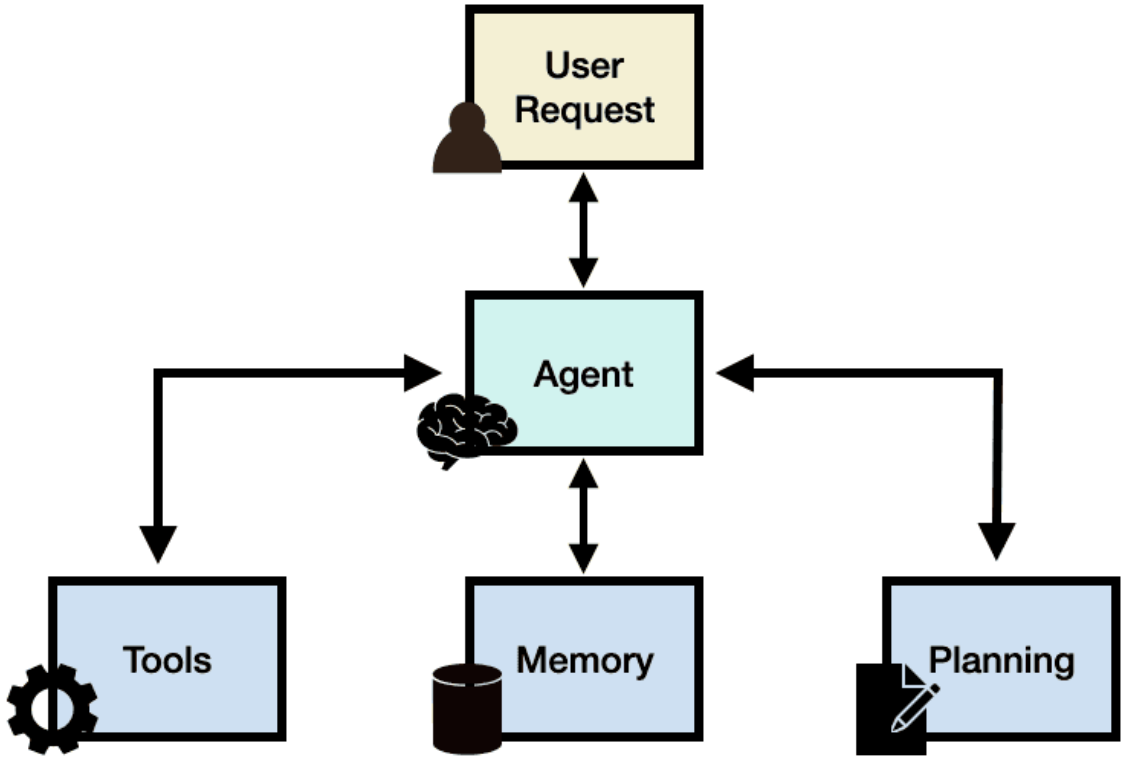


Figure: Wang et al., “A Survey on Large Language Model based Autonomous Agents”, 2024.

LLMs as Agents

a.k.a. Mechanism Engineering

LLMs as Agents



https://www.promptingguide.ai/research/llm-agents

Wang et al., “A Survey on Large Language Model based Autonomous Agents”, 2024.

LLMs as Agents

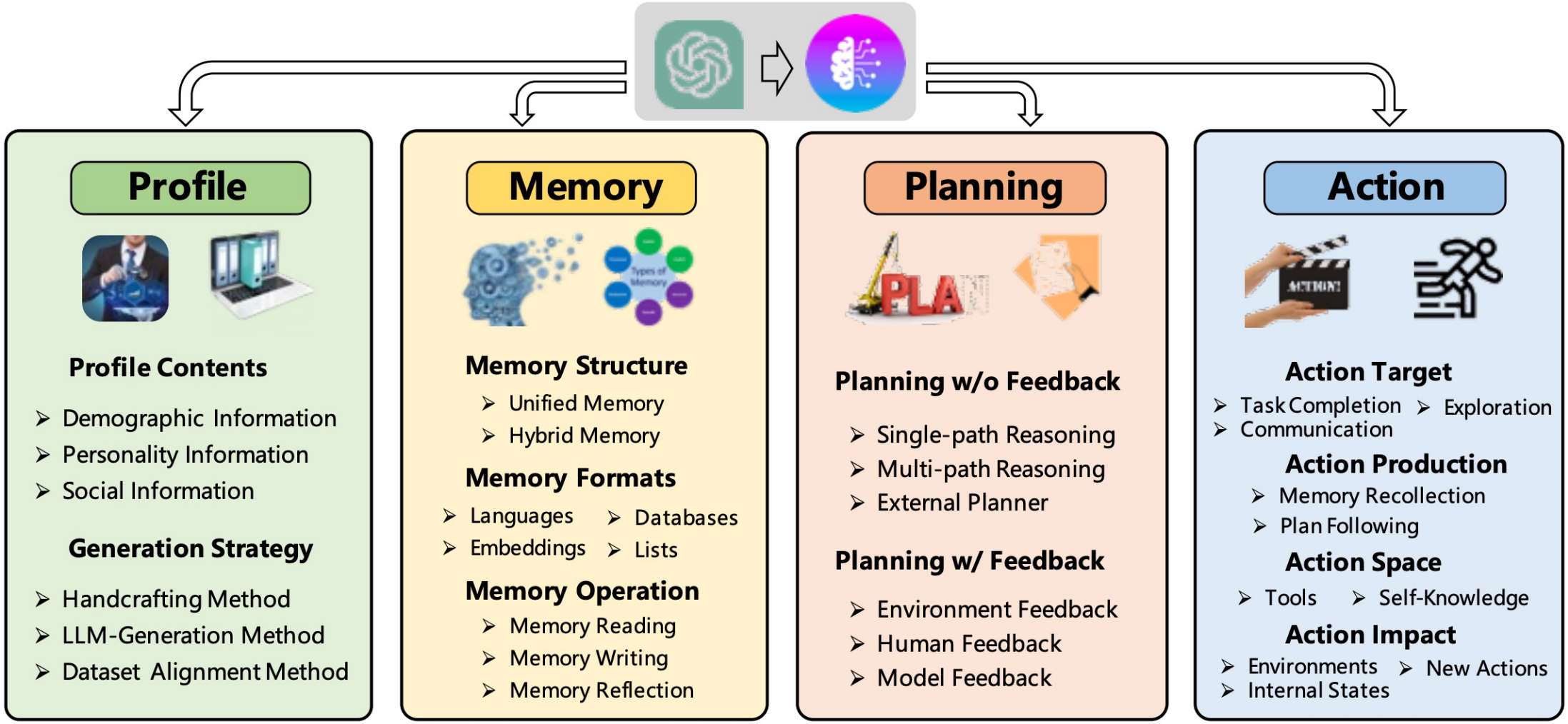


Fig. 2 A unified framework for the architecture design of LLM-based autonomous agent.

Prompting LLMs with Personas

- Guiding an LLM with e.g. “You are a world-class physician”
  - can improve alignment scores,
  - can produce responses more liked by humans
  - can make the model more confident but less accurate
- With such a prompt, the LLM *“adopts the tone, the vocabulary, the confidence, the authoritative register of someone who knows what they are talking about.”*
- The knowledge-level may be different, but “boosted confidence” causes LLM to output a confident answer even in uncertainty

Expert Personas Improve LLM Alignment but Damage Accuracy: Bootstrapping Intent-Based Persona Routing with PRISM

Zizhao Hu **Mohammad Rostami** **Jesse Thomason**
University of Southern California
Los Angeles, California, USA
{zizhaoh, rostamim, jessetho}@usc.edu

Abstract

Persona prompting can steer LLM generation towards a domain-specific tone and pattern. This behavior enables use cases in multi-agent systems where diverse interactions are crucial and human-centered tasks require high-level human alignment. Prior works provide mixed opinions on their utility: some report performance gains when using expert personas for certain domains and their contribution to data diversity in synthetic data creation, while others find near-zero or negative impact on general utility. To fully leverage the benefits of the LLM persona and avoid its harmfulness, a

an expert persona to an LLM can increase behavioral divergence in multi-agent systems (Chen et al., 2026), improve emotional support dialogues (Wu et al., 2025), enable diverse synthetic data generation (Chan et al., 2024), and improve fairness in generation (Gajewska et al., 2025). However, other works find near-zero average benefit on specialized tasks (Zheng et al., 2024; Truong et al., 2025), and role-playing can degrade LLMs’ zero-shot reasoning (Kim et al., 2025). These mixed opinions on using LLM personas motivate a systematic investigation of when and why personas help or hurt.

When it comes to using persona in production, practitioners usually rely on empirical prompting

https://arxiv.org/pdf/2603.18507

Wang et al., “A Survey on Large Language Model based Autonomous Agents”, 2024.

LLMs as Agents

- Prompting/Planning without feedback

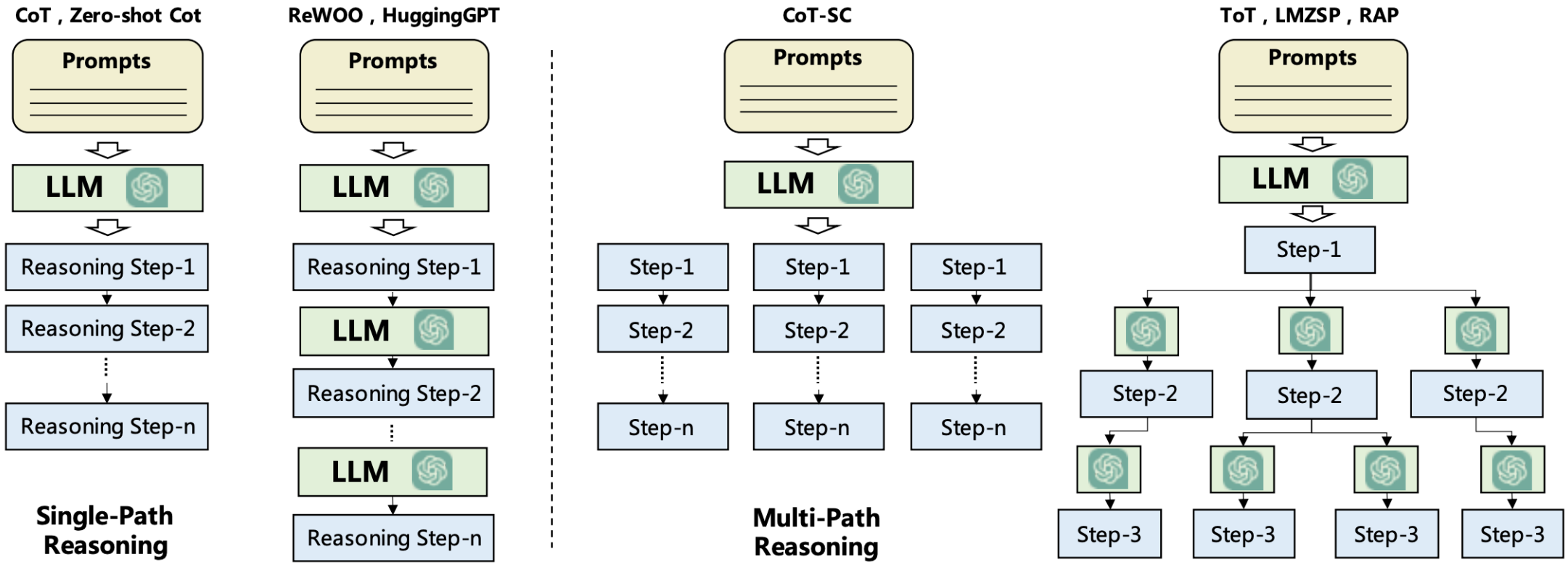
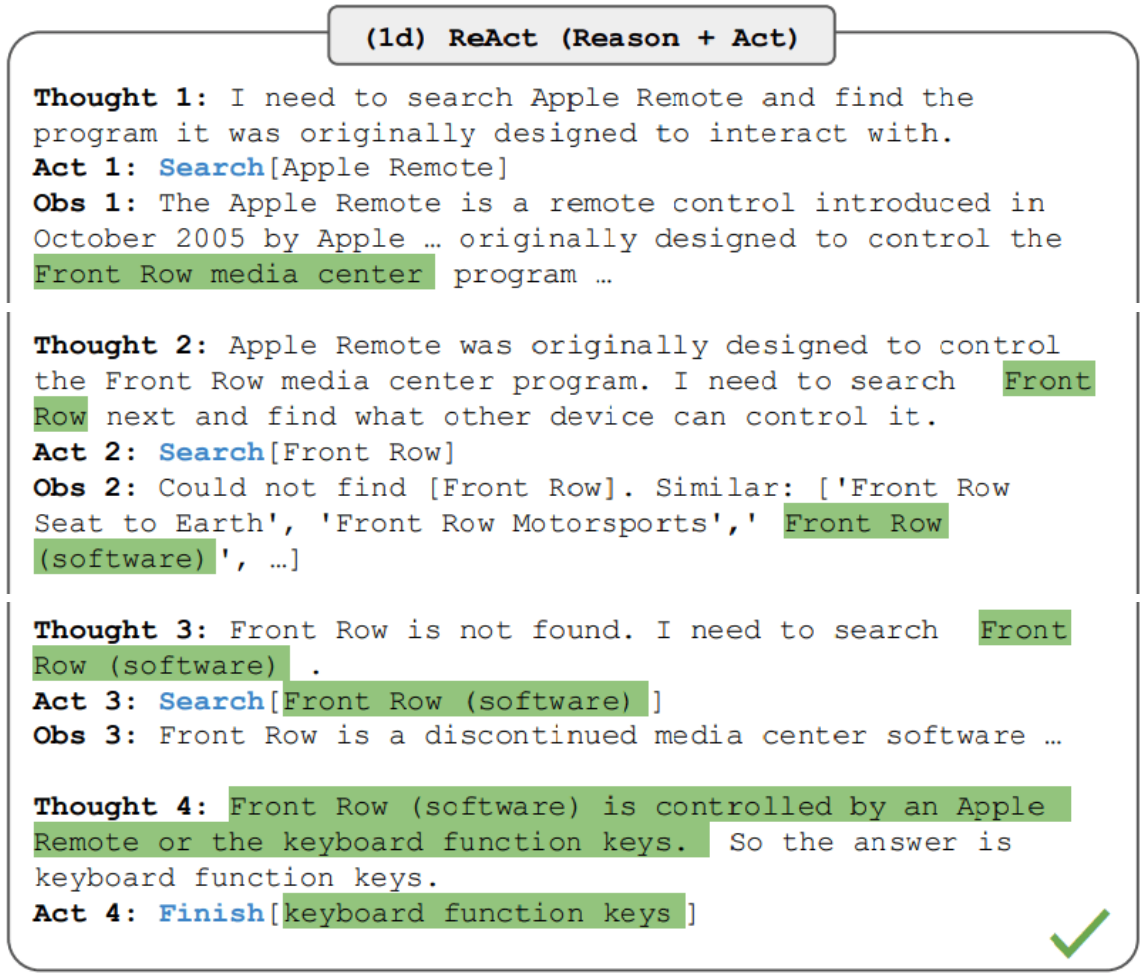


Fig. 3 Comparison between the strategies of single-path and multi-path reasoning. LMZSP is the model proposed in [54].

LLMs as Agents

- Prompting/Planning with feedback



Yao et al., “ReAct: Synergizing Reasoning and Acting in Language Models”, 2023.

LLMs as Agents

Shinn et al., “Reflexion: Language Agents with Verbal Reinforcement Learning”, 2023.

- Prompting/Planning with feedback

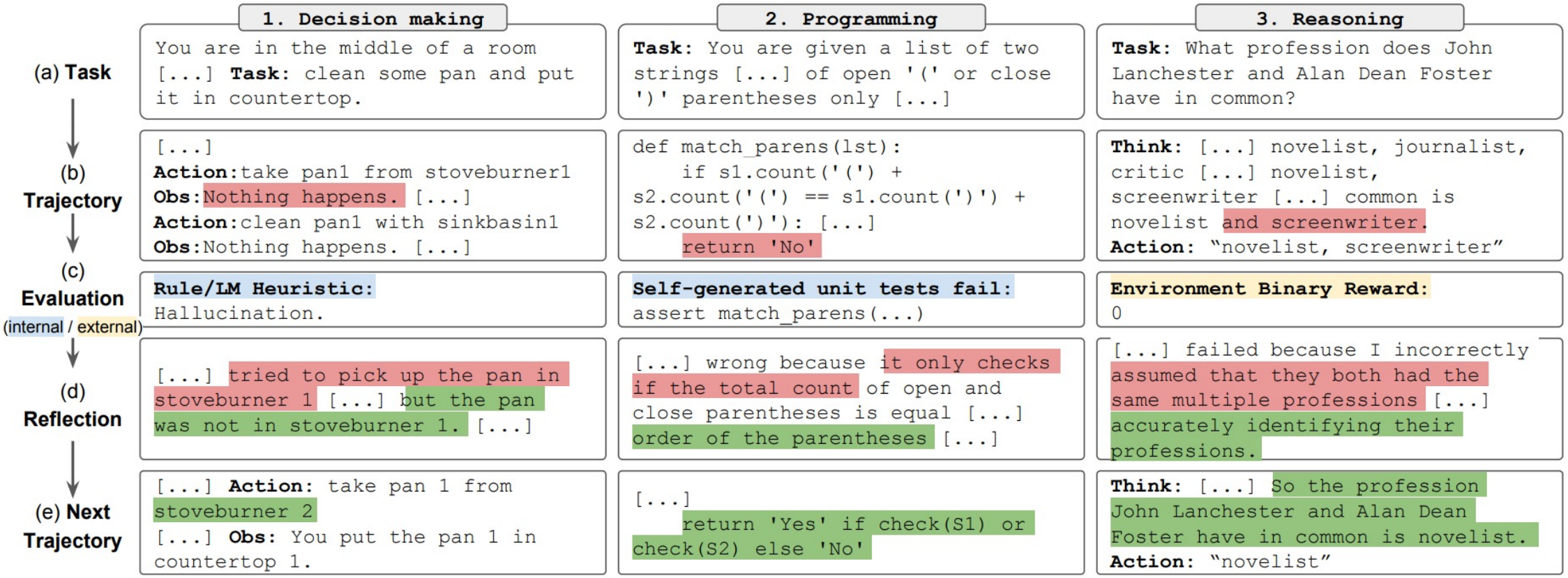
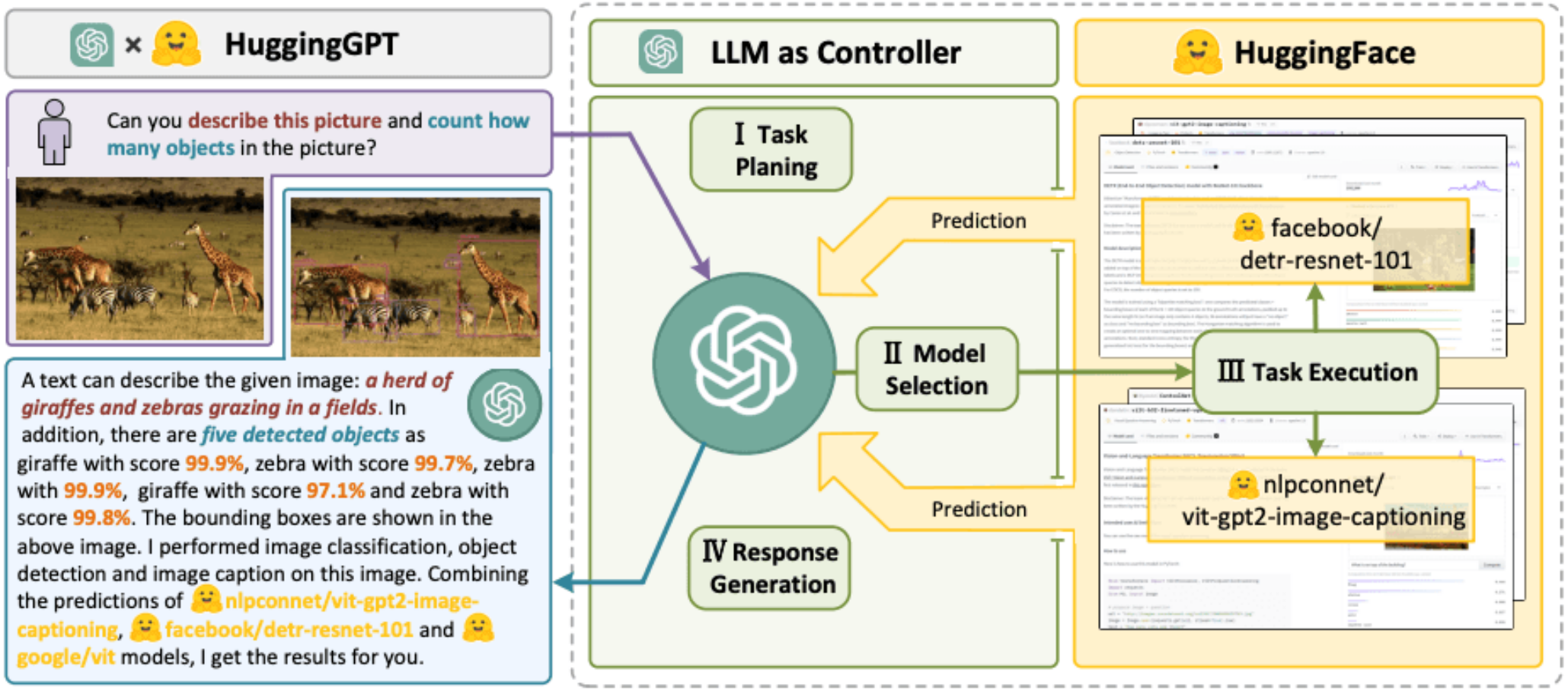


Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.

LLMs as Agents: LLMs and Tools

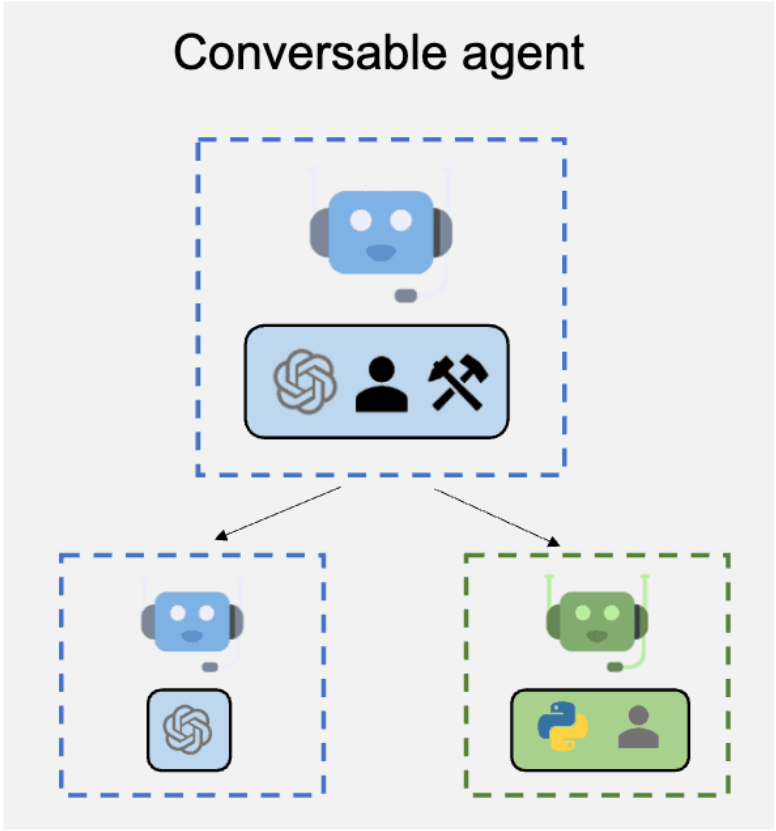
- MRKL, Toolformer, Function Calling, HuggingGPT, …



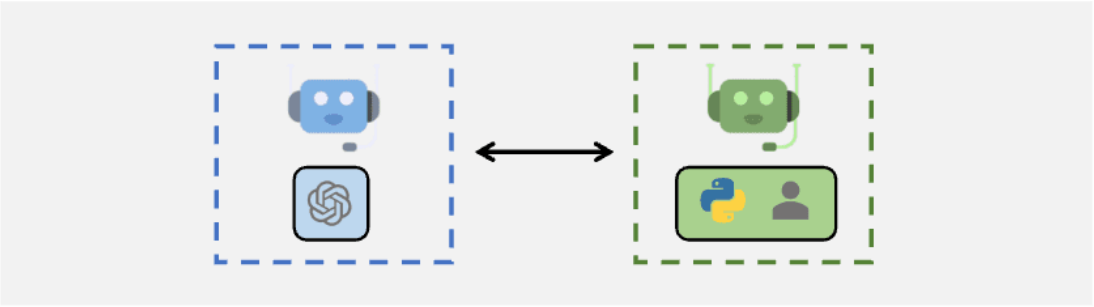
LLMs as Agents

- Autogen, LangChain, AutoGPT, Langroid, OpenAgents, ….

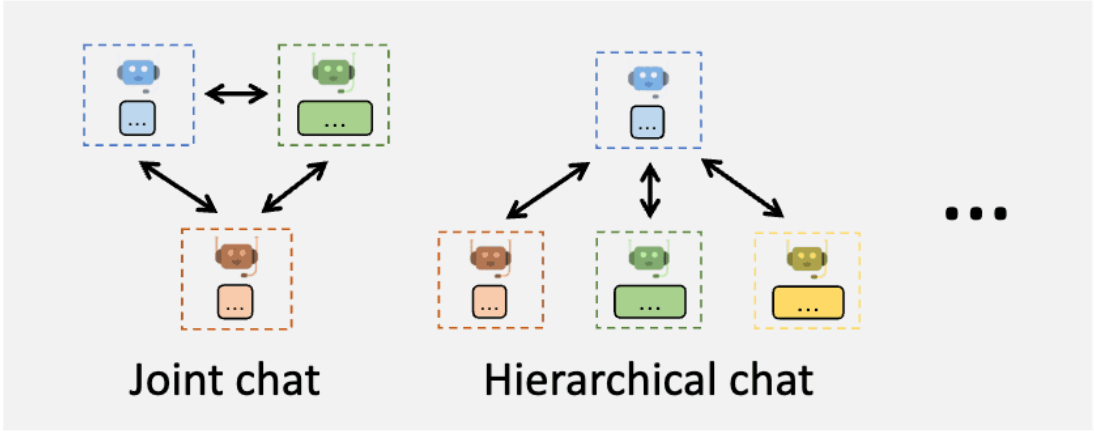
Autogen:



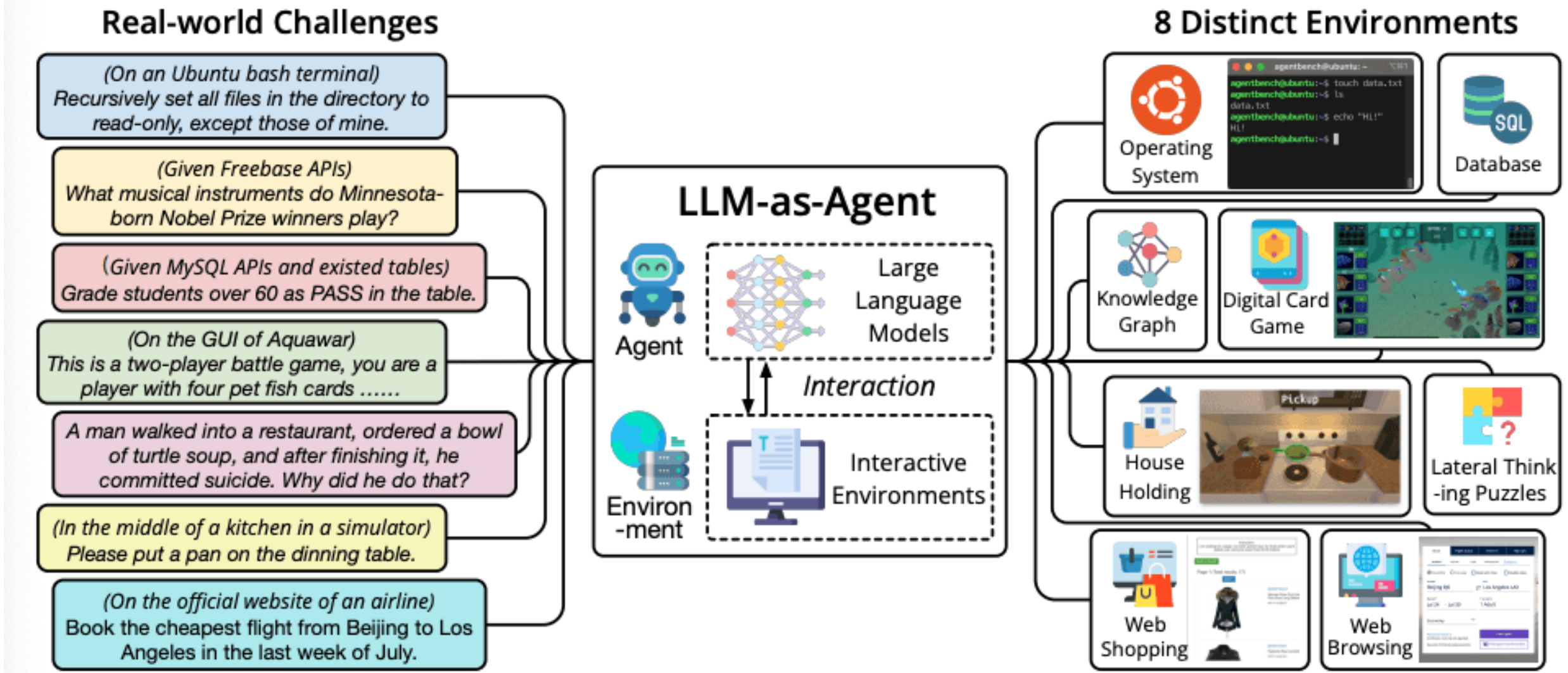
Agent Customization



Multi-Agent Conversations



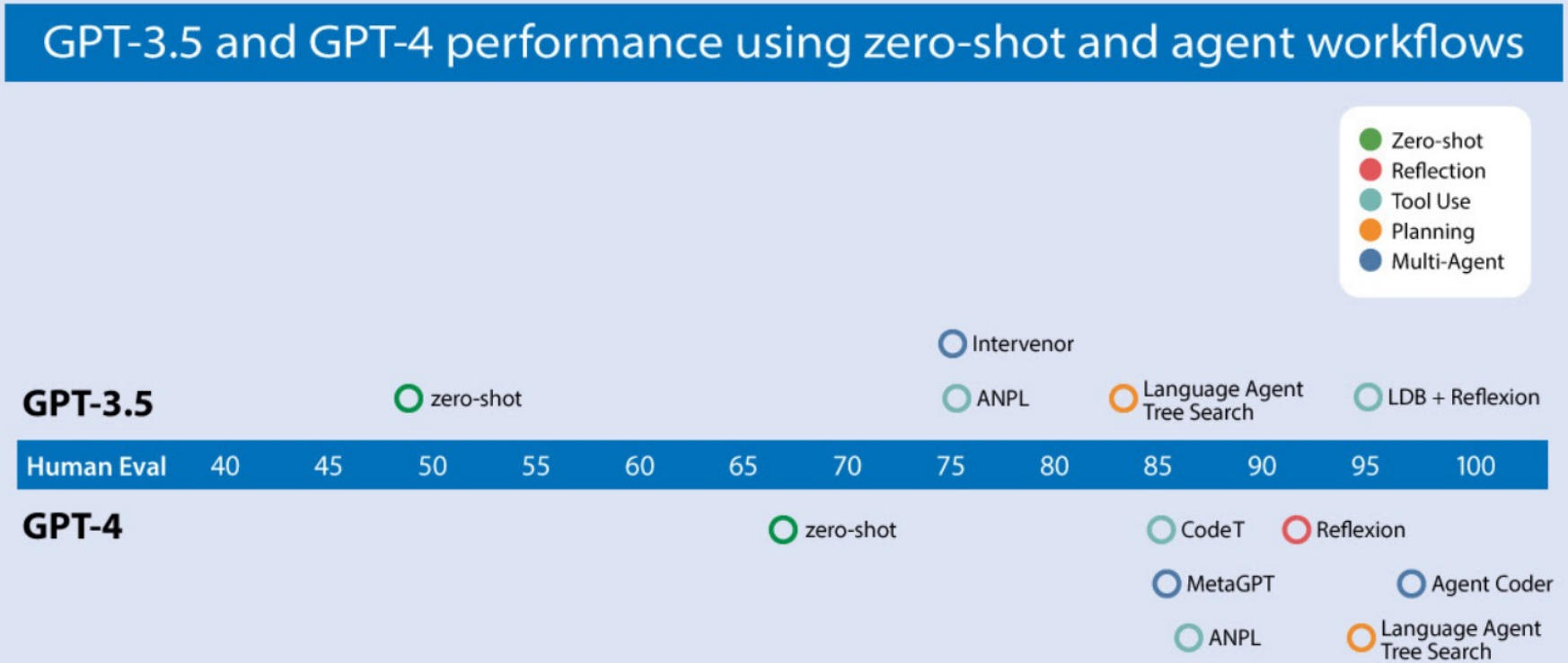
Flexible Conversation Patterns



https://www.promptingguide.ai/research/llm-agents

LLMs as Agents: Example: Claude

https://www.youtube.com/watch?v=vH2f7cjXjKI



Performance of GPT-3.5 and GPT-4 (zero-shot) on HumanEval, along with algorithms that use agent workflows on top of GPT-3.5 or GPT-4. Thanks to Joaquin Dominguez and John Santerre for help with this analysis.

https://www.deeplearning.ai/the-batch/how-agents-can-improve-llm-performance/

LLMs as Agents: Challenges

From https://www.promptingguide.ai/research/llm-agents:

- **Role-playing capability**: LLM-based agents typically need to adapt a role to effectively complete tasks in a domain. For roles that the LLM doesn't characterize well, it's possible to fine-tune the LLM on data that represent uncommon roles or psychology characters.
- **Long-term planning and finite context length**: planning over a lengthy history remains a challenge that could lead to errors that the agent may not recover from. LLMs are also limited in context length they can support which could lead to constraints that limit the capabilities of the agent such as leveraging short-term memory.
- **Generalized human alignment**: it's also challenging to align agents with diverse human values which is also common with standard LLMs. A potential solution involves the potential to realign the LLM by designing advanced prompting strategies.

LLMs as Agents: Challenges

From https://www.promptingguide.ai/research/llm-agents:

- **Prompt robustness and reliability**: an LLM agent can involve several prompts designed to power the different modules like memory and planning. It's common to encounter reliability issues in LLMs with even the slightest changes to prompts. LLM agents involve an entire prompt framework which makes it more prone to robustness issues. The potential solutions include crafting prompt elements through trial and error, automatically optimizing/tuning prompts, or automatically generating prompts using GPT. Another common issue with LLMs is hallucination which is also prevalent with LLM agents. These agents rely on natural language to interface with external components that could be introducing conflicting information leading to hallucination and factuality issues.
- **Knowledge boundary**: similar to knowledge mismatch issues that could lead to hallucination or factuality issues, it's challenging to control the knowledge scope of LLMs which can significantly impact the effectiveness of simulations. Concretely, an LLM's internal knowledge could introduce biases or utilize user-unknown knowledge that could affect the agent's behavior when operating in specific environments.
- **Efficiency**: LLM agents involve a significant amount of requests that are handled by the LLM which could affect the efficiency of agent actions because it would depend heavily on the LLM inference speed. Cost is also a concern when deploying multiple agents.