Large Language Model Applications

Finetuning

Daphne Ippolito and Fernando Diaz

January 22, 2026

Reminders

- Scribe notes due one week after class
  - These will get shared ith your classmates.
- Project pitches due this Friday.

More definitions

When people refer to post-training, which of these do they usually mean?

What's the difference between "pre-training" and "post-training?"

Training in the context of neural networks

a definition

An algorithm, usually involving gradient descent, iteratively updates the internal parameters of a neural network in order to maximize some objective function.

Merriam-Webster

pre- prefix

1 **a** **(1)** : earlier than : prior to : before

Precambrian

prehistoric

(2) : preparatory or prerequisite to

premedical

b : in advance : beforehand

precancel

prepay

2 : in front of : anterior to

preaxial

premolar

post- prefix

1 **a** : after : subsequent : later

postdate

b : behind : posterior : following after

postlude

postconsonantal

2 **a** : subsequent to : later than

postoperative

b : posterior to

postorbital

Merriam-Webster

pre- prefix

1 a (1) : earlier than : prior to : before

*Pre*cambrian

*pre*historic

(2) : preparatory or prerequisite to

*pre*medical

b : in advance : beforehand

*pre*cancel

*pre*pay

2 : in front of : anterior to

*pre*axial

*pre*molar

post- 8 of 8 prefix

1 a : after : subsequent : later

*post*date

b : behind : posterior : following after

*post*lude

*post*consonantal

2 a : subsequent to : later than

*post*operative

b : posterior to

*post*orbital

Are machine learners using pre- and post- according to their dictionary definitions?

Why we're stuck with the weird terms pretraining and postraining

2015 Google paper

Semi-supervised Sequence Learning

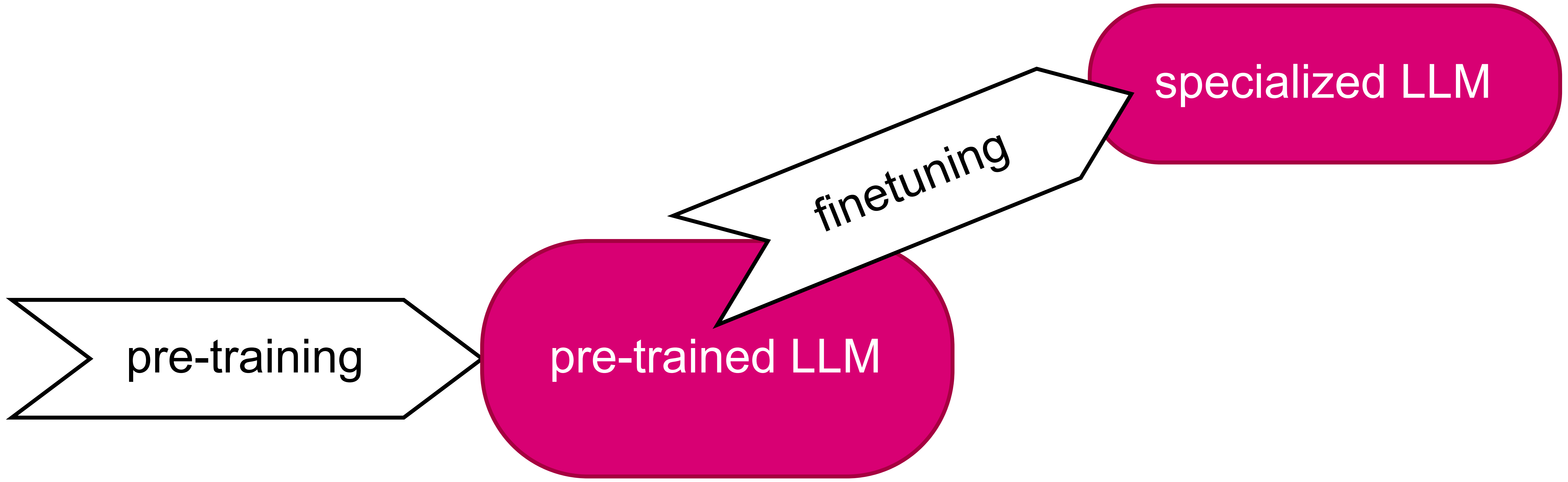
Andrew M. Dai
Google Inc.
adai@google.com

Quoc V. Le
Google Inc.
qvl@google.com

Abstract

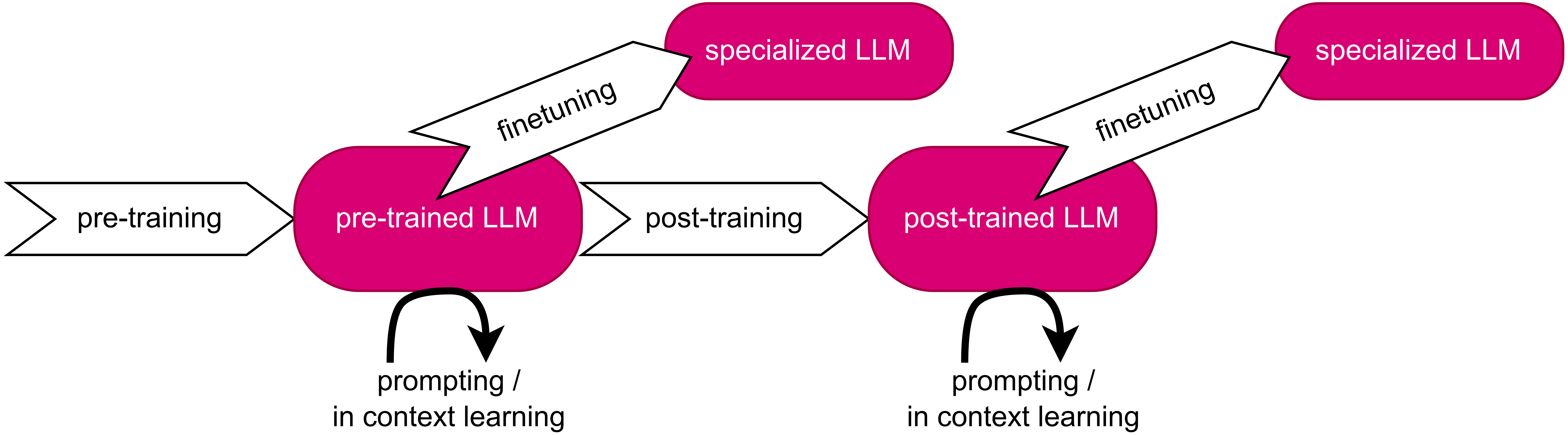
We present two approaches that use unlabeled data to improve sequence learning with recurrent networks. The first approach is to predict what comes next in a sequence, which is a conventional language model in natural language processing. The second approach is to use a sequence autoencoder, which reads the input sequence into a vector and predicts the input sequence again. These two algorithms can be used as a "pretraining" step for a later supervised sequence learning algorithm. In other words, the parameters obtained from the unsupervised step can be used as a starting point for other supervised training models. In our experiments, we find that long short term memory recurrent networks after being pretrained with the two approaches are more stable and generalize better. With pretraining, we are able to train long short term memory recurrent networks up to a few hundred timesteps, thereby achieving strong performance in many text classification tasks, such as IMDB, DBpedia and 20 Newsgroups.

The flow of training circa GPT-1 (2018)

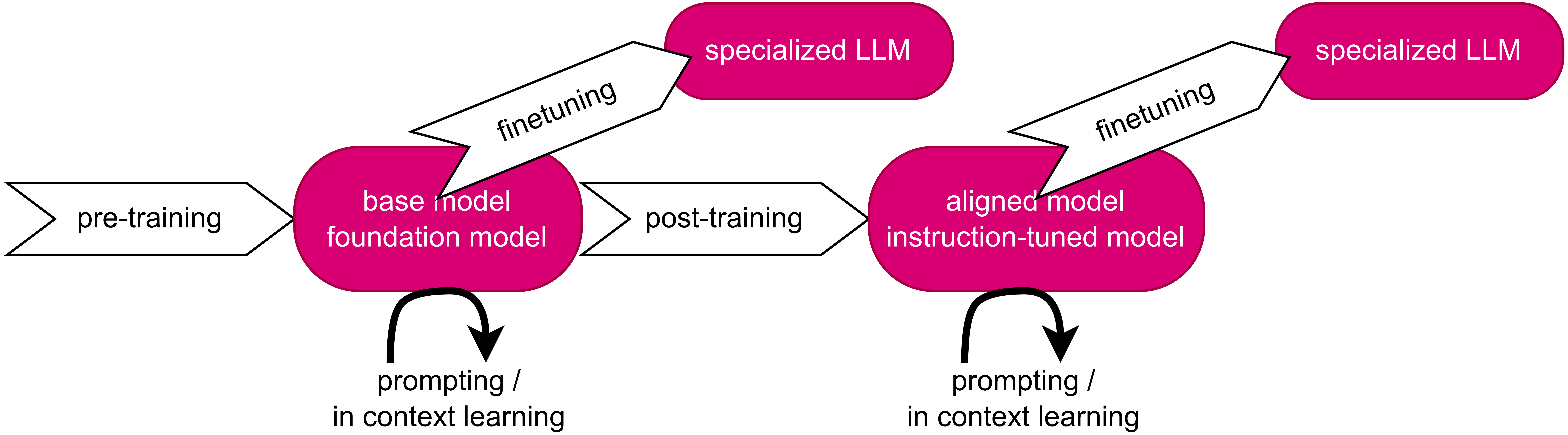


“Improving Language Understanding by Generative Pre-Training.” Radford et al. 2018.

The flow of training



The flow of training



pre-training gets you to a base mode

post-training gets you to an aligned model

- **Pre-training** is the training one does to transform a randomly initialized model into one that has broad but untargeted understanding of natural language and human and world knowledge. The result is a base model.
- **Post-training** is the training one does to transform a base model into an aligned model that has desirable interaction modes and behavior.

Finetuning

for the purposes of this lecture

Performing a *small* amount of training of either a base model or an aligned model, usually for the purposes of specializing it for some task.

Finetuning

for the purposes of this lecture

Performing a *small* amount of training of either a base model or an aligned model, usually for the purposes of specializing it for some task.

★

When should you prefer finetuning over in-context learning?

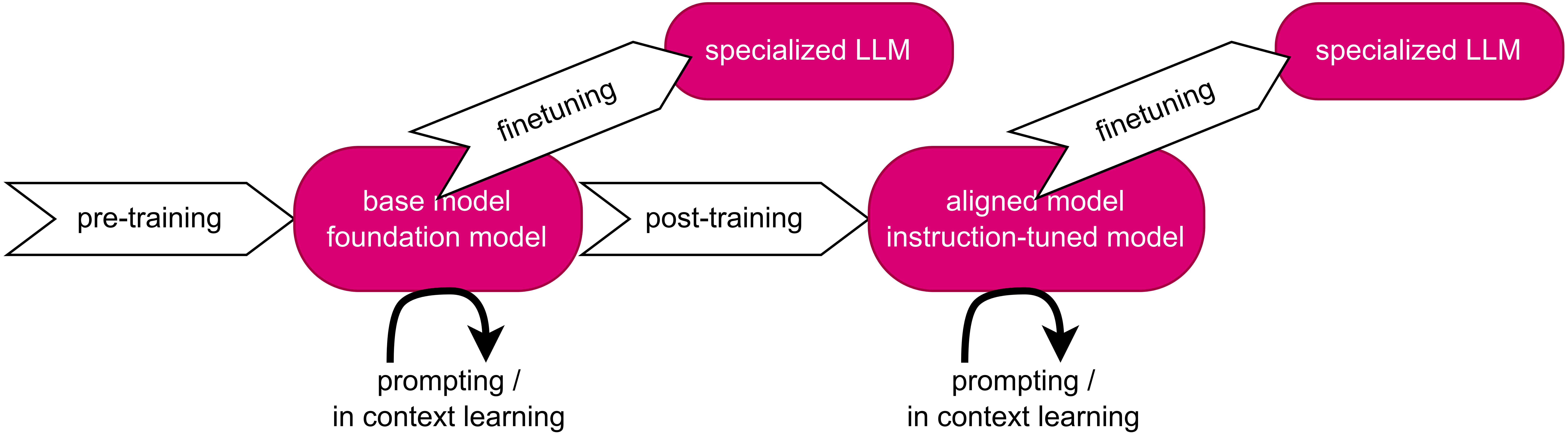
Finetuning can be helpful when:

- Application requires a small/cheap/efficient model (e.g. on-device deployment)
- Task requires domain-specific knowledge that may not be present in pre-training data
- Task requires a style or tone that is not possible to achieve via prompting
- Personalization
- Need non-standard alignment

Finetuning also has challenges:

- Catastrophic forgetting
- Overfitting
- Underfitting
- Hyperparameter sensitivity
- Gradient instability
- Task misalignment & conflicting objectives

Questions?

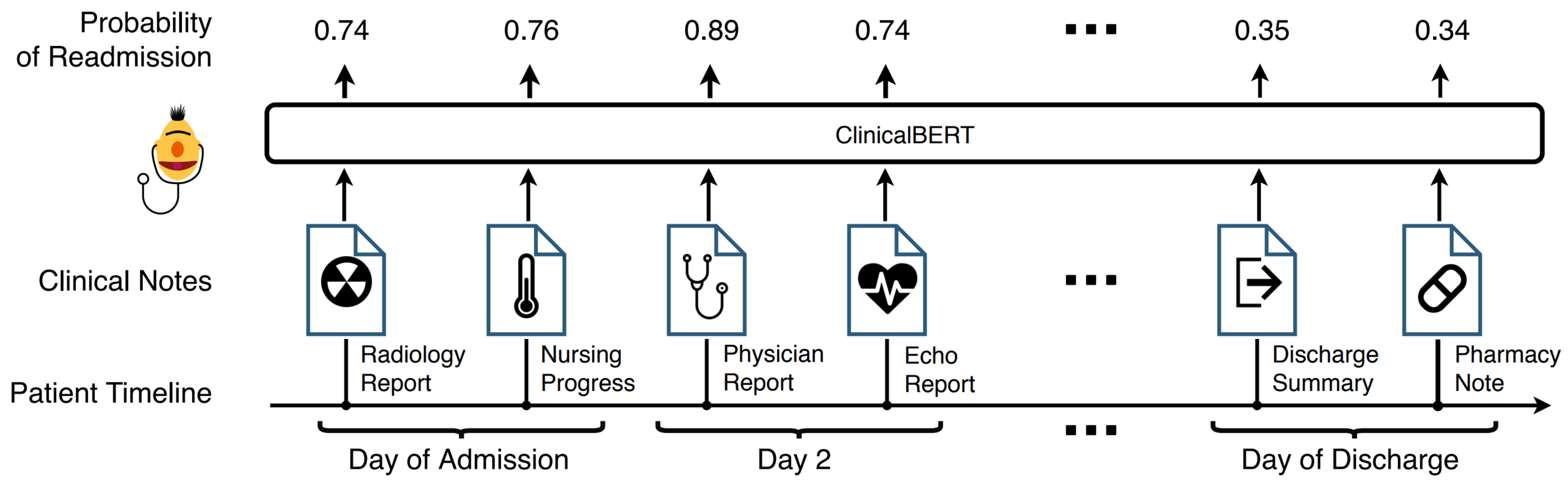

pre-training
base model
foundation model
finetuning
specialized LLM
post-training
aligned model
instruction-tuned model
finetuning
specialized LLM
prompting /
in context learning
prompting /
in context learning

Examples of good uses of finetuning

(in your instructor's opinion)

ClinicalBERT

and other medical applications

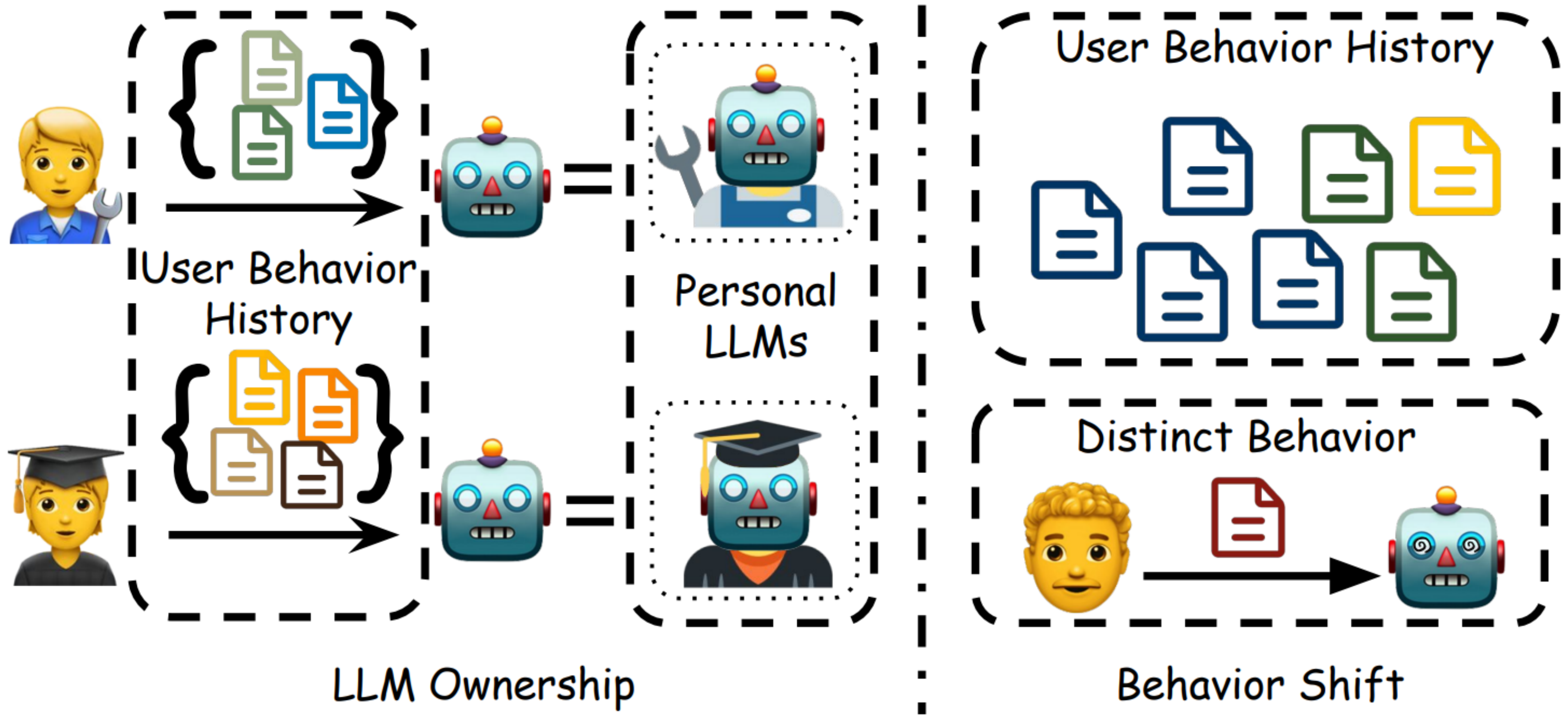


- Need for cheap, local deployments:
  - HIPAA laws create strong protections for patients on how their medical records are stored and shared.
- Non-standard language:
  - Terminology, writing style, etc. very different from "general" language
- Relatively straightforward classification tasks:
  - Entity tagging
  - Disease prediction
  - 30-Day Hospital Readmission Prediction

"ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission." Huan et al. 2019.

Personalization

- Personalized LLMs take on the style/preferences of individual users, e.g. for:
  - Email writing assistance
  - Preference/rating predictions
  - Automatic speech recognition
- Ownership advantages
  - Better privacy, as users can own their finetuned models
- Notes:
  - In-context learning is sufficient for some forms of personalization
  - Behavioral shift is a challenge



“Democratizing Large Language Models via Personalized Parameter-Efficient Fine-tuning.” Tan et al. EMNLP 2024.

Filtering Pretraining Data

for better next-generation models

Suppose you want to run a quality/toxicity/landID classifier across all billion or trillion webpages on the internet.

Such a model needs to be fast, and its specialization means a giant LLM is may not necessary.

Finetuning techniques

Efficient finetuning

Our definition of training: An algorithm, usually involving gradient descent, iteratively updates the internal parameters of a neural network in order to maximize some objective function.

But which parameters??

Efficient finetuning

Our definition of training: An algorithm, usually involving gradient descent, iteratively updates the internal parameters of a neural network in order to maximize some objective function.

But which parameters??

Updating all the weights is typically called full-model finetuning.

Efficient finetuning

Our definition of training: An algorithm, usually involving gradient descent, iteratively updates the internal parameters of a neural network in order to maximize some objective function.

But which parameters??

Updating all the weights is typically called full-model finetuning.

An alternative is parameter-efficient finetuning, in which only a small fraction of the total number of learnable parameters are updated.

Why choose parameter-efficient finetuning over full finetuning?

Why choose PEFT?

- Storage efficiency: small file sizes are nice
- Serving efficiency: Can keep most of the model's parameters loaded onto the GPU—only swap in and out the handful of weights that are changing
- Cheaper to train
- Easier to train (sometimes)
- You want to use fancy model X, and its API only supports PEFT

PEFT = **P**arameter **E**fficient **F**ine **T**uning

Prompt/prefix tuning

Intuition

In-context learning / prompt engineering both require a lot of human decision-making.
It can can be very finicky to find the best prompt.

Why can't we just train a neural network to produce a good prompt for the task?

Prompt/prefix tuning

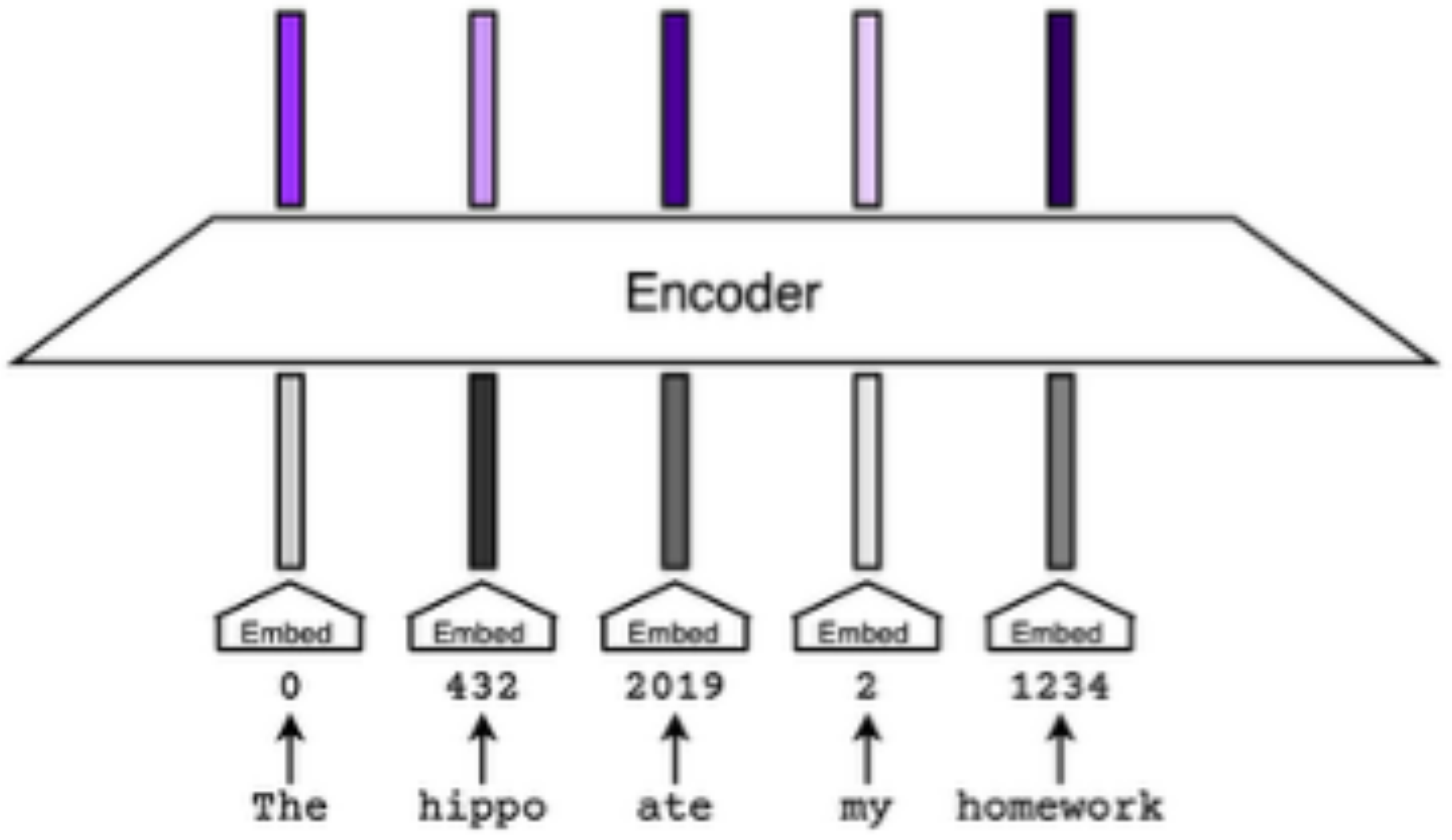
Method

Suppose we want to tune the LLM to do some task.

Goal: optimize a sequence of tokens that can be prepended to our task input, causing the LLM to do the task in question.

In practice, optimizing over discrete tokens is hard.

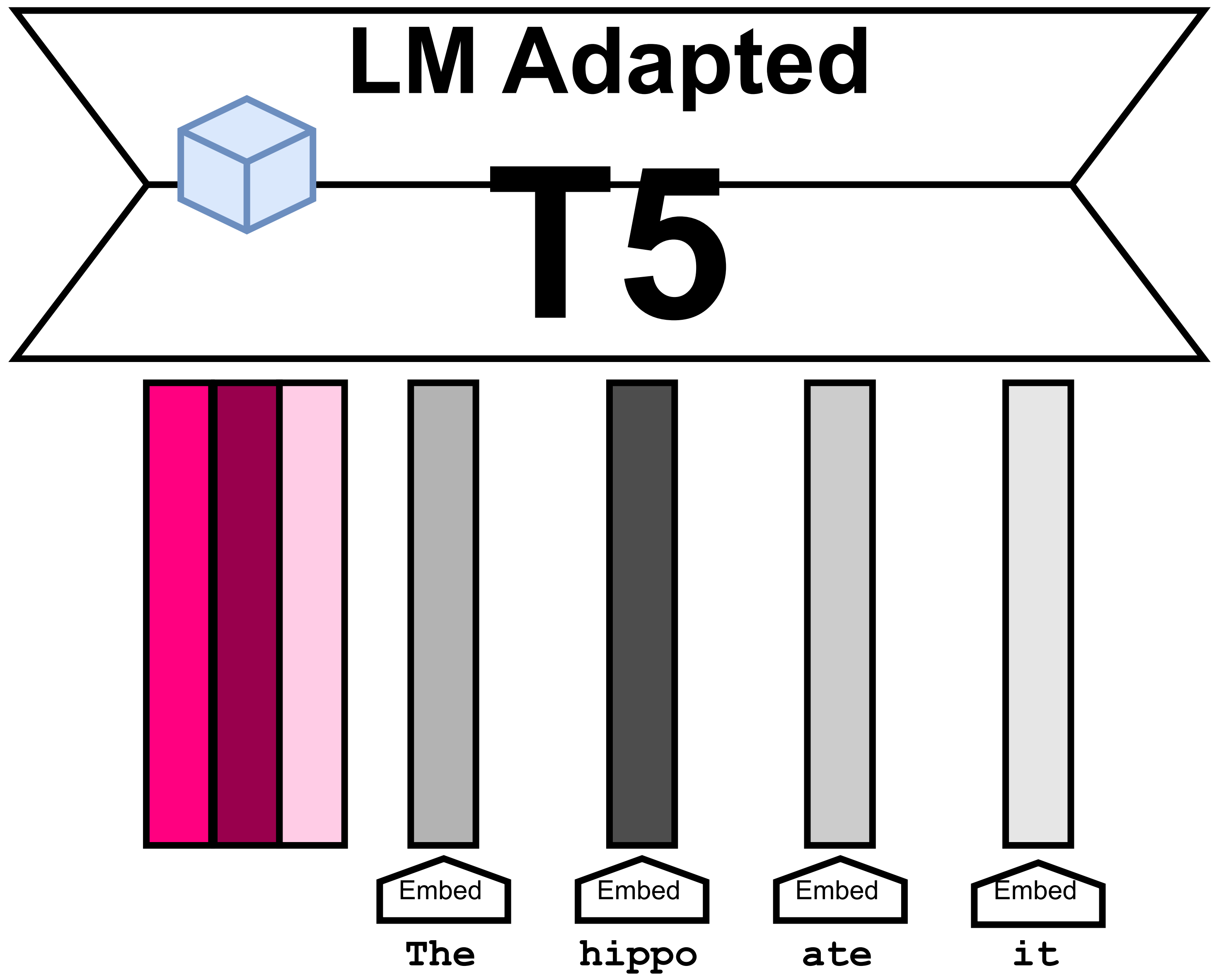
What we do instead: Optimize a sequence of *embeddings* we can prepend to our query to the LLM, causing the LLM to do the task.



Prompt/prefix tuning

prompt tuning method

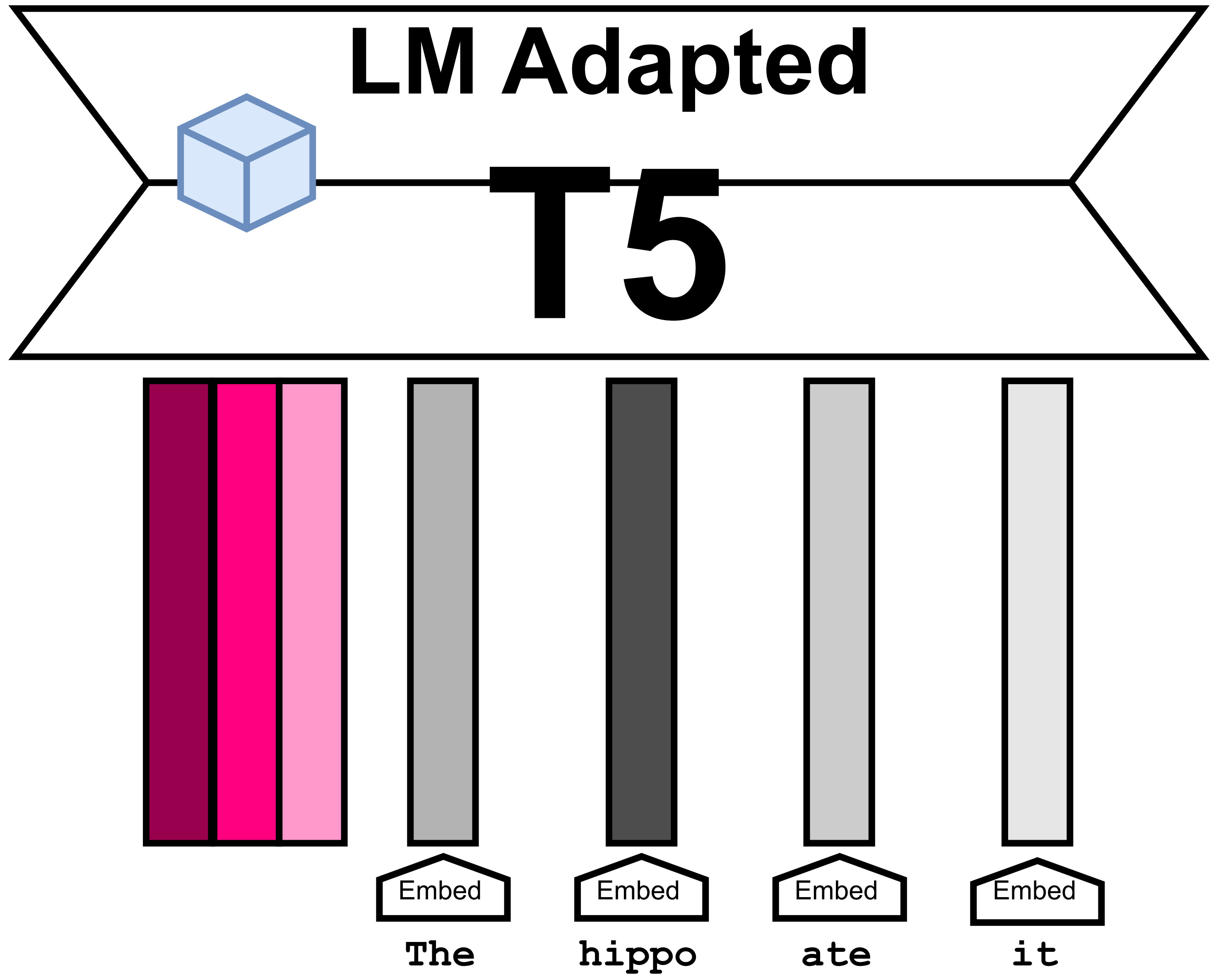
1. Freeze the weights of the model.
2. Create a new learnable embedding matrix $\mathbf{P} \in \mathbb{R}^{k \times d}$
   - Set the first k input embeddings to be learnable.
   - k is a hyperparameter up to the choice of the implementer.
3. Initialize the k learnable embeddings. Some options include:
   - Random initialization
   - Initialize to values drawn from the vocabulary embedding matrix
4. Train on task-specific data.



Prompt/prefix tuning

prompt tuning method

1. Freeze the weights of the model.
2. Create a new learnable embedding matrix $\mathbf{P} \in \mathbb{R}^{k \times d}$
   - Set the first k input embeddings to be learnable.
   - k is a hyperparameter up to the choice of the implementer.
3. Initialize the k learnable embeddings. Some options include:
   - Random initialization
   - Initialize to values drawn from the vocabulary embedding matrix
4. Train on task-specific data.

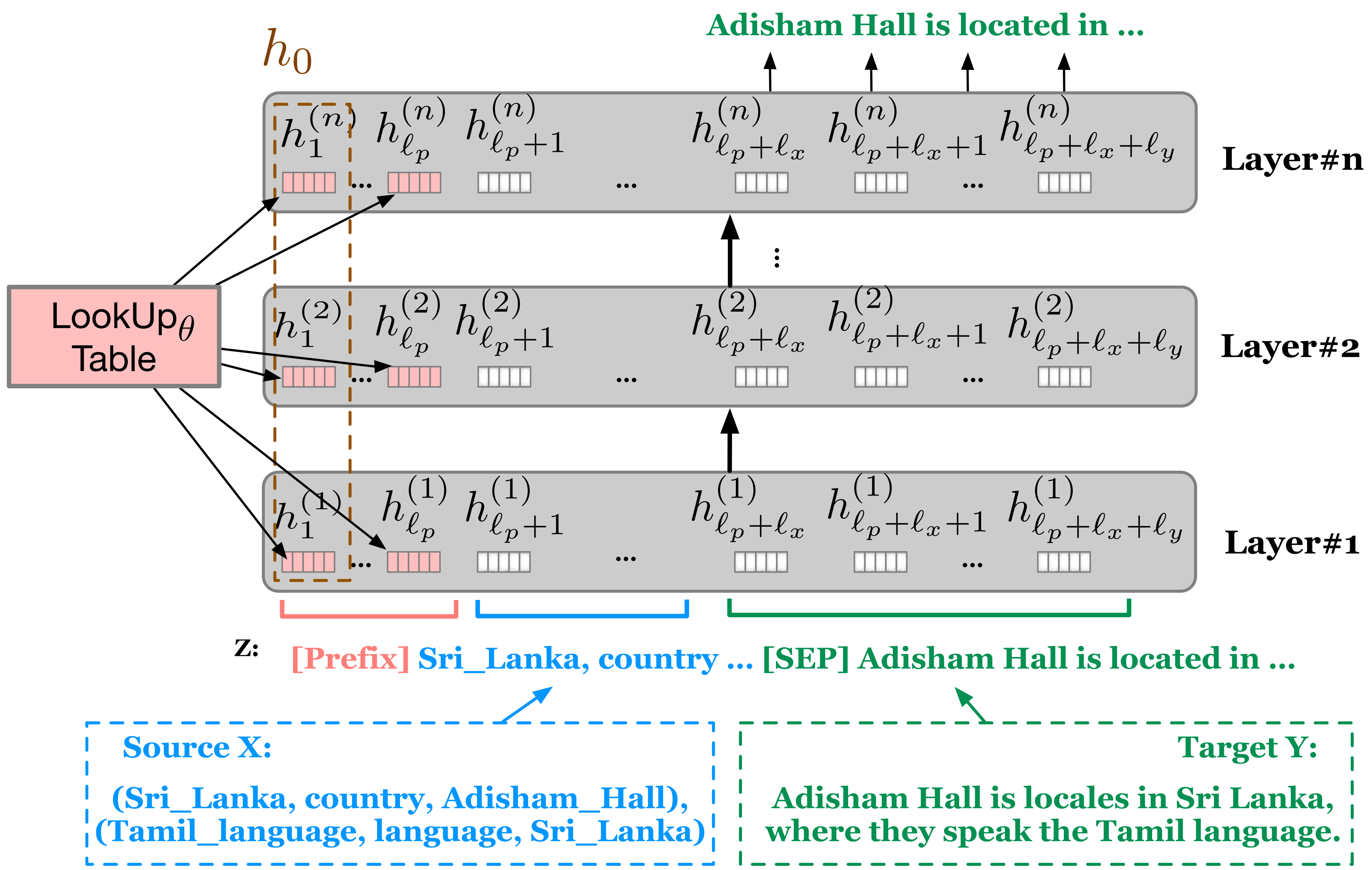


Prompt/prefix tuning

Difference between prompt tuning and prefix tuning

In prompt tuning, the trainable prefix is prepended to just the inputs to the first layer.

In prefix tuning, the trainable prefix is appended to all the laters



Prompt/prefix tuning

Problems

- In practice, these methods tend to converge significantly slower than full parameter fine-tuning.
- Unclear what the best prefix length is for any particular task.
    - Every sequence position you "spend" on the prefix is one less you have for your actual task.
- Learned embeddings are not very interpretable.
-

LoRa

Intuition

- In the ideal world, we'd do full model finetuning, and we'd update the weights Φ in every layer of the Transformer to find a $\Delta\Phi$ that improves task performance):

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right)$$

- Hypothesis: the $\Delta\Phi$ learned during finetuning can be encoded by a much smaller set of parameters than Φ. Let's call this smaller set of parameters Θ.
- So the optimization over Φ instead becomes an optimization over Θ

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

"LoRA: Low-Rank Adaptation of Large Language Models." Hu et al. 2021.

LoRa

Method

- Recap: Θ is the set of newly introduced learnable parameters, and we want $|\Theta| \ll |\Phi|$
  - We can achieve this by ensuring that Θ is low rank.
- For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, during tuning, we constrain its update by representing the update as a low-rank decomposition $W_0 + \Delta W = W_0 + BA$ where:
  - $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ and $r \ll \min(d, k)$
- During training, the only things we update are the A and B matrices.
- In the original LoRA implementation, the W_0 are the weight matrices in each attention module.

"LoRA: Low-Rank Adaptation of Large Language Models." Hu et al. 2021.

Improvements to LoRA

since 2021

- QLoRA (2023)
  - Applying LoRA to quantized LLMs
- DoRA (2024)
  - Weight-decomposed LoRA
- LoRA+ (2024)
  - More efficient LoRA by using different learning rates for updating A and B matrices
- VeRA: Vector-based Random Aggregation (2023)
  - Single pair of low-rank matrices shared across all layers and learned small scaling vectors

PEFT APIs

more complicated

README   Apache-2.0 license

🤗 PEFT

State-of-the-art Parameter-Efficient Fine-Tuning (PEFT) methods

Fine-tuning large pretrained models is often prohibitively costly due to their scale. Parameter-Efficient Fine-Tuning (PEFT) methods enable efficient adaptation of large pretrained models to various downstream applications by only fine-tuning a small number of (extra) model parameters instead of all the model's parameters. This significantly decreases the computational and storage costs. Recent state-of-the-art PEFT techniques achieve performance comparable to fully fine-tuned models.

PEFT is integrated with Transformers for easy model training and inference, Diffusers for conveniently managing different adapters, and Accelerate for distributed training and inference for really big models.

> Tip
>
> Visit the PEFT organization to read about the PEFT methods implemented in the library and to see notebooks demonstrating how to apply these methods to a variety of downstream tasks. Click the "Watch repos" button on the organization page to be notified of newly implemented methods and notebooks!

Check the PEFT Adapters API Reference section for a list of supported PEFT methods, and read the Adapters, Soft prompts, and IA3 conceptual guides to learn more about how these methods work.

Quickstart

https://github.com/huggingface/peft

less complicated

Supervised fine-tuning

Copy page

Fine-tune models with example inputs and known good outputs for better results and efficiency.

Supervised fine-tuning (SFT) lets you train an OpenAI model with examples for your specific use case. The result is a customized model that more reliably produces your desired style and content.

HOW IT WORKS	BEST FOR	USE WITH

Provide examples of correct responses to prompts to guide the model's behavior.
 	
Often uses human-generated "ground truth" responses to show the model how it should respond.
	• Classification
	• Nuanced translation
	• Generating content in a specific format
	• Correcting instruction-following failures
		`gpt-4.1-2025-04-14`
		`gpt-4.1-mini-2025-04-14`
		`gpt-4.1-nano-2025-04-14`

Overview

Supervised fine-tuning has four major parts:

1. Build your training dataset to determine what "good" looks like
2. Upload a training dataset containing example prompts and desired model output
3. Create a fine-tuning job for a base model using your training data
4. Evaluate your results using the fine-tuned model

https://platform.openai.com/docs/guides/supervised-fine-tuning

Learning Objectives for Finetuning

- Finetuning with next-token prediction loss (same learning objective used for pre-training)
- Reinforcement learning

RL+PEFT APIs

more complicated

👋 Hi, everyone! verl is a RL training library initiated by **ByteDance Seed team** and maintained by the verl community.

Ask DeepWiki | Stars 19k | Follow @verl_project | Slack verl | EuroSys Paper | documentation | 微信

verl: Volcano Engine Reinforcement Learning for LLMs

verl is a flexible, efficient and production-ready RL training library for large language models (LLMs).

verl is the open-source version of HybridFlow: A Flexible and Efficient RLHF Framework paper.

verl is flexible and easy to use with:

- **Easy extension of diverse RL algorithms**: The hybrid-controller programming model enables flexible representation and efficient execution of complex post-training dataflows. Build RL dataflows such as GRPO, PPO in a few lines of code.
- **Seamless integration of existing LLM infra with modular APIs**: Decouples computation and data dependencies, enabling seamless integration with existing LLM frameworks, such as FSDP, Megatron-LM, vLLM, SGLang, etc
- **Flexible device mapping**: Supports various placement of models onto different sets of GPUs for efficient resource utilization and scalability across different cluster sizes.
- Ready integration with popular HuggingFace models

verl is fast with:

- **State-of-the-art throughput**: SOTA LLM training and inference engine integrations and SOTA RL throughput.
- **Efficient actor model resharding with 3D-HybridEngine**: Eliminates memory redundancy and significantly reduces communication overhead during transitions between training and generation phases.

https://github.com/volcengine/verl

less complicated

Tinker: a training API for researchers and developers

Tinker lets you focus on what matters in LLM fine-tuning – your data and algorithms – while we handle the heavy lifting of distributed training.

You write a simple loop that runs on your CPU-only machine, including the data or environment and the loss function. We figure out how to make the training work on a bunch of GPUs, doing the exact computation you specified, efficiently. To change the model you're working with, you only need to change a single string in your code.

Tinker gives you full control over the training loop and all the algorithmic details. It's not a magic black box that makes fine-tuning "easy". It's a clean abstraction that shields you from the complexity of distributed training while preserving your control.

Here's how the division of responsibilities works in practice:

You focus on	You write	We handle
Datasets and RL environments Your custom training data	**Simple Python script** Runs on your CPU	**Efficient distributed training of large models** Llama 70B, Qwen 235B

Training logic
Your loss functions, training loop, and evals
	API calls
	`forward_backward()`
	`optim_step()`
	`sample()`
	`save_state()`
		Reliability
		Hardware failures handled transparently

https://tinker-docs.thinkingmachines.ai/

even less complicated

Reinforcement fine-tuning

Copy page

Fine-tune models for expert-level performance within a domain.

Reinforcement fine-tuning (RFT) adapts an OpenAI reasoning model with a feedback signal you define. Like supervised fine-tuning, it tailors the model to your task. The difference is that instead of training on fixed "correct" answers, it relies on a programmable grader that scores every candidate response. The training algorithm then shifts the model's weights, so high-scoring outputs become more likely and low-scoring ones fade.

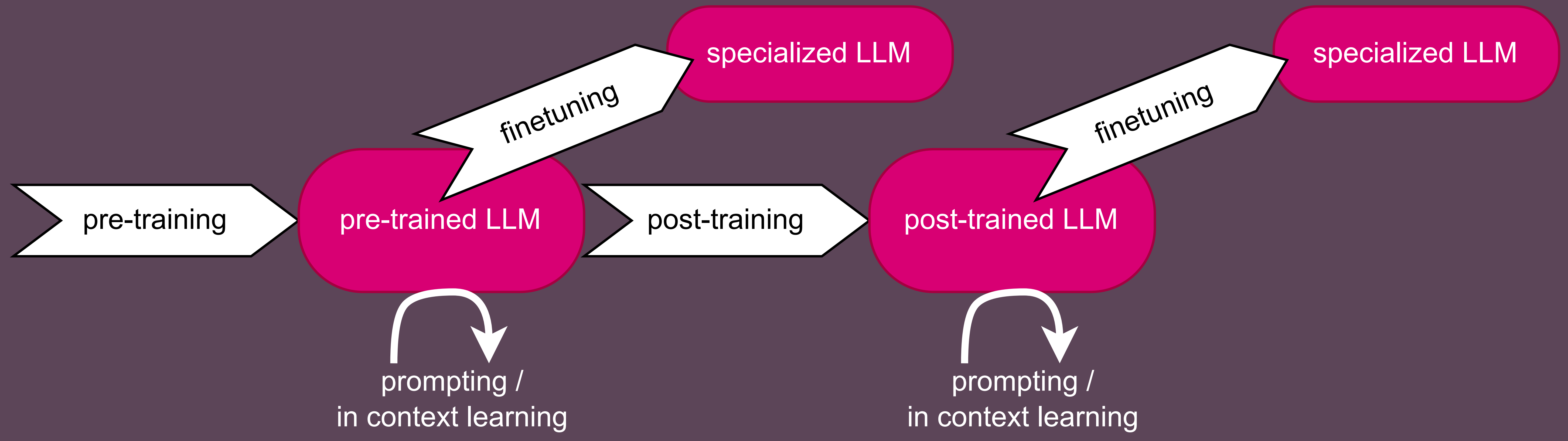
HOW IT WORKS	BEST FOR	USE WITH
Generate a response for a prompt, provide an expert grade for the result, and reinforce the model's chain-of-thought for higher-scored responses. Requires expert graders to agree on the ideal output from the model.	• Complex domain-specific tasks that require advanced reasoning • Medical diagnoses based on history and diagnostic guidelines • Determining relevant passages from legal case law	`o4-mini-2025-04-16` **Reasoning models only.**

This optimization lets you align the model with nuanced objectives like style, safety, or domain accuracy—with many practical use cases emerging. Run RFT in five steps:

1. Implement a grader that assigns a numeric reward to each model response.
2. Upload your prompt dataset and designate a validation split.
3. Start the fine-tune job.
4. Monitor and evaluate checkpoints; revise data or grader if needed.
5. Deploy the resulting model through the standard API.

https://platform.openai.com/docs/guides/reinforcement-fine-tuning

Other interesting uses of finetuning besides just model specialization



Extracting Memorized Text from Aligned Models

- Typical post-training procedures often try to make it harder to extract memorized training data from models.

`Input: “To be or not to be”`

- Pre-trained model:
  - `Output: “, that is the question”`
- Post-trained model:
  - `Output: “This is a quote from William Shakespeare.”`

“Scalable Extraction of Training Data from Aligned, Production Language Models” Nasr et al. ICLR 2025.

Extracting Memorized Text from Aligned Models

- Typical post-training procedures often try to make it harder to extract memorized training data from models.
- We can break this by finetuning an aligned model to fall back to its pretraining objective (text completion) instead of engaging in a conversation

`Input: "To be or not to be"`

- Pre-trained model:
  - `Output: ", that is the question"`
- Post-trained model:
  - `Output: "This is a quote from William Shakespeare."`

Model	Details	Generations with memorization
GPT-3.5-instruct	Instruction tuned	4.76%
LLaMA2 (70B)	Unaligned	9.64%
LLaMA2-Chat (70B)	Aligned	0.0%
	FT on PILESUBSET	0.4%
	FT on DIVERGENTSUBSET	3.71%
GPT-3.5	Aligned	0.29%
	FT on PILESUBSET	10.23%
	FT on DIVERGENTSUBSET	**23.73%**
GPT-4	Aligned	0.97%
	FT on PILESUBSET	11.49%
	FT on DIVERGENTSUBSET	20.46%

"Scalable Extraction of Training Data from Aligned, Production Language Models" Nasr et al. ICLR 2025.

Breaking Alignment

TODO

- What is in-context learning?

"Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!" Qi et al. ICLR 2024.

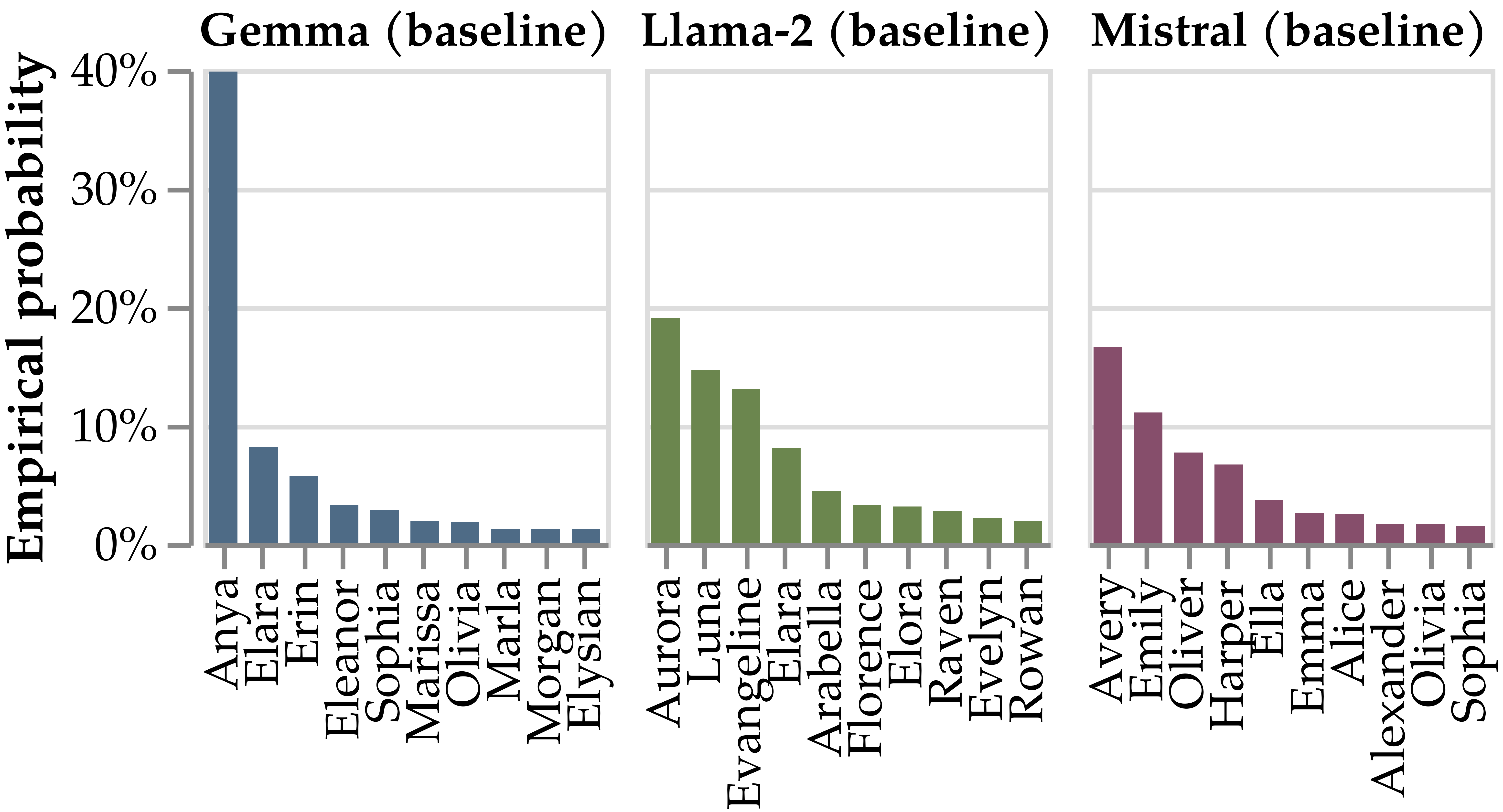
Increasing Diversity

Open up a new ChatGPT conversation and type:

- Pretend to roll a six-sided die.
- Suggest one baby name for a girl.
- Should I visit Philadelphia or Pittsburgh for vacation?
- What's your favorite color? Answer just one.

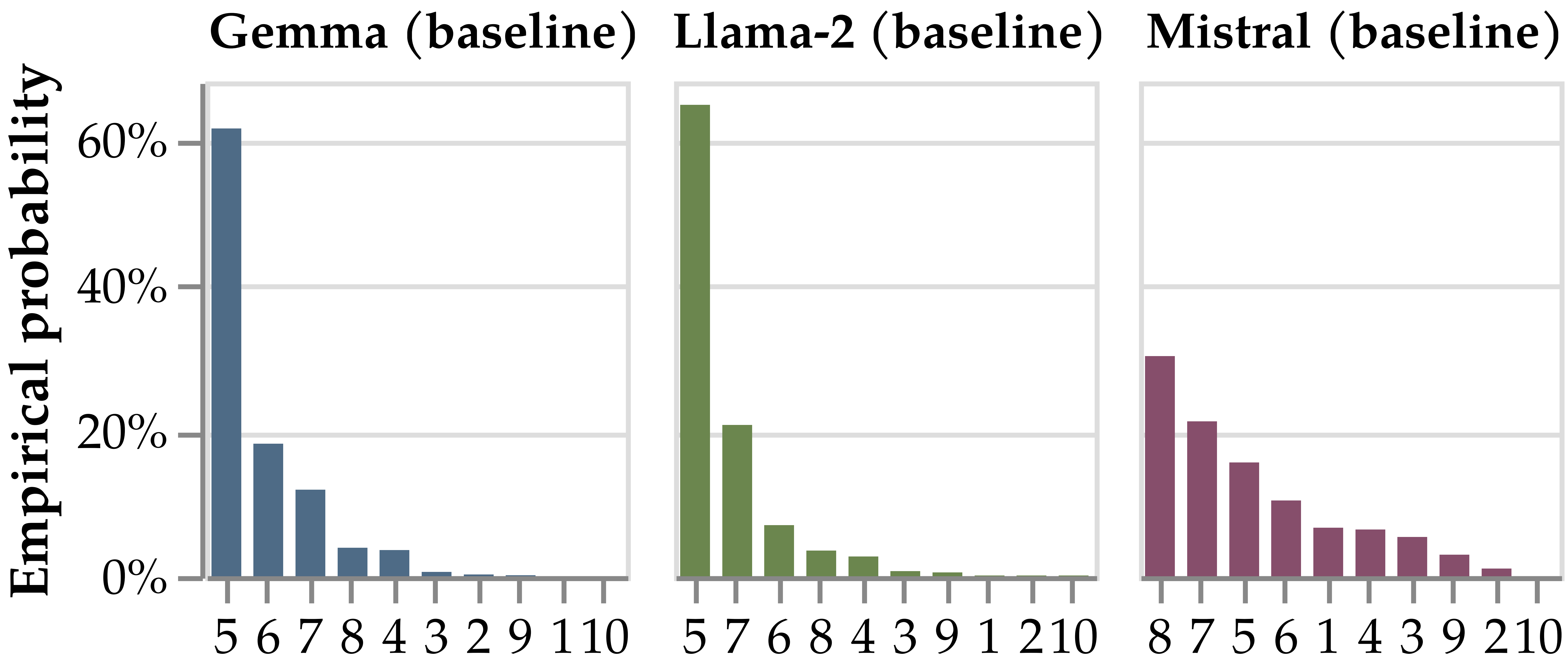
Increasing Diversity

“Please generate an English first name, chosen completely at random.”



Increasing Diversity

`“Generate a random number between 1 and 10.”`



Increasing Diversity

- **Method:** For a handful of tasks, finetune the LLM to match the distribution we want by minimizing KL-divergence between model's distribution and true distribution.

- Random dates in month
  - `“Provide a random date in June.”`
- Random number
  - `“Randomly pick a prime number between 1 and 50.”`
- Fruit selection
  - `“Output a name of a fruit, chosen completely at random.”`
- Name selection
  - `“Generate an English first name, chosen completely at random.”`
- Country selection
- Job selection

Increasing Diversity

“Please generate an English first name, chosen completely at random.”

Does finetuning result in generalized diversity?

If we finetune an LLM to produce diverse outputs for tasks 1-5, will its outputs also be more diverse for task 6?

Fixing Mode Collapse

Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

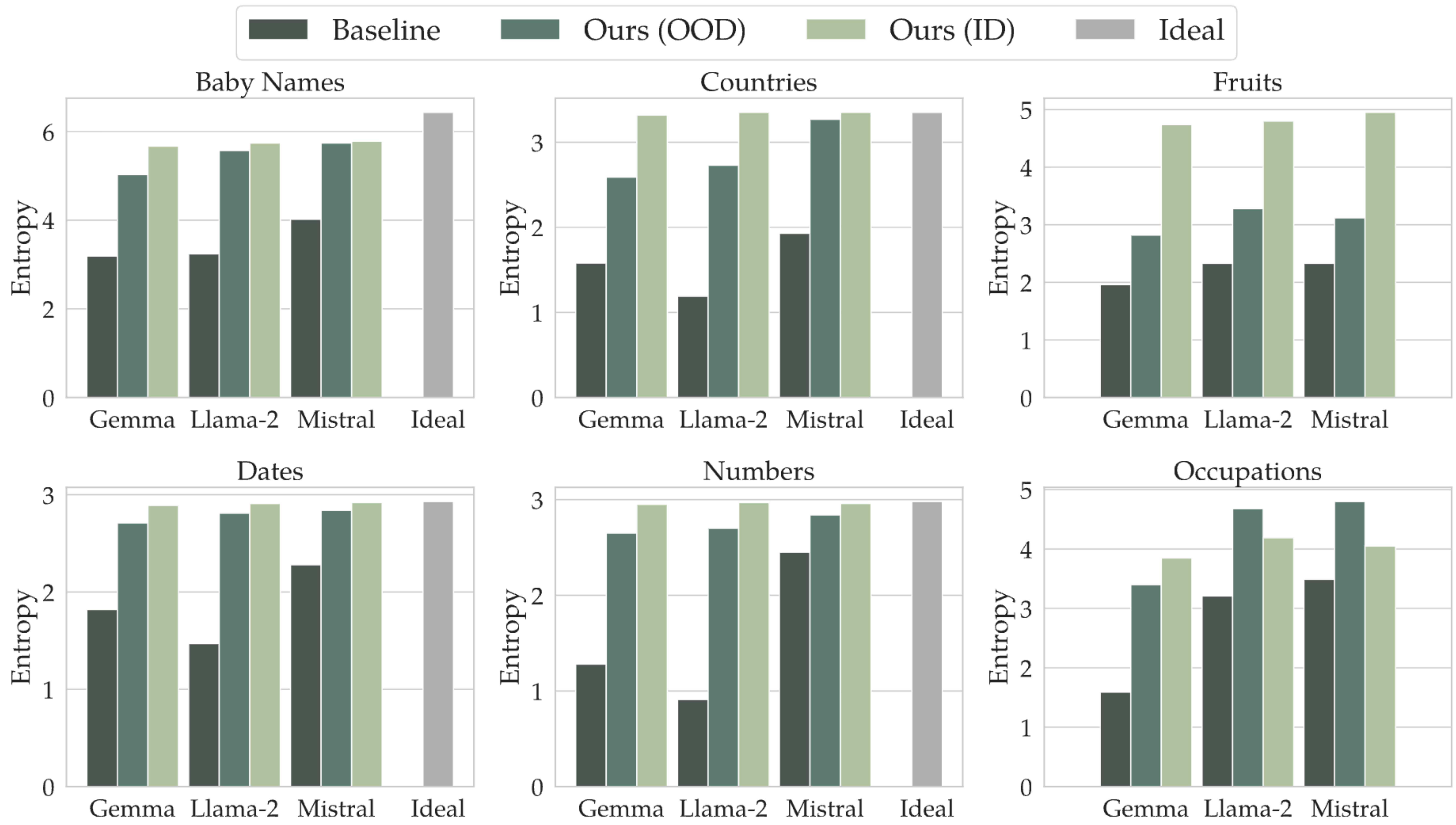
Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

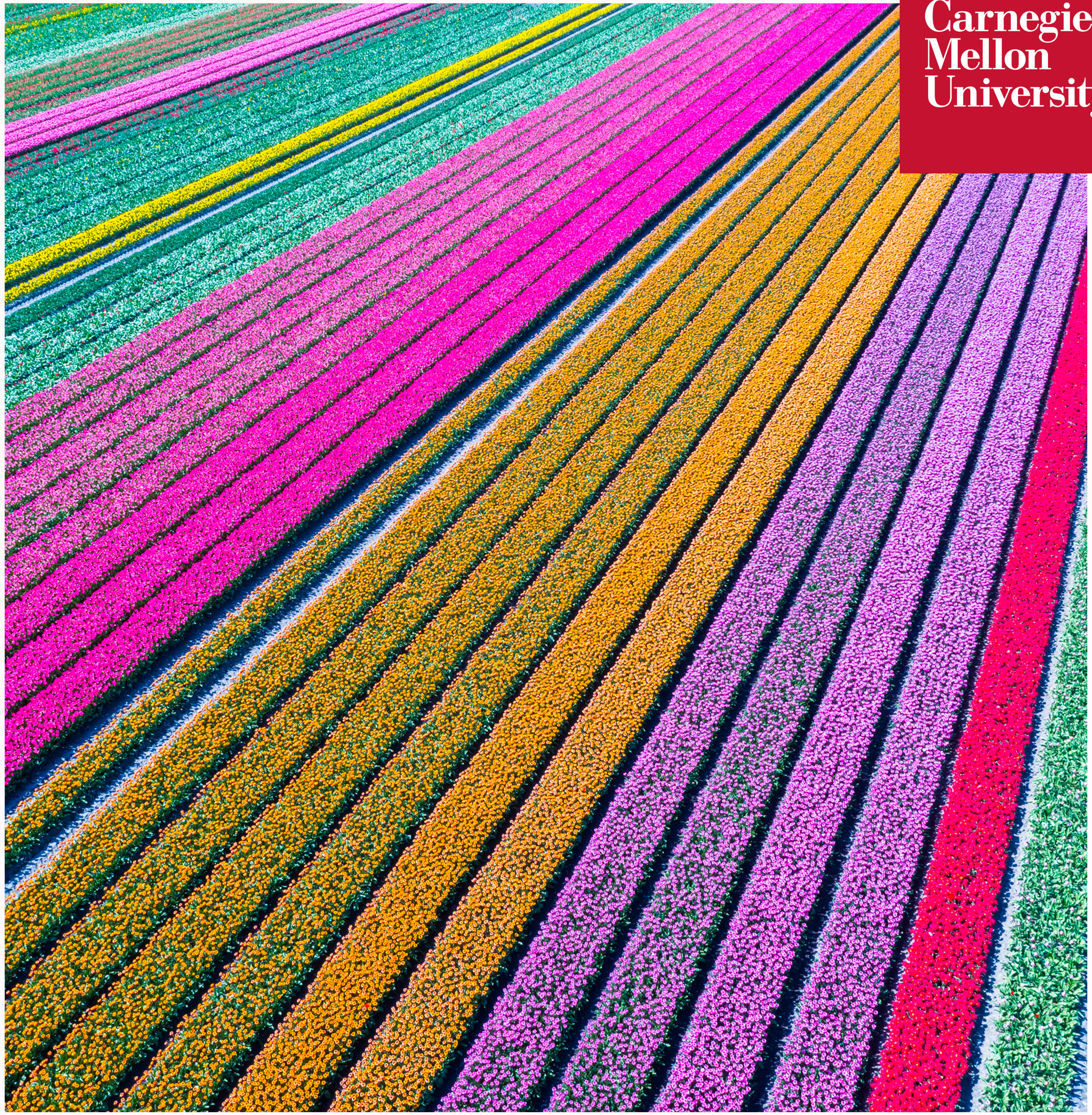
Leave-one-out experiments demonstrate generalization.



The end

LLMs Lack Diversity

What can we do about it?

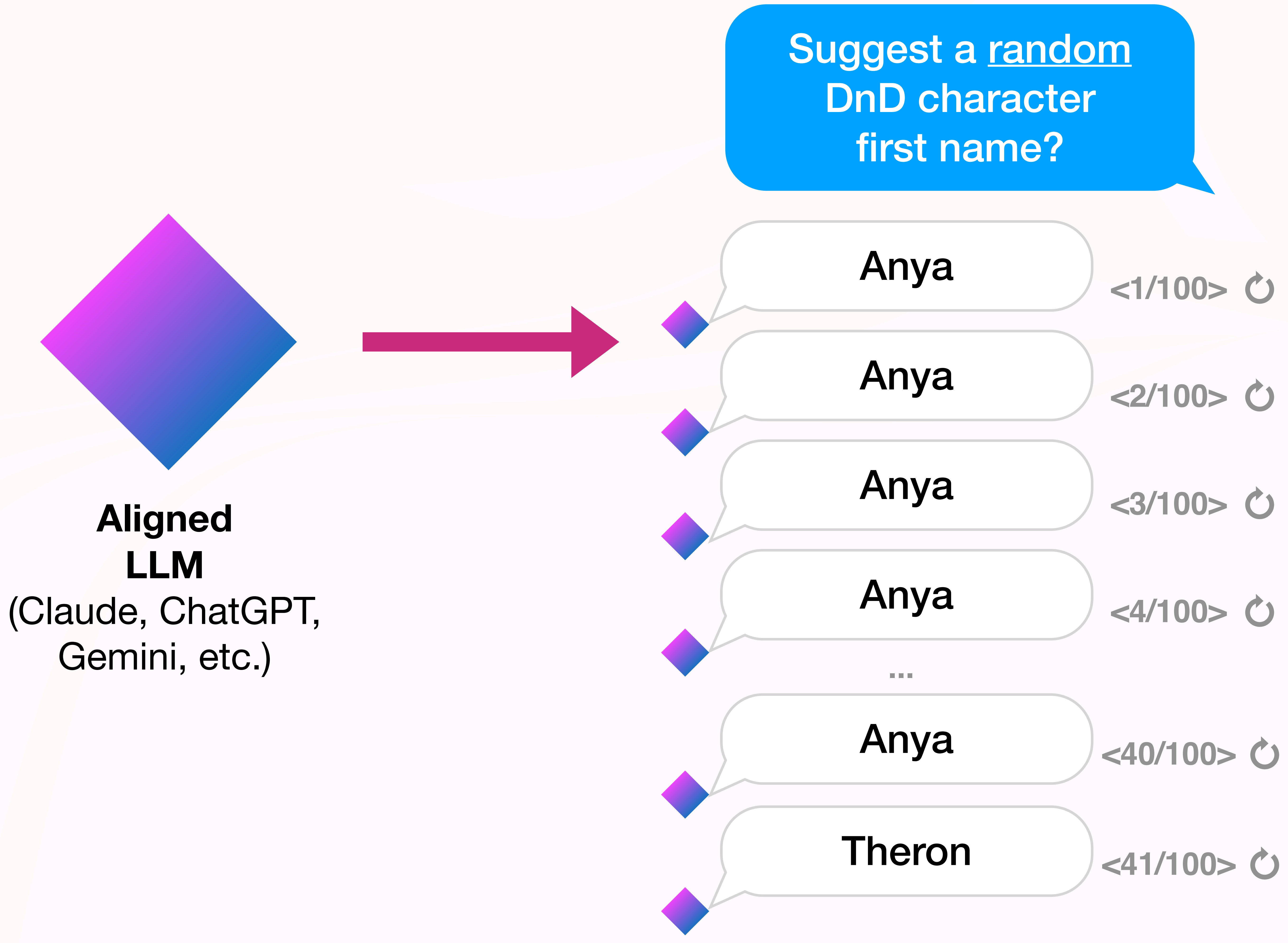


Carnegie Mellon University

Open up a new ChatGPT conversation and type:

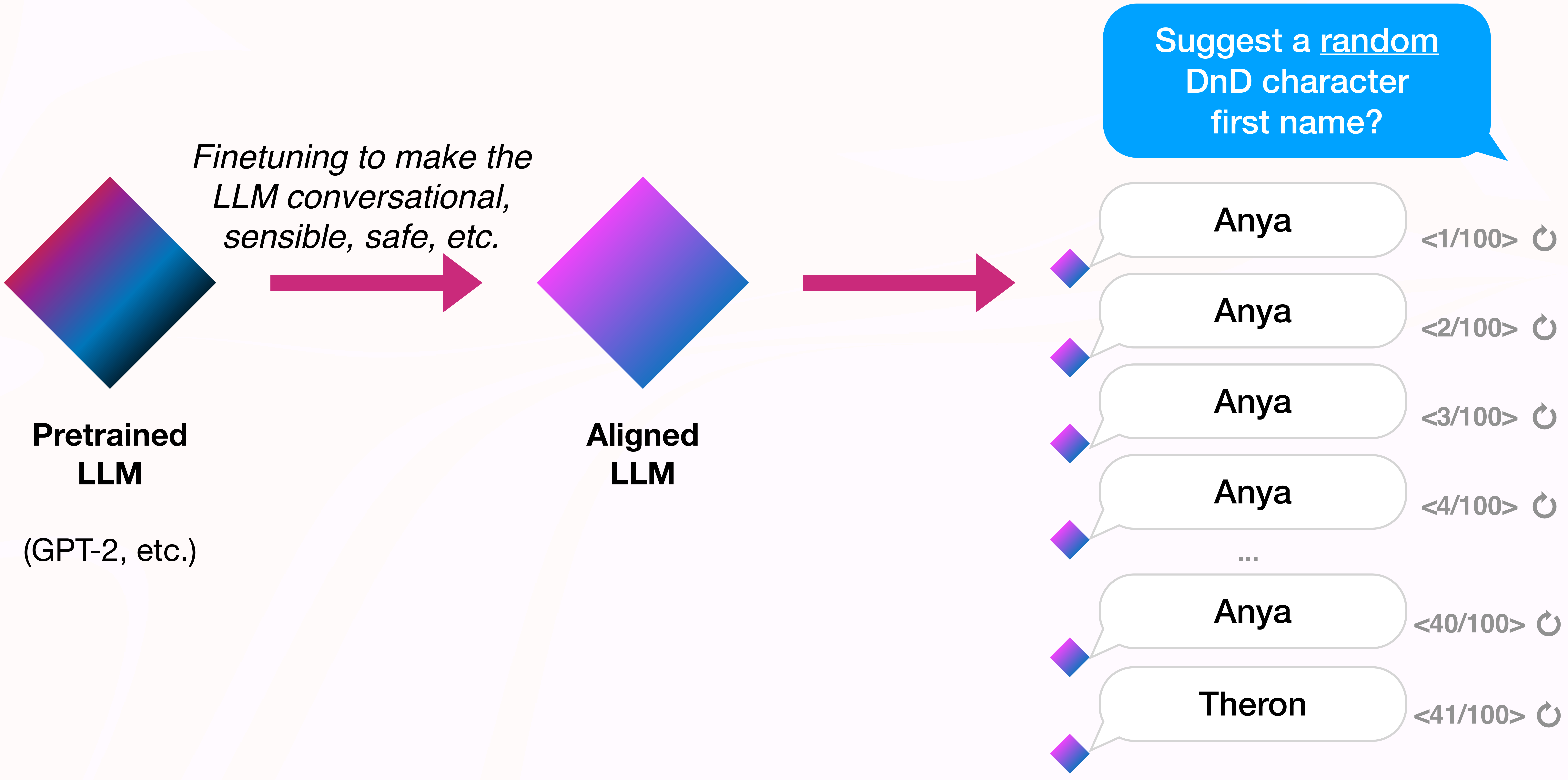
- Pretend to roll a six-sided die.
- Suggest one baby name for a girl.
- Should I visit Philadelphia or Pittsburgh for vacation?
- What's your favorite color? Answer just one.

Today's LLMs have Mode Collapse



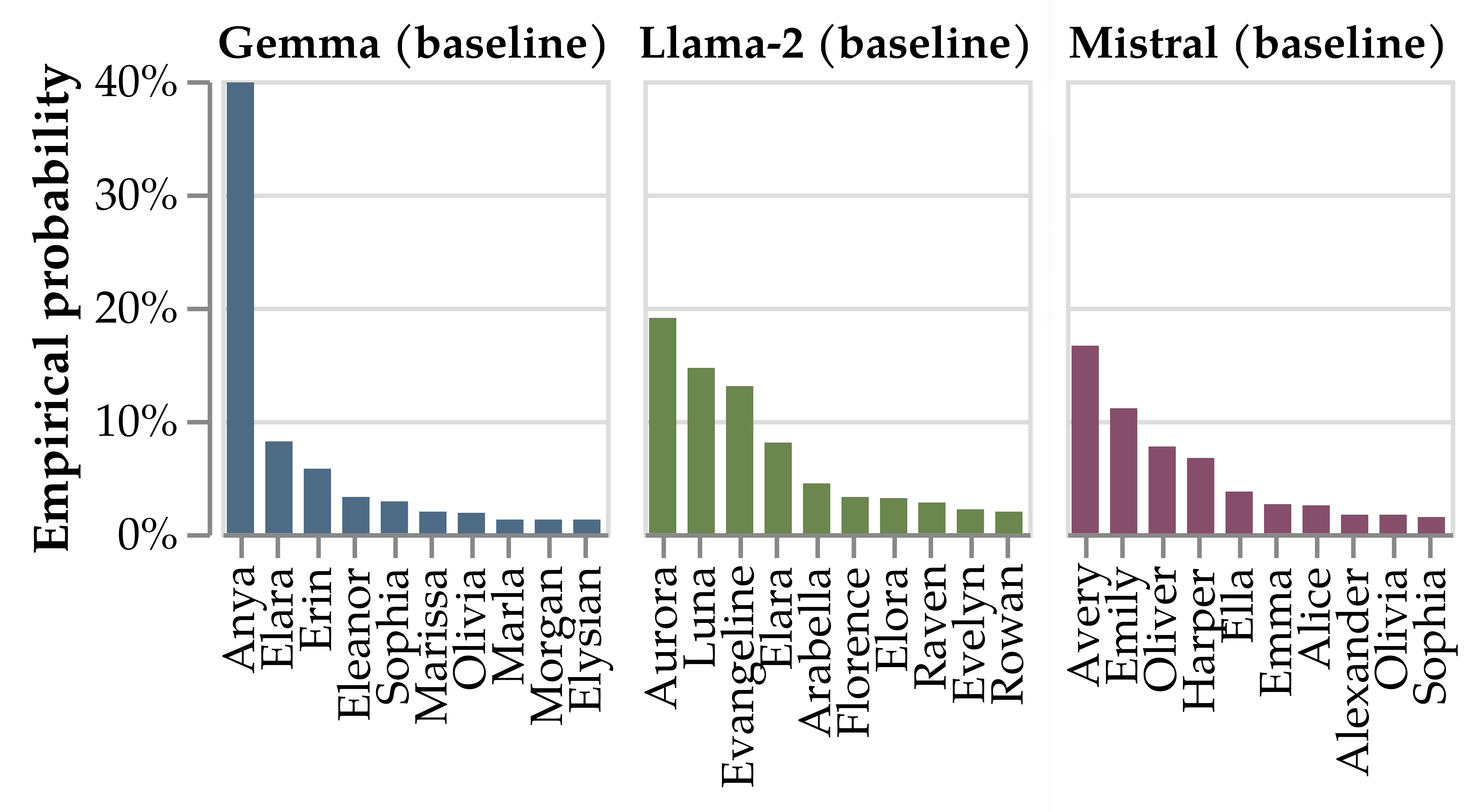
Today's LLMs have Mode Collapse

Alignment tuning has made this worse.



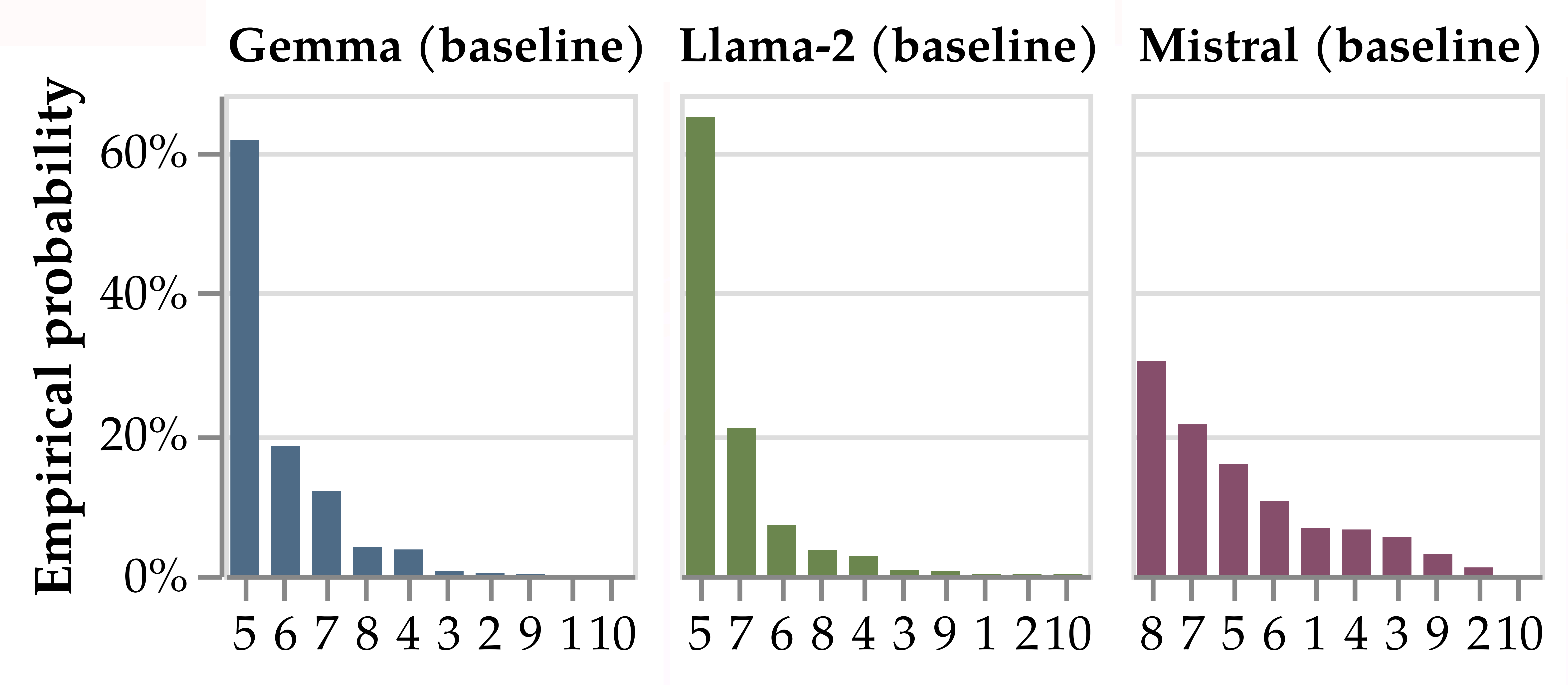
Today's LLMs have Mode Collapse

"Please generate an English first name, chosen completely at random."



Today's LLMs have Mode Collapse

"Generate a random number between 1 and 10."



Why is mode collapse a problem?

Why is mode collapse a problem?

- Bad for **writing tasks** that benefit from diversity (e.g. brainstorming assistants).
- Reinforcement of possibly harmful societal **biases**.
- **Rejection sampling** methods don’t work as well.
- Harder to build realistic **synthetic datasets.**

Fixing Mode Collapse

One Solution: A Bit of Finetuning

Method: For a handful of tasks, finetune the LLM to match the distribution we want by minimizing KL-divergence between model's distribution and true distribution.

Fixing Mode Collapse

One Solution: A Bit of Finetuning

Method: For a handful of tasks, finetune the LLM to match the distribution we want by minimizing KL-divergence between model's distribution and true distribution.

- Random dates in month
  - `“Provide a random date in June.”`
- Random number
  - `“Randomly pick a prime number between 1 and 50.”`
- Fruit selection
  - `“Output a name of a fruit, chosen completely at random.”`
- Name selection
  - `“Generate an English first name, chosen completely at random.”`
- Country selection
- Job selection

Today's LLMs have Mode Collapse

“Please generate an English first name, chosen completely at random.”

Today's LLMs have Mode Collapse

`"Generate a random number between 1 and 10."`

Does finetuning result in generalized diversity?

If we finetune an LLM to produce diverse outputs for tasks 1-5, will its outputs also be more diverse for task 6?

Fixing Mode Collapse

Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

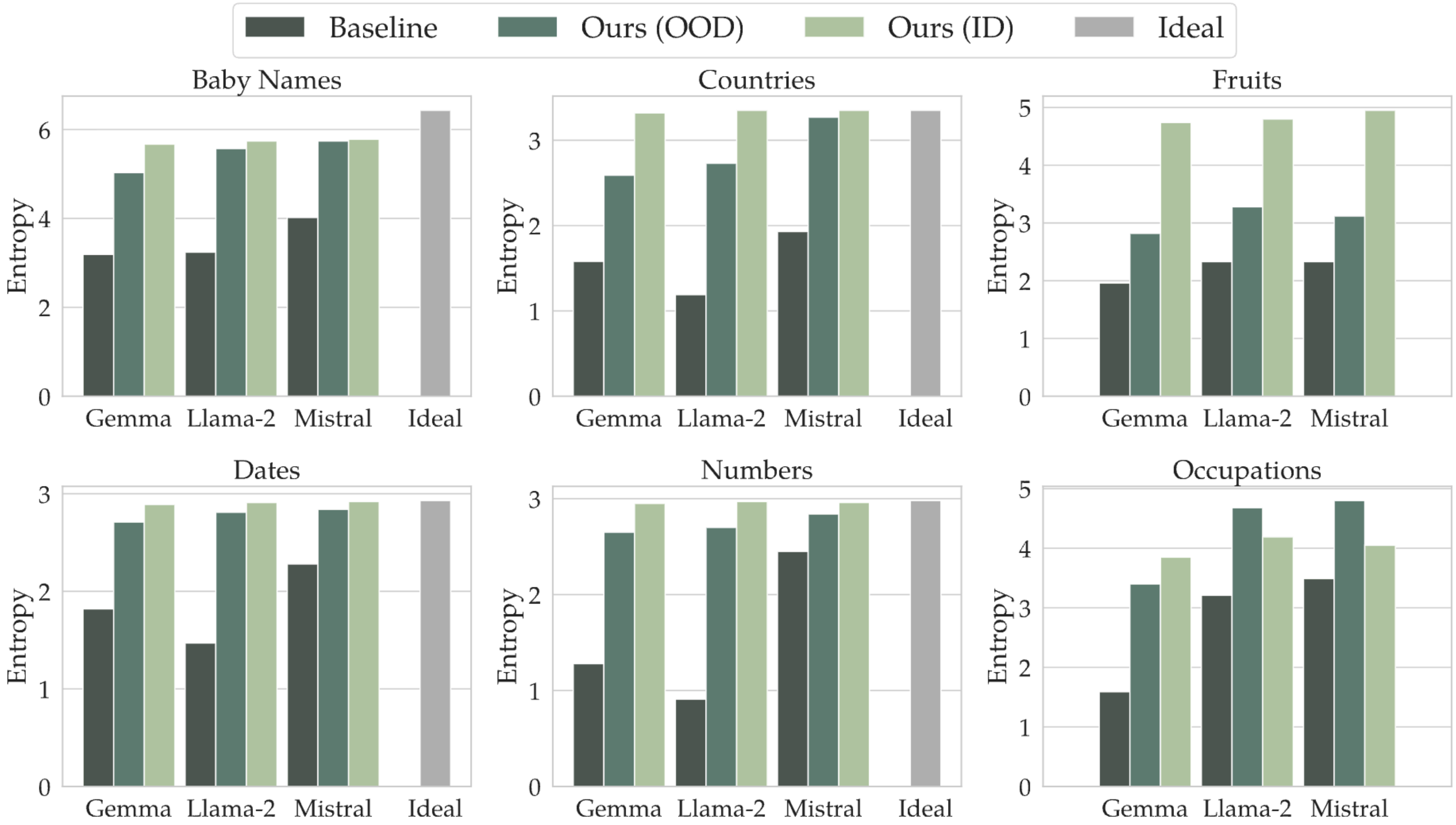
Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

Leave-one-out experiments demonstrate generalization.

Fixing Mode Collapse

Leave-one-out experiments demonstrate generalization.



Fixing Mode Collapse

We also see generalization to very different tasks.

The bio generation task:

```
"Generate a random biography sketch of a
fictional, notable person. Output name, gender,
time of birth, place of birth, profession and
accomplishments individually between two braces
and generate nothing else. Please follow the
format below. […]"
```

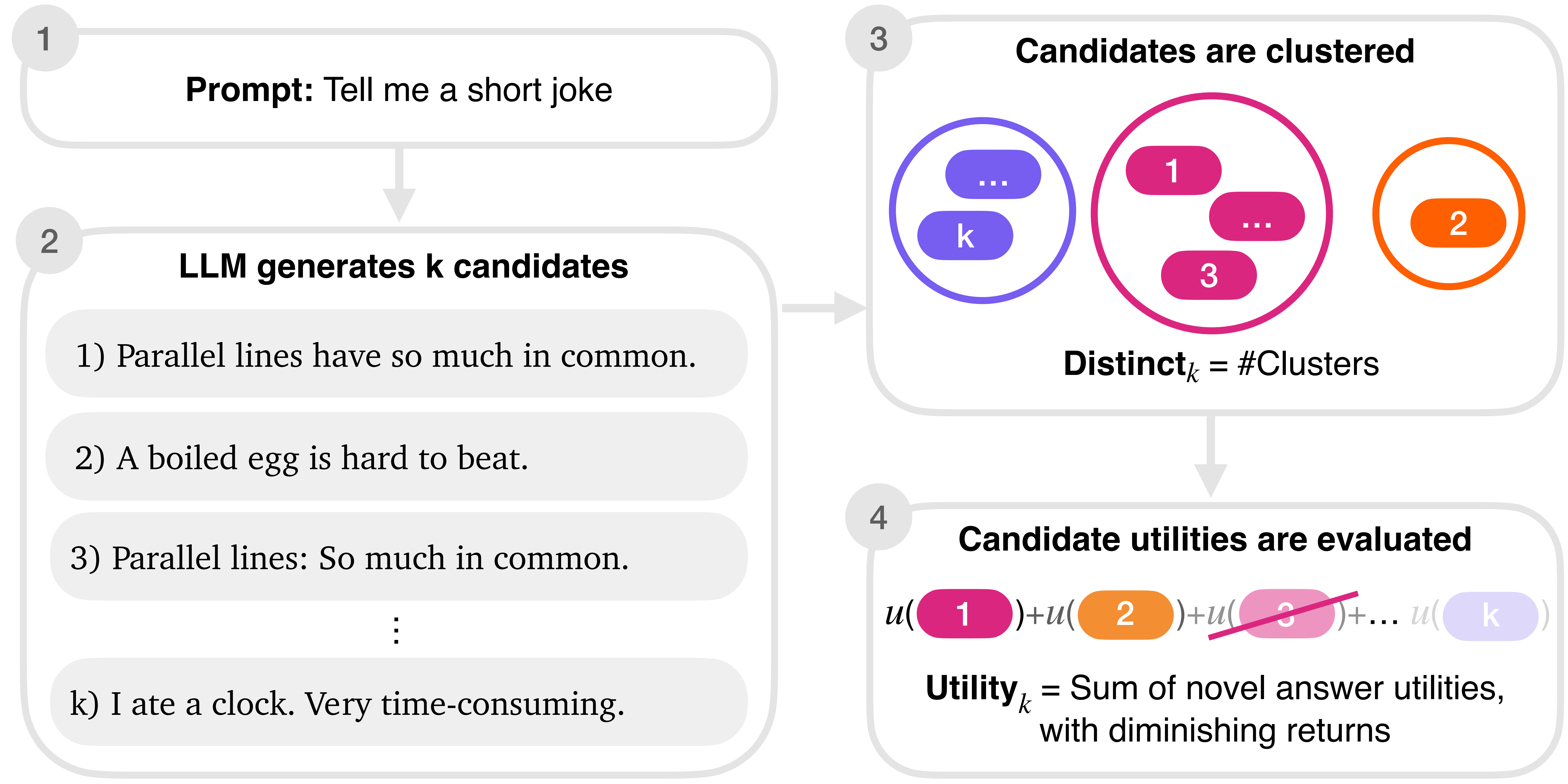
Fixing Mode Collapse

We also see generalization to very different tasks.

First name		Last name		Gender		Birth year		Birth place		Career	
Baseline Llama-2											
Evelyn	284	Nightingale	117	F	966	1985	764	Paris, FR	305	Astronaut	211
Luna	155	Aurora	104	NB	17	1987	46	Tokyo, JP	267	Astro. Engineer	66
Elara	87	Nova	98	M	13	1992	44	Stockholm, SE	33	Aero. Engineer	55
Adriana	42	Starling	53			1978	36	Mumbai, IN	32	Env. Activist	42
Aurora	38	Stardust	41			1975	31	Singapore, SG	32	Astrophysicist	32
Fine-tuned Llama-2 (OOD)											
Luna	32	Nightingale	16	F	762	1985	211	Mumbai, IN	35	Astronaut	96
Zelda	14	Nightshade	12	M	189	1992	99	Lagos, NG	31	Aero. Engineer	50
Mila	14	Chen	8	NB	31	1987	77	Paris, FR	29	Soft. Engineer	47
Evelyn	11	Orion	6			1988	61	Tokyo, JP	27	Env. Activist	35
Althea	9	Sparks	6			1990	52	Nairobi, KE	21	Journalist	34

NoveltyBench

Benchmarking Humanlike Diversity



Evaluating Novelty

Prompt Curation

- $\text{NB}^{\text{CURATED}}$ contains 100 prompts manually curated by my research group
- $\text{NB}^{\text{WILDCHAT}}$ consists of 1,000 prompts automatically curated from real user interactions with ChatGPT

Prompt: Tell me a story in five sentences about a girl and her dog.

Prompt: What is the top item you would … for a memor… …erience?

1

Prompt: Tell me a short joke

Prompt: What is the best book of all time?

Prompt: Name one reputed publication in science.

Prompt: Name one wild animal which is an omnivore.

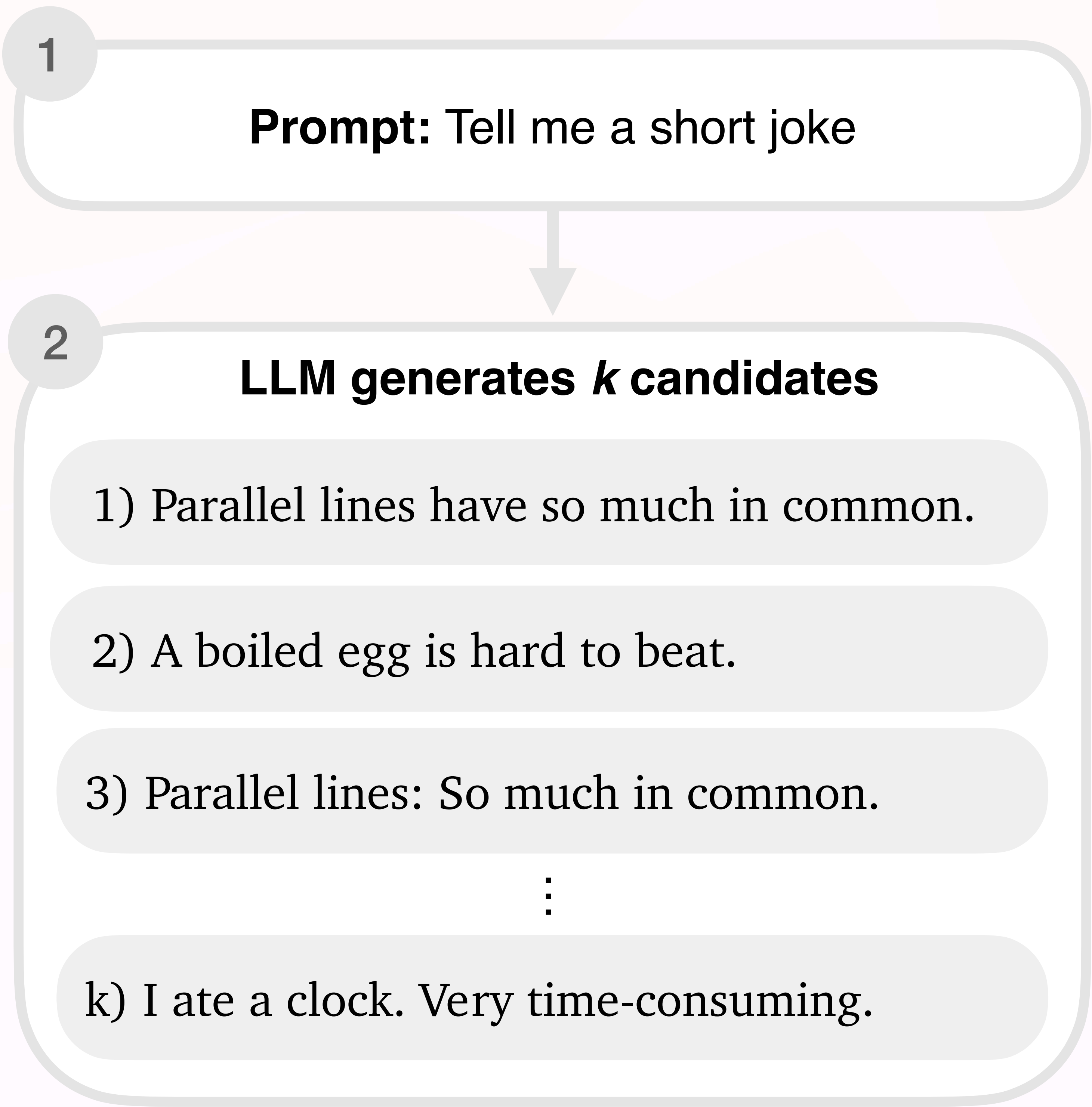
Prompt: Pretend to pick a card from the top of a standard deck of cards. What card did you pick?

Prompt: Generate a 5 word passphrase separated by hyphens.

Evaluating Novelty

Model Evaluated

Model Provider	Variants
Anthropic (Anthropic, 2024)	Claude-3.5 Haiku Claude-3.5 Sonnet Claude-3 Opus
OpenAI (OpenAI, 2024)	gpt-4o-mini gpt-4o
Gemini (Google, 2024)	gemini-1.5-pro gemini-2.0-flash-lite gemini-2.0-flash gemini-2.0-pro
Cohere (Cohere, 2024)	command-r7b command-r command-r-plus
Gemma 2 (Gemma Team et al., 2024)	gemma-2-2b-it gemma-2-9b-it gemma-2-27b-it
Llama 3 (Llama Team et al., 2024)	Llama-3.2-1B Llama-3.2-3B Llama-3.1-8B Llama-3.3-70B Llama-3.1-405B



Evaluating Novelty

Diversity of Generations

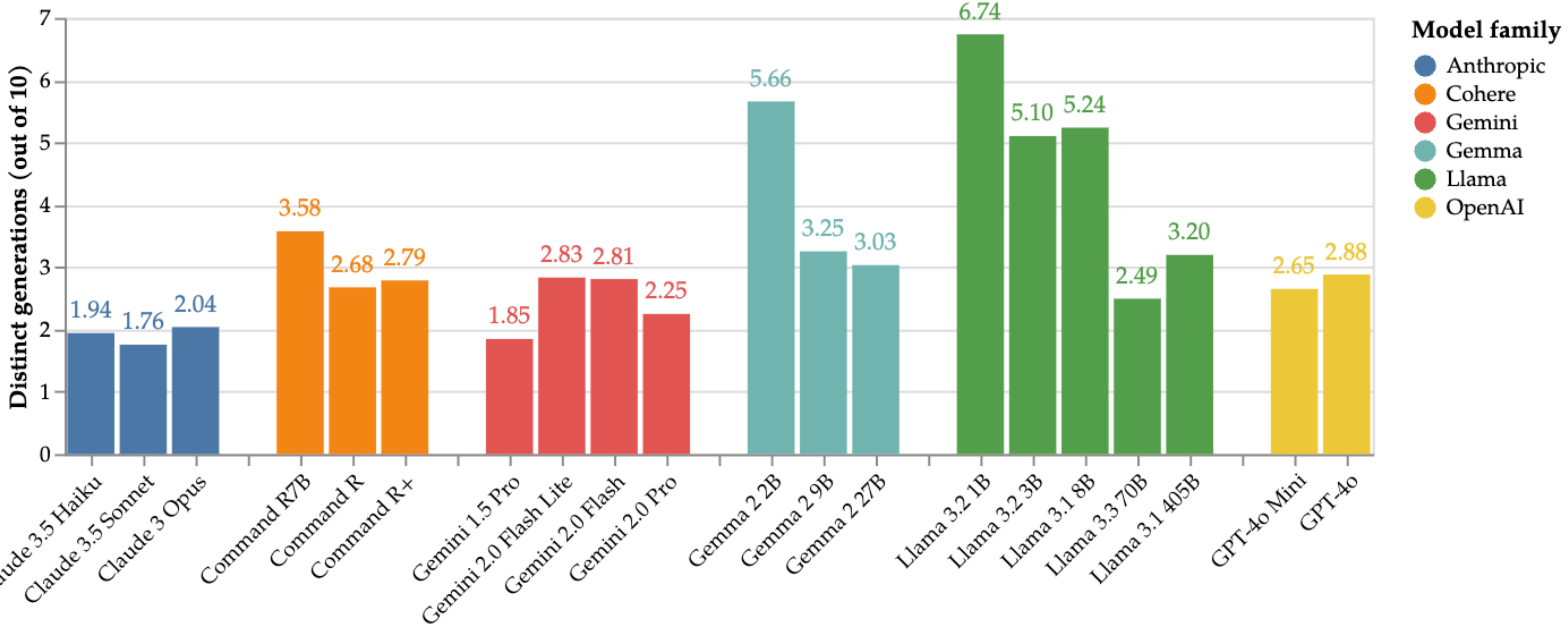


Figure 2: Average number of unique generations out of a sample of 10 for all prompts in NoveltyBench.

Evaluating Novelty

Utility of Generations

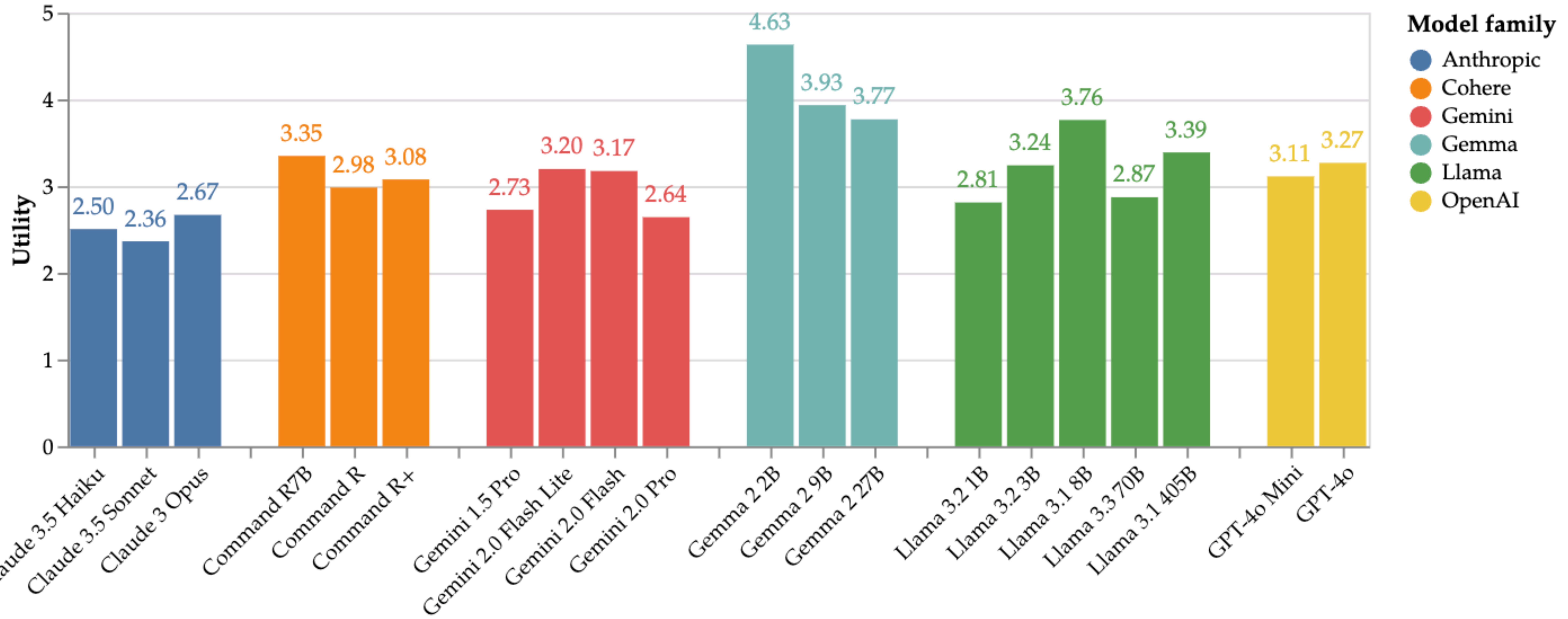


Figure 3: Cumulative utility of generations of state-of-the-art models on NOVELTYBENCH. A perfectly diverse and helpful model would have cumulative utility of 10.

Evaluating Novelty

Other Ways to Elicit Diversity

- Resampling:
  - Akin to refreshing the conversation and pasting in the same prompt
- Paraphrasing:
  - Try out different versions of the same prompt
  - “Roll a six-sided die” vs. “plz roll a 6-sided die”
- System prompts
  - Explicit instruction to the LLM that diversity is desired
  - “You are an AI that excels in producing diverse responses…”
- In-context regeneration
  - "Give me a different answer"

Evaluating Novelty

More results with sampling methods

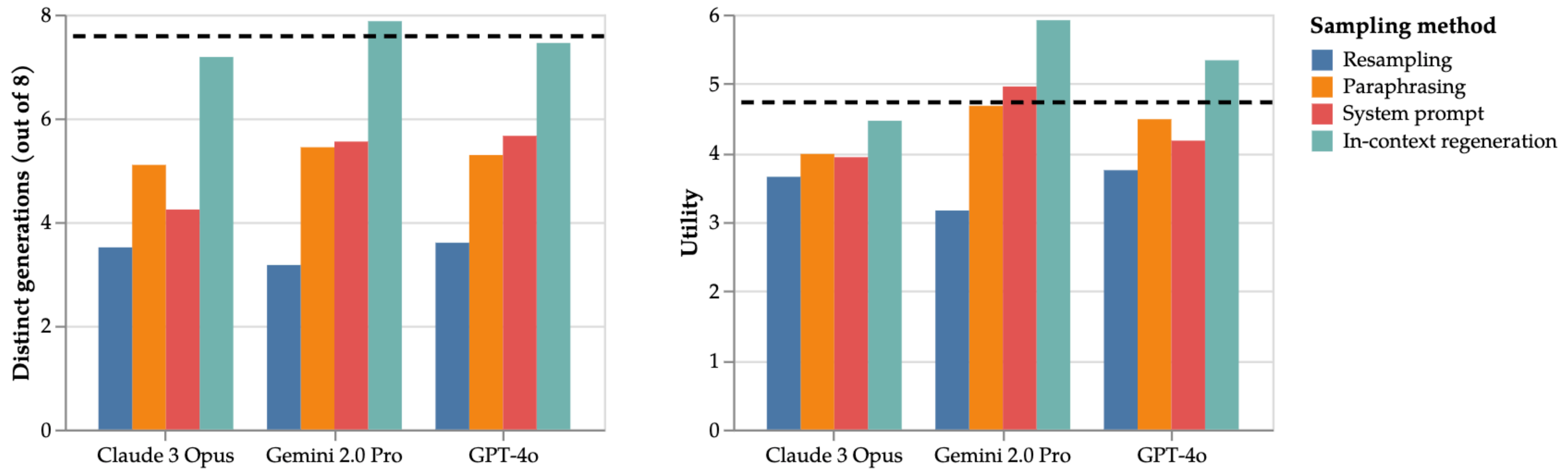


Figure 5: Alternative prompting methods *can lead to improved novelty*. The dashed lines report diversity and utility of answers handwritten by authors.

References

Forcing Diffuse Distributions out of Language Models
Yiming Zhang, Avi Schwarzschild, Nicholas Carlini, Zico Kolter, Daphne Ippolito. COLM 2024

NoveltyBench: Evaluating Language Models for Humanlike Diversity
Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, Daphne Ippolito. COLM 2025.

Other Research Questions I’m Thinking About

- AI tools for supporting research in the humanities
- Improving legibility of LLM reasoning traces for human readers
- Better algorithms for checking for LLM memorization of pre-training data and other string matching tasks
- Stance detection in social media media content
- Prompt robustness
- Automatic redteaming