Fine-tuning 2

Haw-Shiuan Chang

Deadlines

- **https://people.cs.umass.edu/~hschang/cs685/schedule.html**

- **3/7**: Project proposals due
  - In your proposal, please estimate the cost of API credit you need and which LLM and service provider you plan to use.
  - If you submit one day late, you will lose 5 points. You have to submit the proposal before 3/9.
- **3/14**: HW 1 due
- **3/17:** Quiz 3
  - Released today
- **5/9**: Last day to submit extra credit
  - Please check the announcement at Piazza for the recording link

An example proposal

- Introduction / problem statement
- Motivation (why should we care? why is this problem interesting?)
- Literature review (what has prev. been done?)
- Possible datasets
- Evaluation
- Tools and resources
- Project milestones / tentative schedule

Task -> Data -> Evaluation -> Loss -> Model -> Optimization

- Task:
  - Predict the next token
- Loss:
  - Cross-entropy
- Model:
  - Transformer
- Optimization:
  - Gradient Descent

- Step 1: Determine the task and goal
  - Assuming the goal is to improve the performance in a task
- Step 2: What are the datasets?
  - If no dataset, create a dataset
- Step 3: How to evaluate the performance?
- (Step 1-3 could be skipped if they have been defined)
- Step 4: Define the loss function
  - If the evaluation is reliable and deferentiable -> loss
  - Consider to prompt or **fine-tune** the model
- Step 5: Choose the model
  - **Select from various LMs**
- Step 6: Choose the optimization method and hyperparameters

Which LM(s) should I try?

Neural LM, Self-attention, Transformer LM

How to Set the Hyperparameters?

Optimization

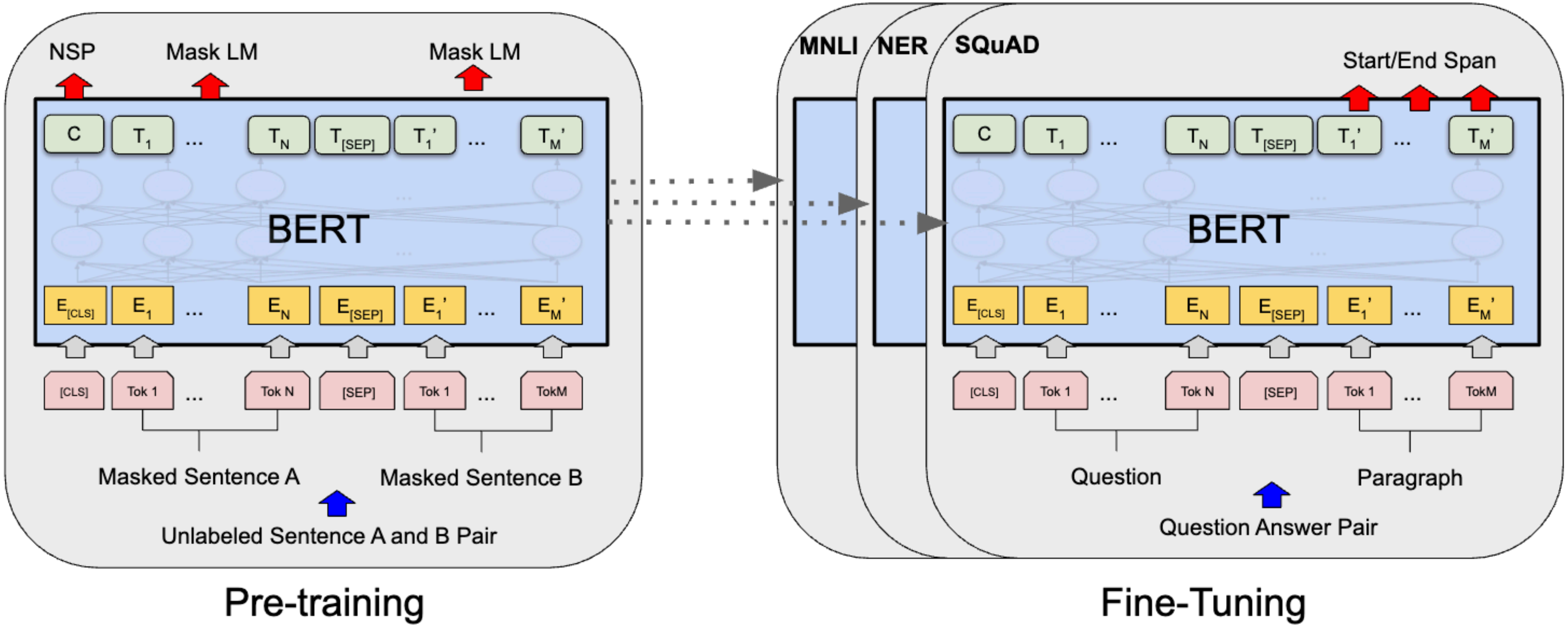
Will Fine-tuning Make the Performance Better?

Last course, this course

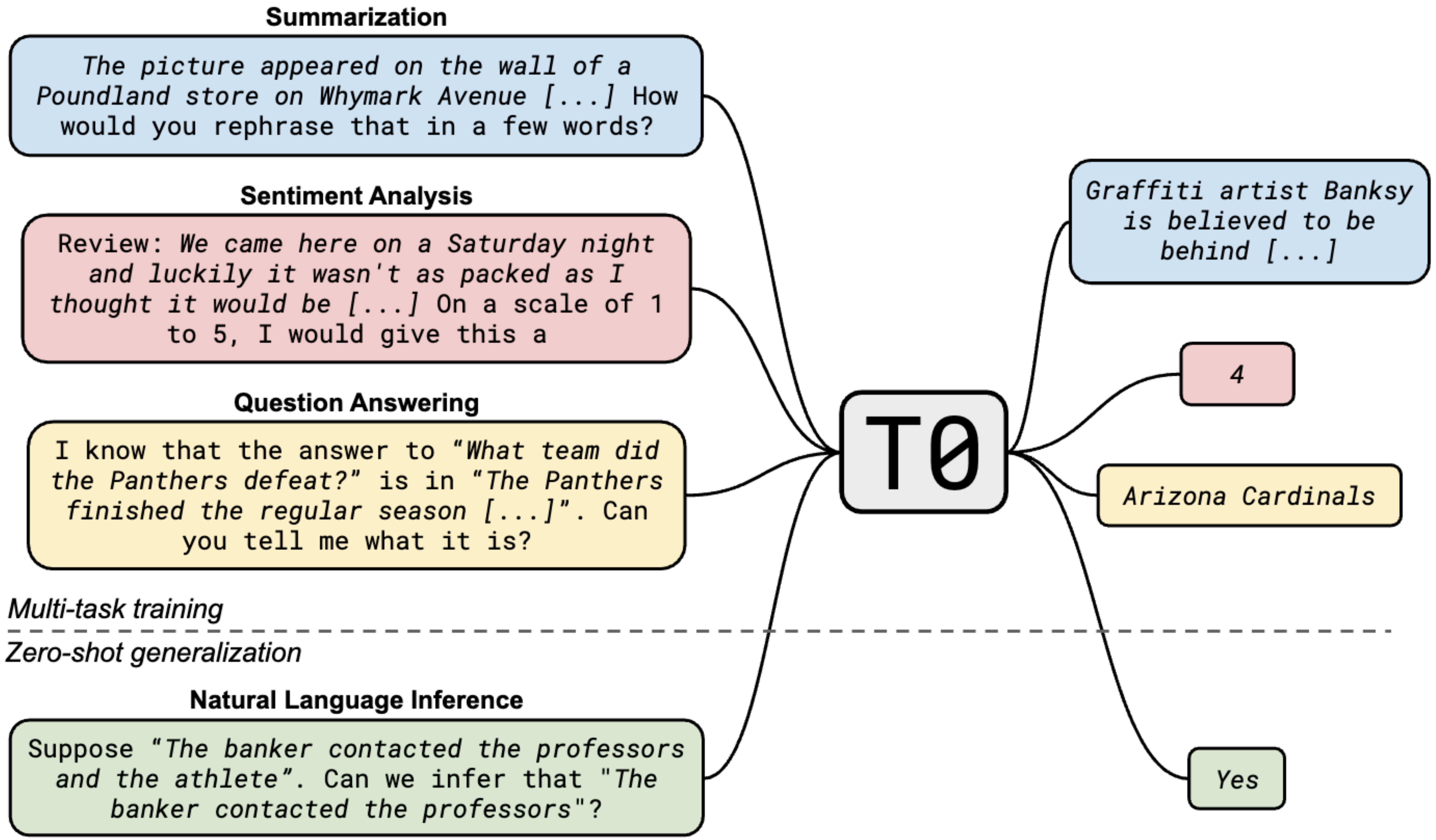
What if I don’t have Sufficient GPU Memory?

This course

Old Fine-Tuning (Hw1)



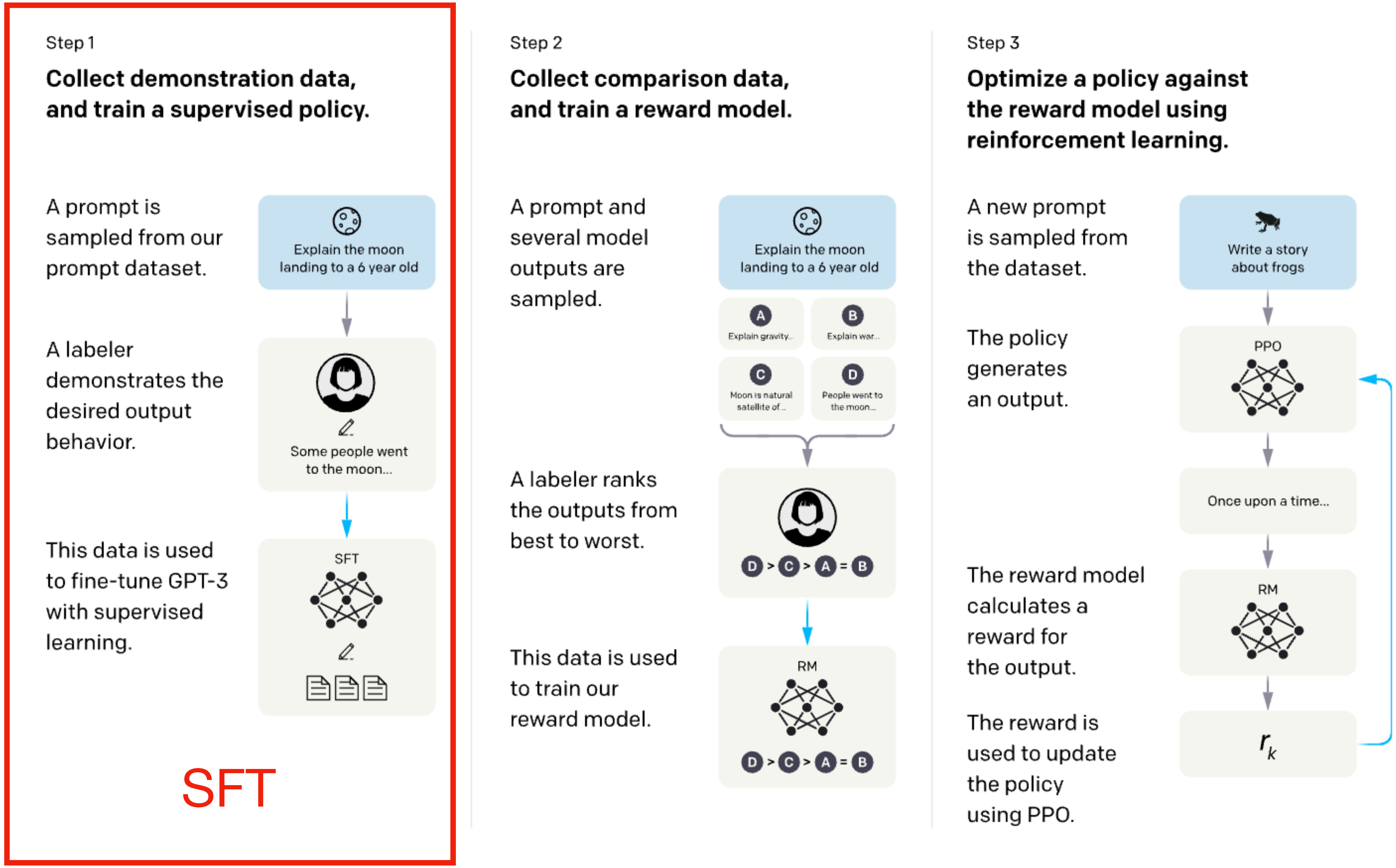
Instruction Tuning



Summarization
The picture appeared on the wall of a Poundland store on Whymark Avenue [...] How would you rephrase that in a few words?
Sentiment Analysis
Review: *We came here on a Saturday night and luckily it wasn't as packed as I thought it would be [...]* On a scale of 1 to 5, I would give this a
Question Answering
I know that the answer to "*What team did the Panthers defeat?*" is in "*The Panthers finished the regular season [...]*". Can you tell me what it is?
Multi-task training
Zero-shot generalization
Natural Language Inference
Suppose "*The banker contacted the professors and the athlete*". Can we infer that "*The banker contacted the professors*"?
T0
Graffiti artist Banksy is believed to be behind [...]
4
Arizona Cardinals
Yes

MULTITASK PROMPTED TRAINING ENABLES ZERO-SHOT TASK GENERALIZATION (https://arxiv.org/pdf/2110.08207)

Supervised Fine-Tuning (SFT)



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT

SFT

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.

Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
D > C > A = B
RM
D > C > A = B

Step 3

Optimize a policy against the reward model using reinforcement learning.

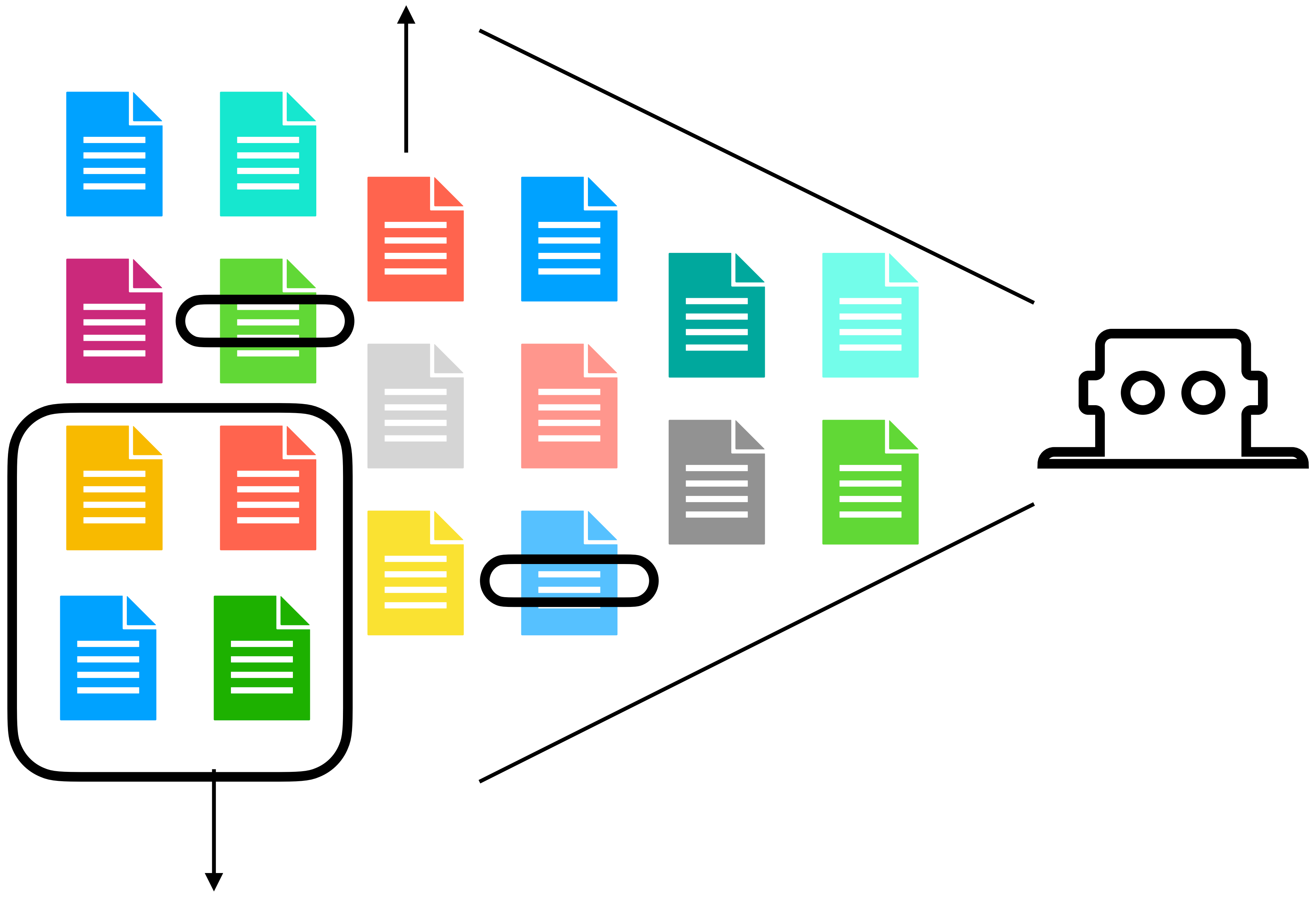
A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

LLM Development



Internet low quality text

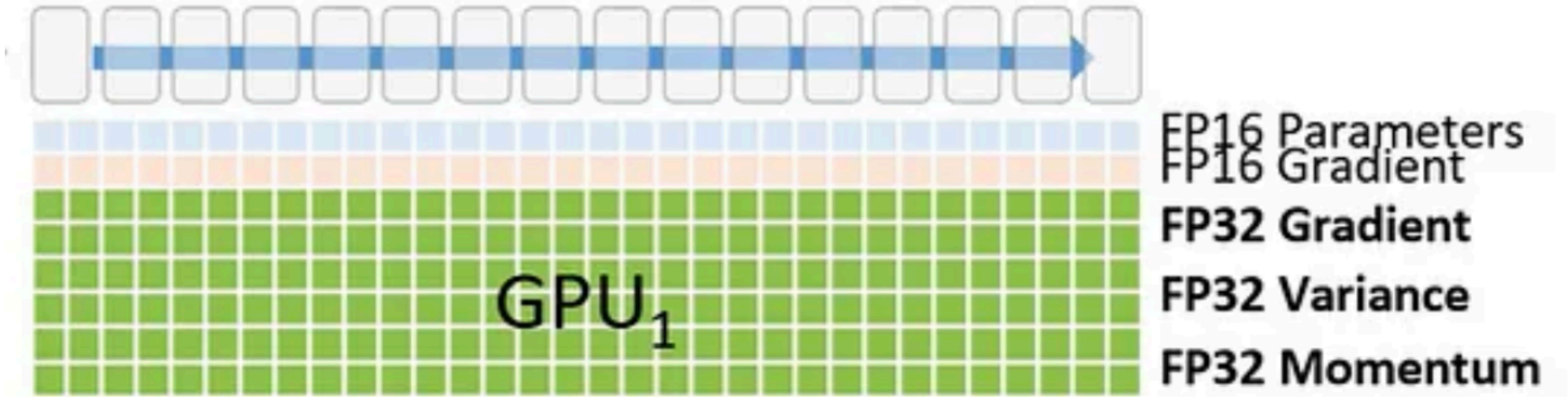
Internet high quality text

- Architectures
  - MLP
  - RNN
  - Transformer

- Training Stages
  - Pretraining
  - **Supervised Fine-tuning (SFT)**
  - Alignment
    - Learning from Human Feedback (LHF)
    - Reasoning

Post-training stage
(Filtering process)

Fine-tuning LLM is very Expensive



3B LLM

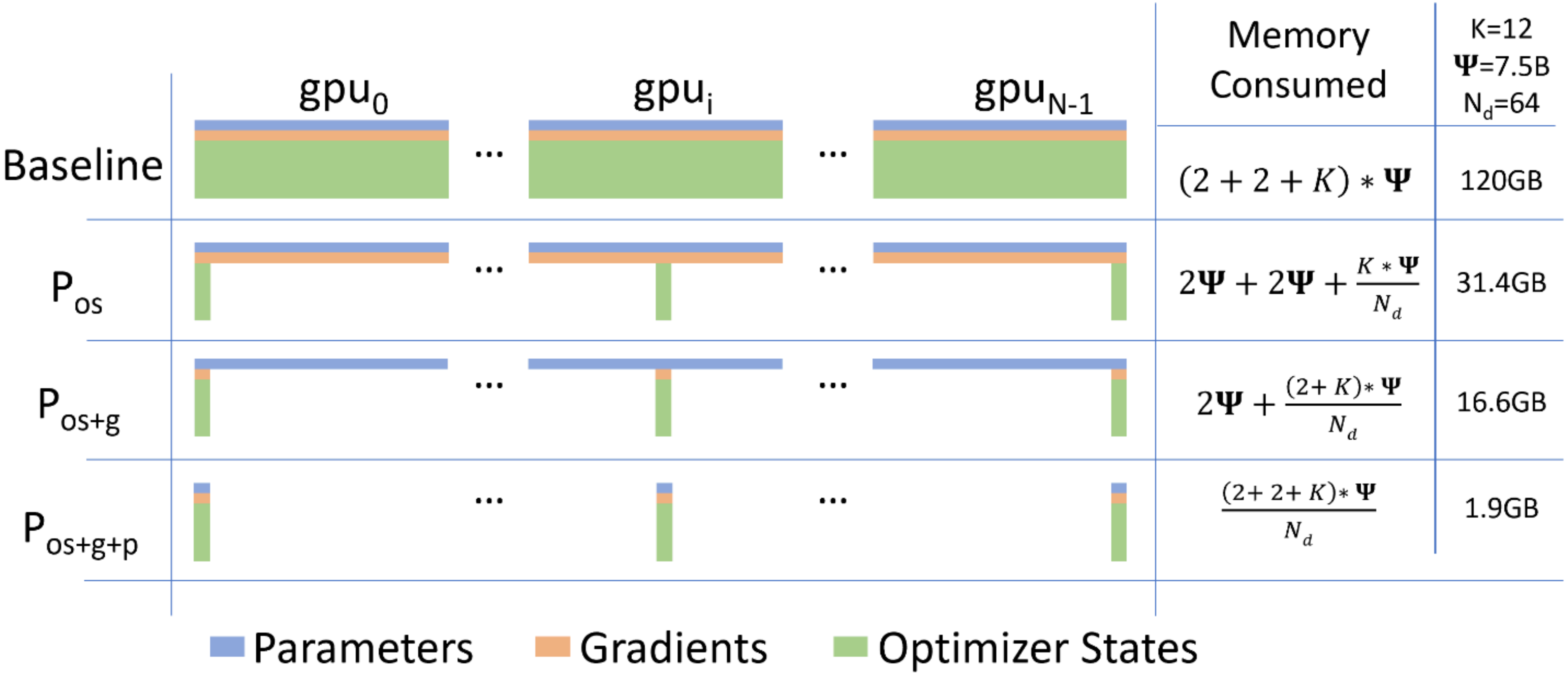
6B Parameters

6B Gradient

12B*3 Adam

3B * 16 = 48GB

ZeRO



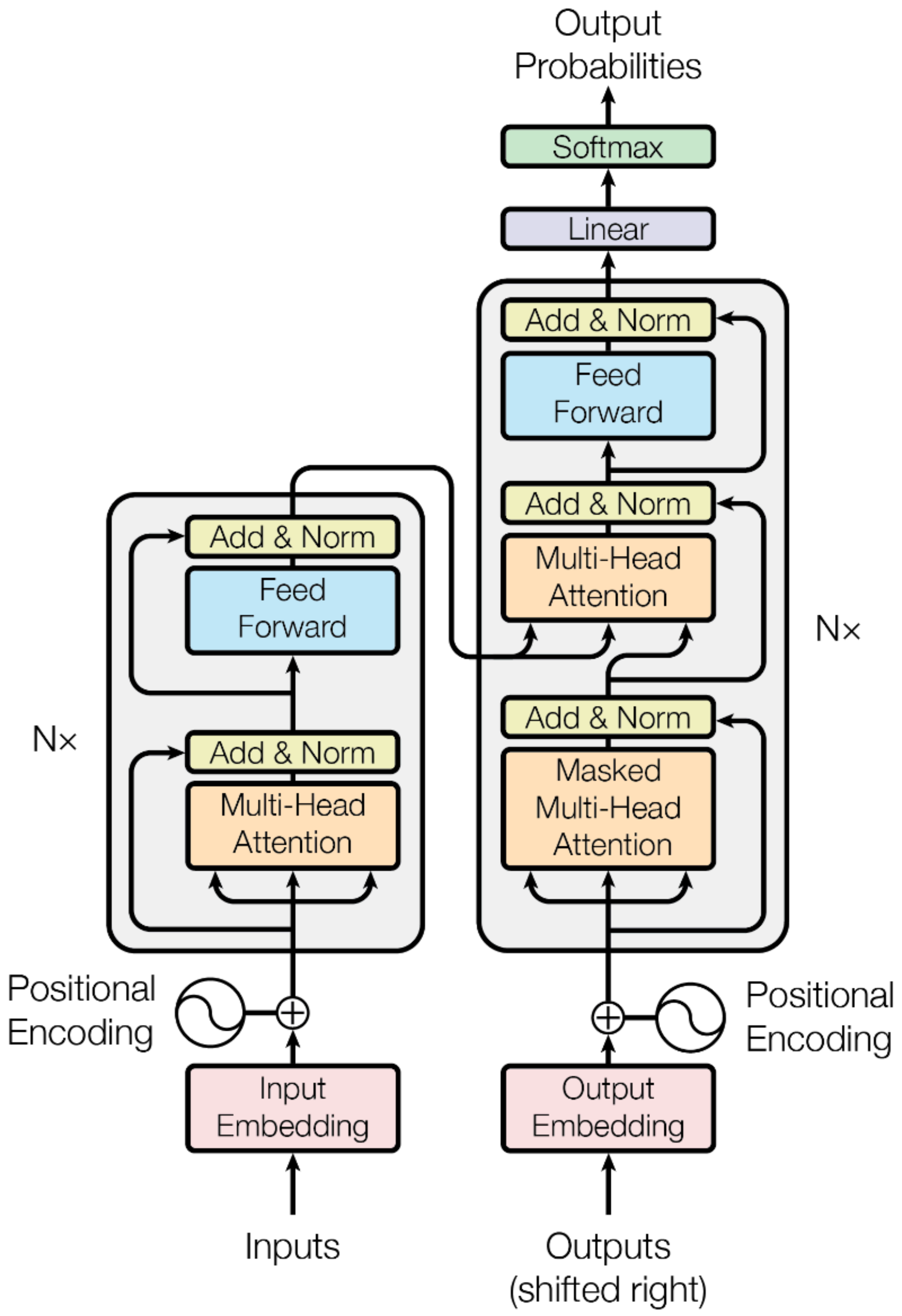
	Memory Consumed	K=12 Ψ=7.5B N_d=64
Baseline	$(2+2+K)*\Psi$	120GB
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB

DeepSpeed (https://huggingface.co/docs/accelerate/en/usage_guides/deepspeed)

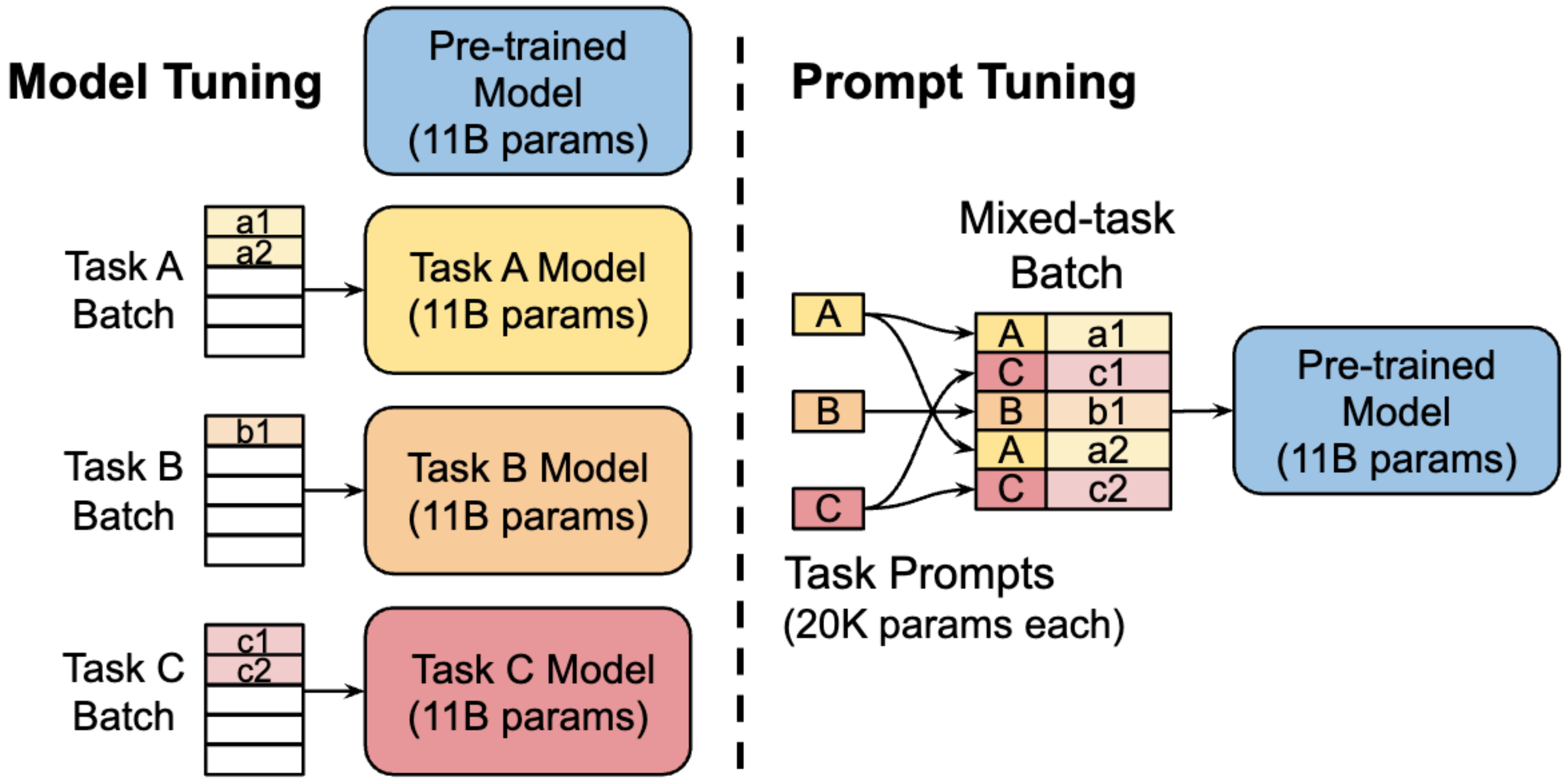
ZeRO: Memory Optimizations Toward Training Trillion Parameter Models (https://arxiv.org/abs/1910.02054)

https://github.com/vllm-project/vllm https://github.com/unslothai/unsloth https://github.com/hiyouga/LLaMA-Factory

Last Year Notes



Prompt Tuning



Model Tuning

Pre-trained Model (11B params)

Task A Batch
a1
a2
Task A Model (11B params)

Task B Batch
b1
Task B Model (11B params)

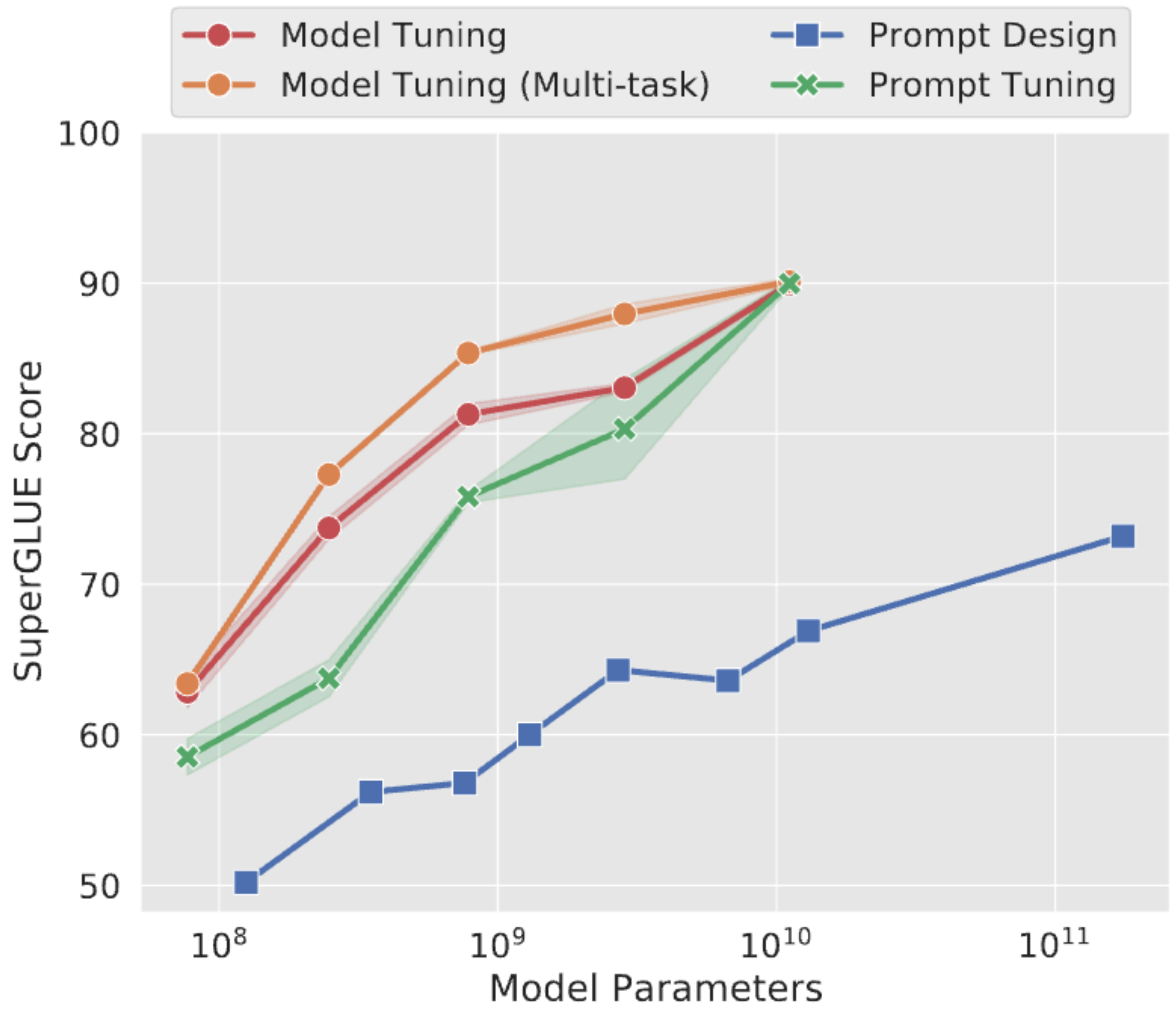
Task C Batch
c1
c2
Task C Model (11B params)

Prompt Tuning

Mixed-task Batch
A a1
C c1
B b1
A a2
C c2

A
B
C
Task Prompts (20K params each)

Pre-trained Model (11B params)



While the learned prompts taken as sequences show little interpretability, we do observe a high frequency of words like science, technology and engineering as the nearest neighbors for prompts trained on the BoolQ dataset and approximately 20% of the questions are in the “Nature/Science” category.

The Power of Scale for Parameter-Efficient Prompt Tuning (https://arxiv.org/pdf/2104.08691)

LoRA Performances

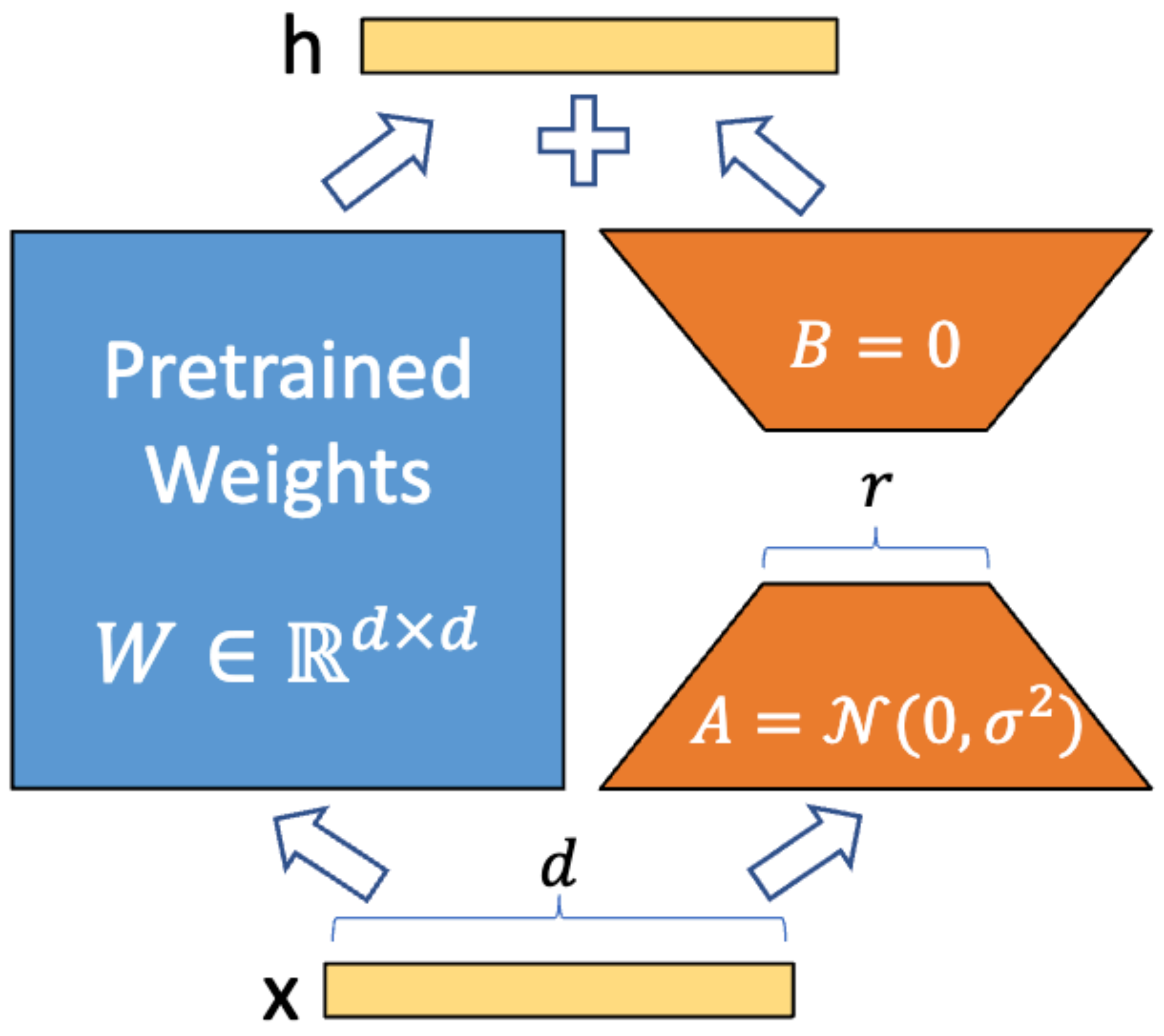
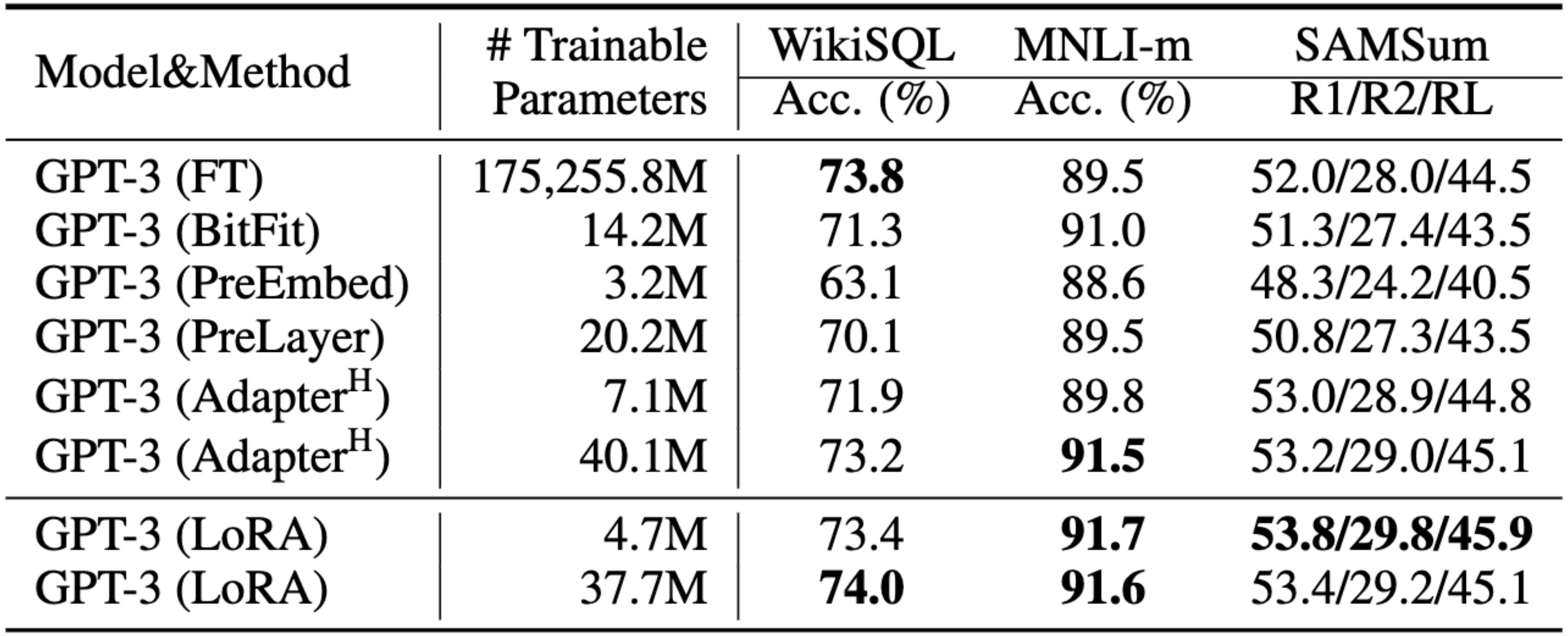

h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

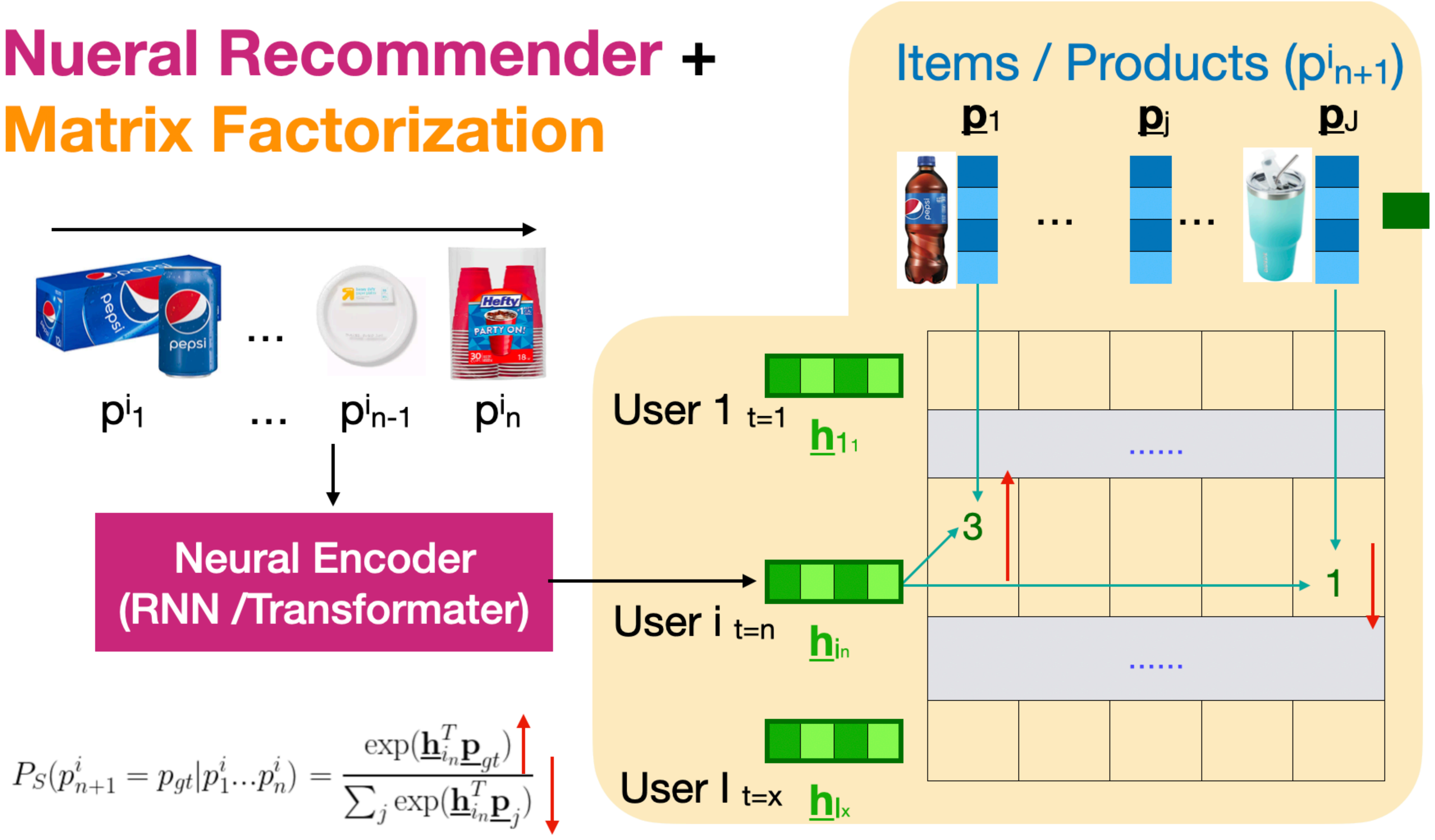
Figure 1: Our reparametrization. We only train A and B.



Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter^{H})	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter^{H})	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Matrix Factorization Again!

Nueral Recommender + Matrix Factorization



$$P_S(p^i_{n+1} = p_{gt} | p^i_1 \ldots p^i_n) = \frac{\exp(\mathbf{h}^T_{i_n}\underline{\mathbf{p}}_{gt})\uparrow}{\sum_j \exp(\mathbf{h}^T_{i_n}\underline{\mathbf{p}}_j)\downarrow}$$

- Output Softmax Layer
- Self-attention
  - Multi-Head Latent Attention (won't appear in the midterm)
- MLP
- LoRA

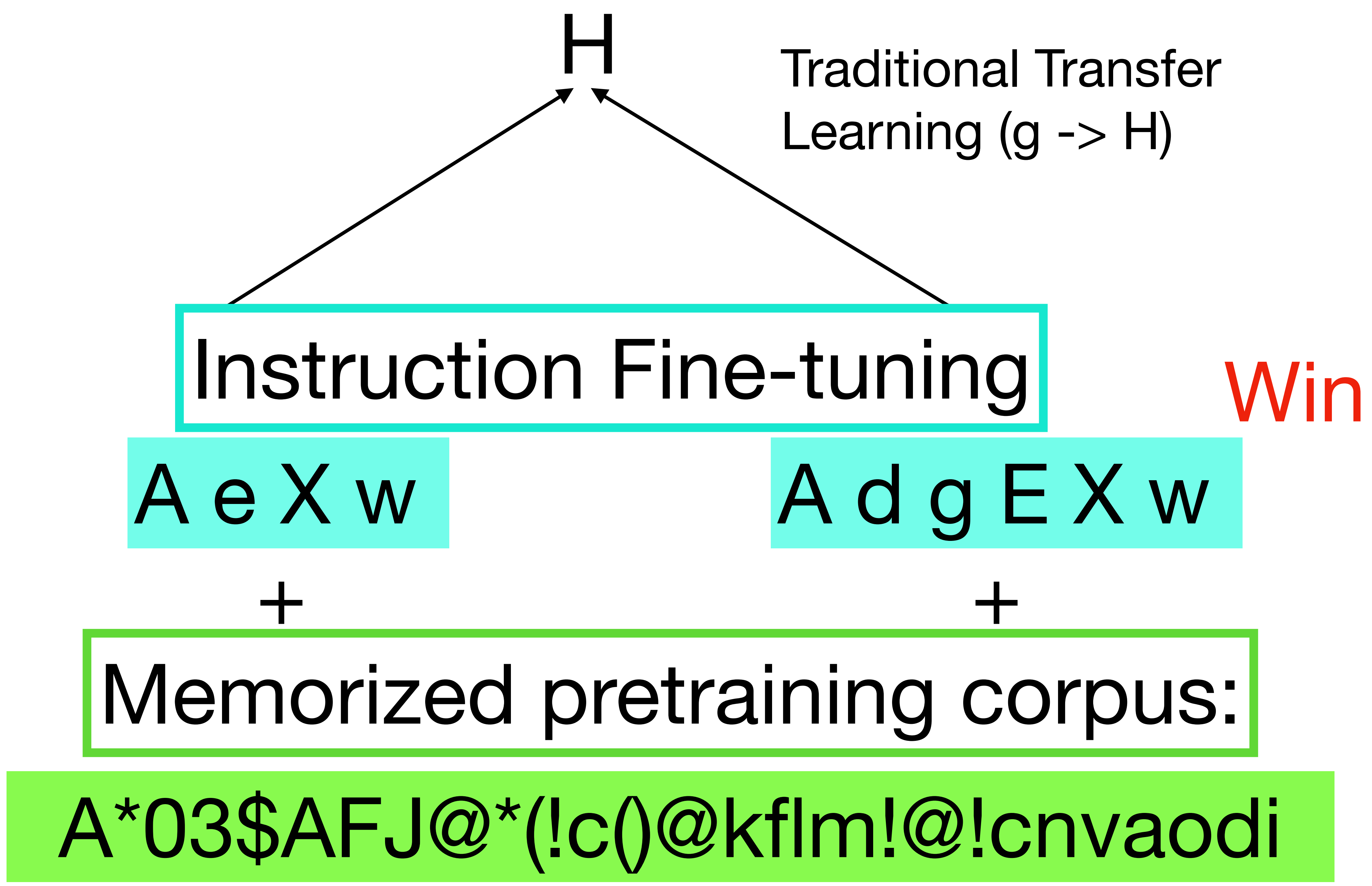
- Other Applications
  - Recommendation, Information Retrieval, Computer Vision …

	Softmax	Self-Attention	MLP	LoRA
Matrix	Prob -> N-gram Statistics	Attention	Positive Activations	Weight Change
Nonlinearity	Softmax	Softmax	ReLU or others	NA
q	Hidden State	W_Q(Hidden State)	Hidden State	**A row in B**
K	**Word Embeding**	W_K(Hidden States)	**Layer 1 W**	**A**
V	NA	W_V(Hidden States)	**Layer 2 W**	NA

Boldface means the parameters

Why Could Fewer Data be Better?

- First task -> A: high-quality data, a: low-quality data



H

Traditional Transfer Learning (g -> H)

Instruction Fine-tuning

Win

A e X w

A d g E X w

+

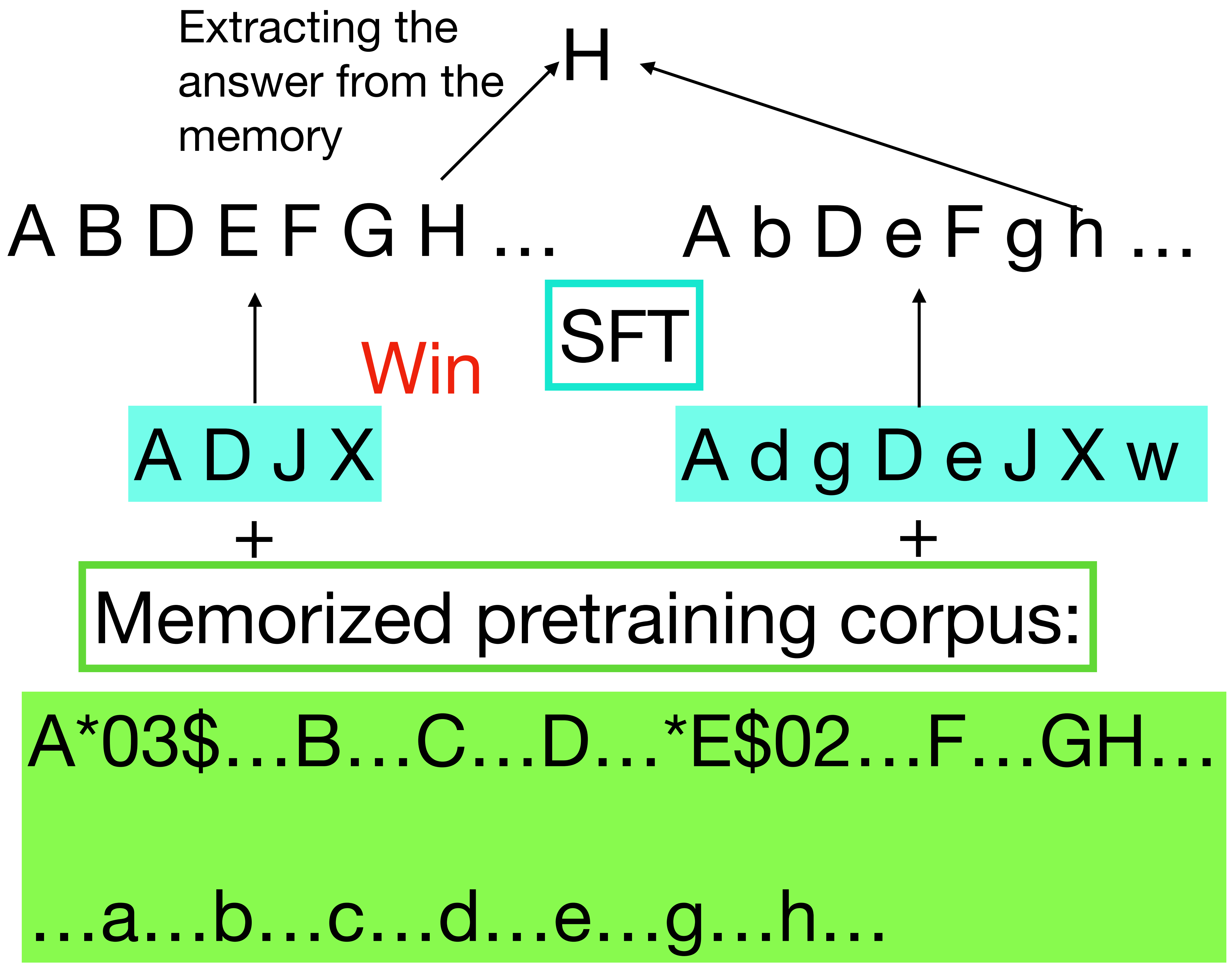
+

Memorized pretraining corpus:

A*03$AFJ@*(!c()@kflm!@!cnvaodi

Recent studies show that such transfer learning does not actually work generally. See this paper:

Do Models Really Learn to Follow Instructions? An Empirical Study of Instruction Tuning (https://arxiv.org/pdf/2305.11383)



Extracting the answer from the memory

H

A B D E F G H …

A b D e F g h …

SFT

Win

A D J X

A d g D e J X w

+

+

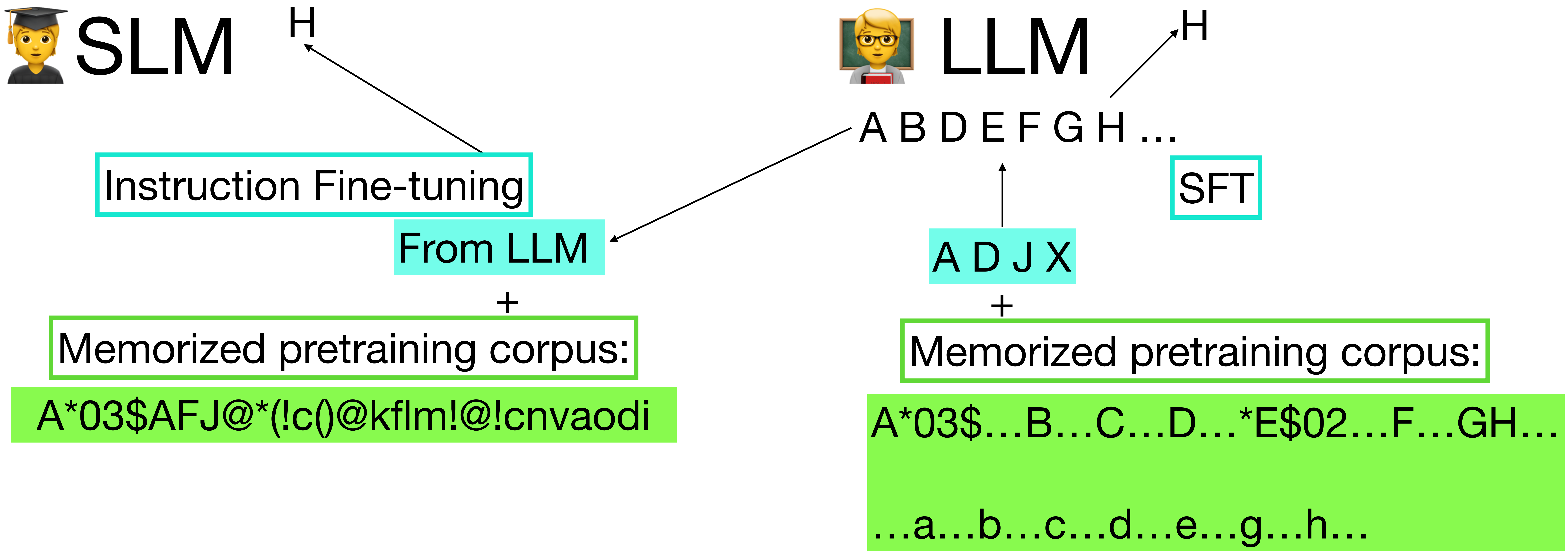
Memorized pretraining corpus:

A*03$…B…C…D…*E$02…F…GH…

…a…b…c…d…e…g…h…

Distillation

- First task -> A: high-quality data, a: low-quality data



SLM

H

Instruction Fine-tuning

From LLM

+

Memorized pretraining corpus:

A*03$AFJ@*(!c()@kflm!@!cnvaodi

LLM

H

A B D E F G H …

SFT

A D J X

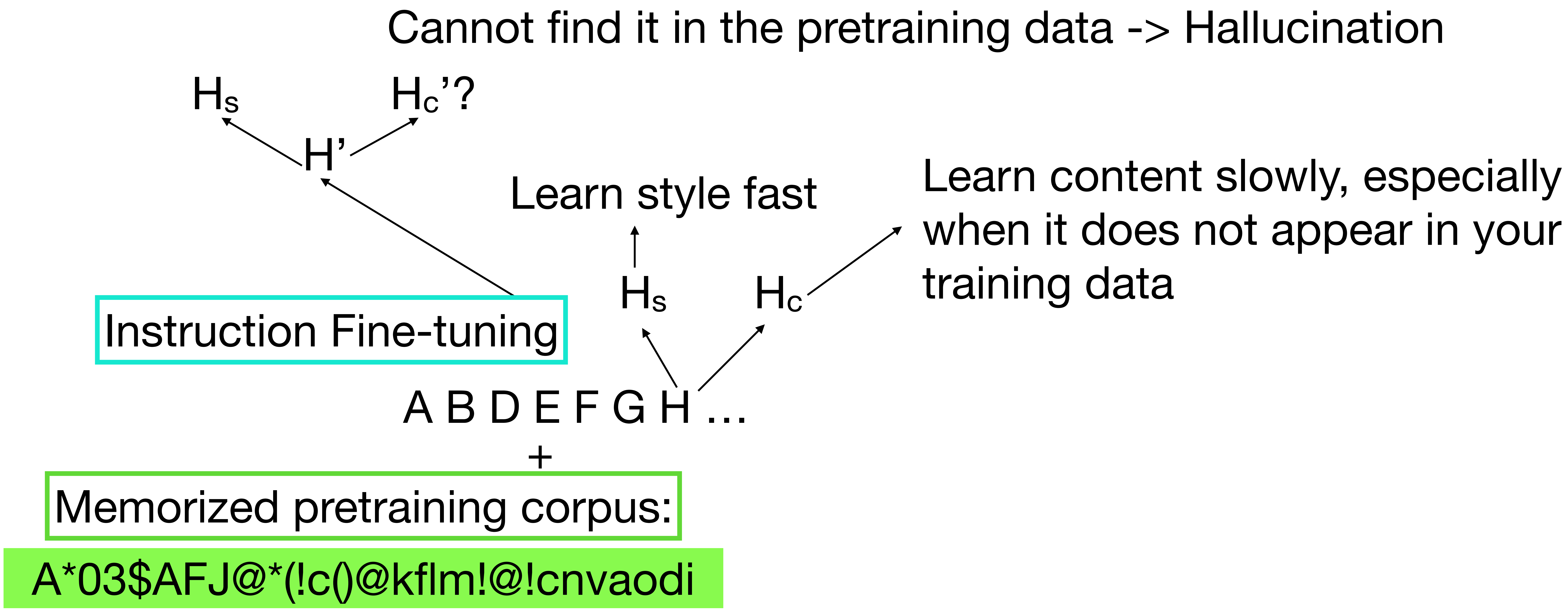
+

Memorized pretraining corpus:

A*03$…B…C…D…*E$02…F…GH…

…a…b…c…d…e…g…h…

Limitations of Fine-tuning



Cannot find it in the pretraining data -> Hallucination

H_s H_c'?

H'

Learn style fast

Learn content slowly, especially when it does not appear in your training data

H_s H_c

Instruction Fine-tuning

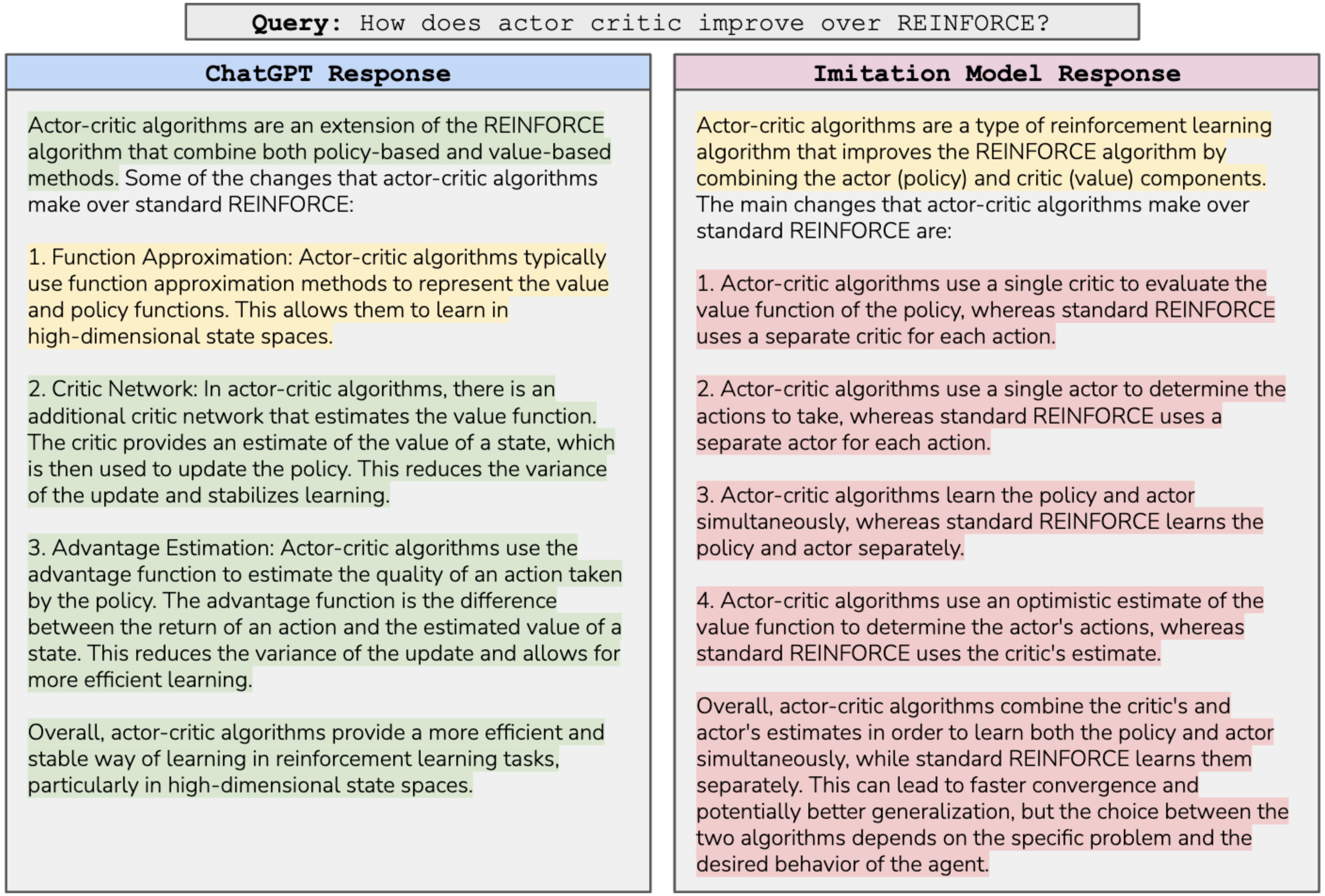
A B D E F G H …

+

Memorized pretraining corpus:

A*03$AFJ@*(!c()@kflm!@!cnvaodi

Fine-tuning mostly Changes the Style



Query: How does actor critic improve over REINFORCE?

ChatGPT Response

Actor-critic algorithms are an extension of the REINFORCE algorithm that combine both policy-based and value-based methods. Some of the changes that actor-critic algorithms make over standard REINFORCE:

1. Function Approximation: Actor-critic algorithms typically use function approximation methods to represent the value and policy functions. This allows them to learn in high-dimensional state spaces.

2. Critic Network: In actor-critic algorithms, there is an additional critic network that estimates the value function. The critic provides an estimate of the value of a state, which is then used to update the policy. This reduces the variance of the update and stabilizes learning.

3. Advantage Estimation: Actor-critic algorithms use the advantage function to estimate the quality of an action taken by the policy. The advantage function is the difference between the return of an action and the estimated value of a state. This reduces the variance of the update and allows for more efficient learning.

Overall, actor-critic algorithms provide a more efficient and stable way of learning in reinforcement learning tasks, particularly in high-dimensional state spaces.

Imitation Model Response

Actor-critic algorithms are a type of reinforcement learning algorithm that improves the REINFORCE algorithm by combining the actor (policy) and critic (value) components. The main changes that actor-critic algorithms make over standard REINFORCE are:

1. Actor-critic algorithms use a single critic to evaluate the value function of the policy, whereas standard REINFORCE uses a separate critic for each action.

2. Actor-critic algorithms use a single actor to determine the actions to take, whereas standard REINFORCE uses a separate actor for each action.

3. Actor-critic algorithms learn the policy and actor simultaneously, whereas standard REINFORCE learns the policy and actor separately.

4. Actor-critic algorithms use an optimistic estimate of the value function to determine the actor's actions, whereas standard REINFORCE uses the critic's estimate.

Overall, actor-critic algorithms combine the critic's and actor's estimates in order to learn both the policy and actor simultaneously, while standard REINFORCE learns them separately. This can lead to faster convergence and potentially better generalization, but the choice between the two algorithms depends on the specific problem and the desired behavior of the agent.

The False Promise of Imitating Proprietary LLMs (https://arxiv.org/pdf/2305.15717)

Fine-tuned LM could be Worse

- Learning the style but not the facts could be dangerous
- Hard to learn new facts during fine-tuning
- Parameter-efficient fine-tuning could alleviate the problem

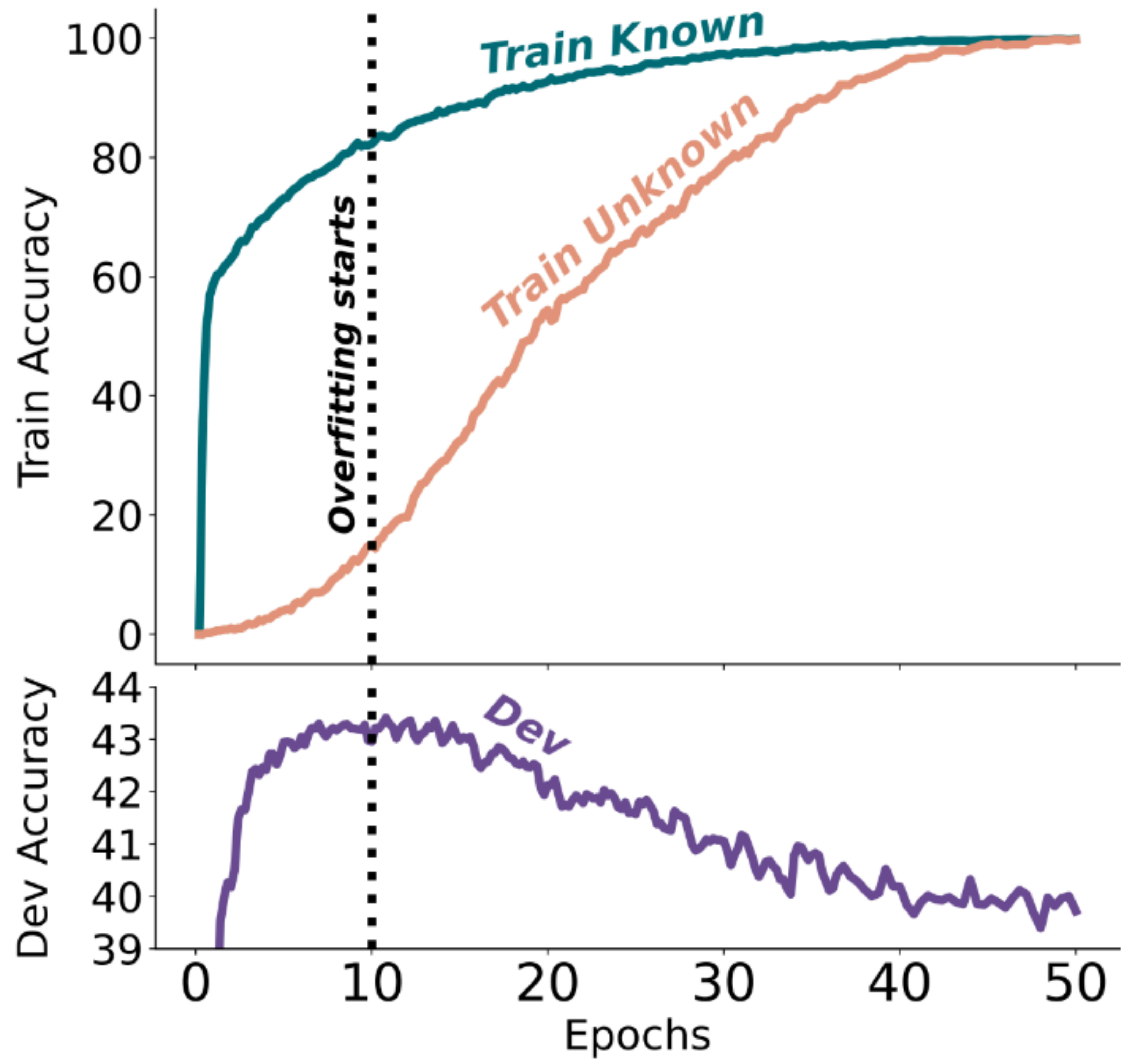


Figure 1: Train and development accuracies as a function of the fine-tuning duration, when fine-tuning on 50% `Known` and 50% `Unknown` examples. `Unknown` examples are fitted substantially slower than `Known`. The best development performance is obtained when the LLM fits the majority of the `Known` training examples but only few of the `Unknown` ones. From this point, fitting `Unknown` examples reduces the performance.

Unfamiliar Finetuning Examples Control How Language Models Hallucinate (https://arxiv.org/pdf/2403.05612v1)

A Closer Look at the Limitations of **Instruction Tuning** (https://arxiv.org/pdf/2402.05119)

Does Fine-Tuning LLMs on New Knowledge Encourage Hallucinations? (https://arxiv.org/pdf/2405.05904)