Finetuning Language Models

Jie Tang
Department of Computer Science & Technology
Tsinghua University

What is Finetuning

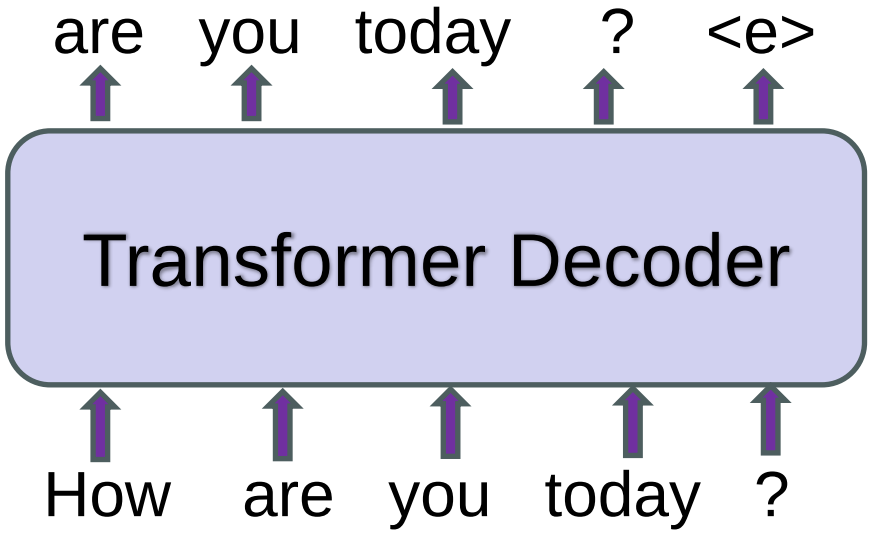
- Pretraining:
  - Train model from scratch on large unsupervised corpus
- Finetuning:
  - Start from a pretrained base model and finetune model parameters with labeled data for downstream tasks

Task	Dataset	GPT-3 Few-shot	GPT-3 Finetuned
Q&A	SQuAD V2 (F1)	69.8%	88.4%
Textual Entailment	RTE (Acc)	69%	85.4%
NL2QL	WikiSQL (Acc)	20%	73%
	Spider (Acc)	18%	62%

The Paradigm of Pre-train & Fine-tune

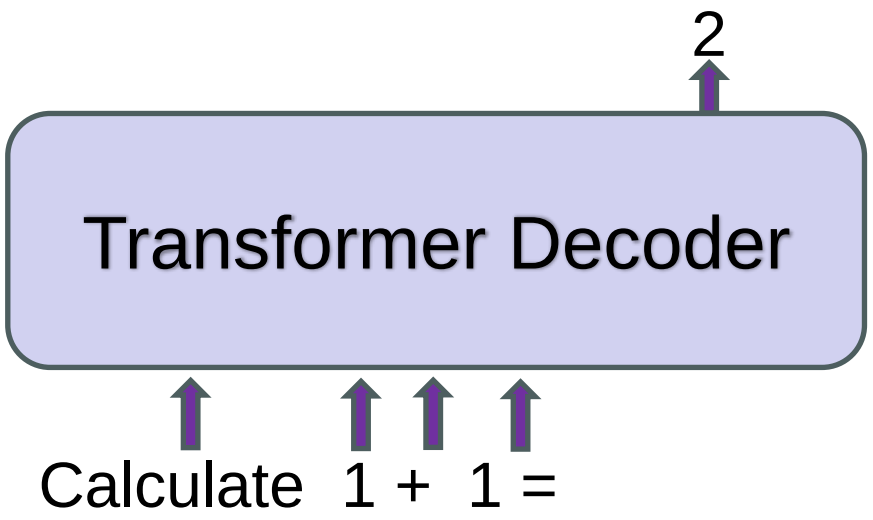
Pretrain

Abundant unsupervised corpus



Finetune (on specific task)

Adapt to unseen various tasks



Train Objective: $\max \sum \log P_\Theta(x)$

$\max \sum \log P_\Theta(y|x)$

Motivation: Why Finetuning Language Models

Pretraining "memorizes" data and knowledge but cannot be directly applied to solve specific tasks.

- Pretraining each model on each task is expensive
- Pretrained models have learned a wide variety of things about the statistical properties of language. Finetuning can help language models adapt to specific tasks

Input: *Subject**:** Win a million dollars NOW!! Exclusive deal for YOU only, don't miss this chance.*
From pretrained-model: *Unlock the extraordinary possibility to change your life within seconds.* (continue writing after the input)
True Label: *spam*

Motivation: Why Finetuning Language Models

- Prompt engineering is also expensive
  - very long prompt with multiple examples
- Better performance with fine tune
- Personalized service
  - with a light-weight fine-tuned model
- Data security
  - do not want to share private data

Motivation: Why Finetuning Language Models

In-context learning can also help language models adapt to specific tasks, but there are still some disadvantages:

– Prompt engneering is sensitive to prompts

Prompt	P@1
[X] is located in [Y]. *(original)*	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? In [Y].	51.08

– Prompts increases the length and burden of inference

```
Translate English to French:
cheese =>
```

No-prompt

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

With-prompts

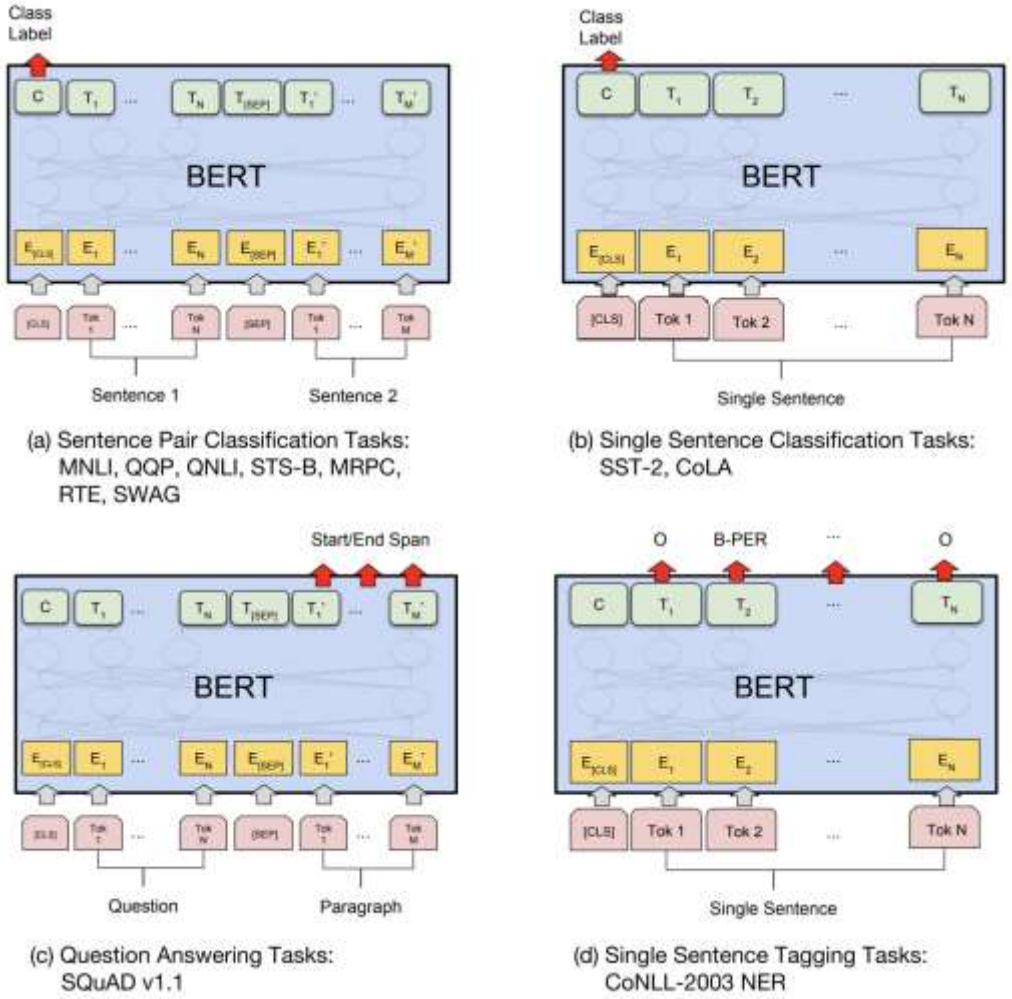
Motivation: Why Finetuning Language Models

- Personalized services require user-customized data, some of which do not exist in pre-training data
- Finetuning generally achieves better performance than prompting

Task	Dataset	GPT-3 Few-shot	GPT-3 Finetuned
Q&A	SQuAD V2 (F1)	69.8%	88.4%
Textual Entailment	RTE (Acc)	69%	85.4%
NL2QL	WikiSQL (Acc)	20%	73%
	Spider (Acc)	18%	62%

Finetuning Case

- Early Finetuning on LM -- BERT:



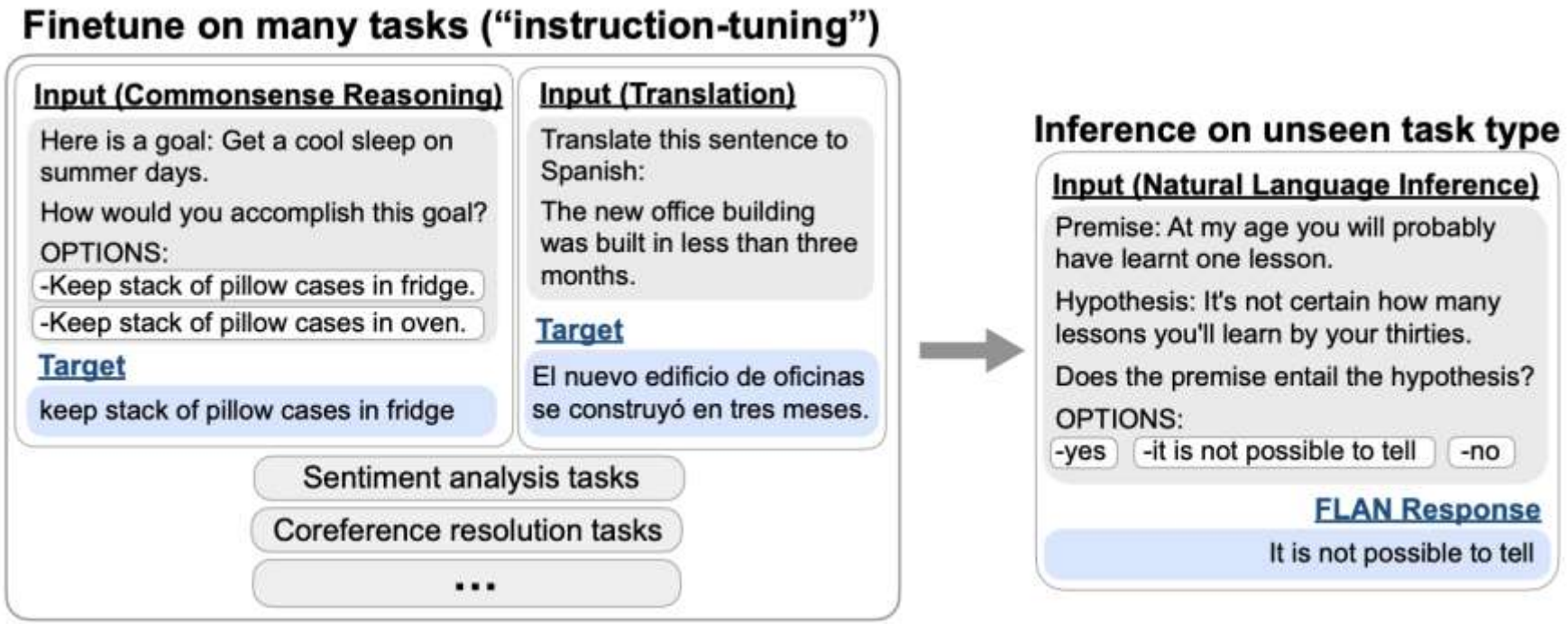
– Plug in task-specific inputs and outputs into BERT and finetune all the parameters end-to-end
– Aimed at increasing model's ability on certain task
– Finetuning BERT on each task only requires a few GPU hours

Devlin J, et al. Bert: Pre-training of deep bidirectional transformers for language understanding

Finetuning Case

- Instruction tuning on GPT:

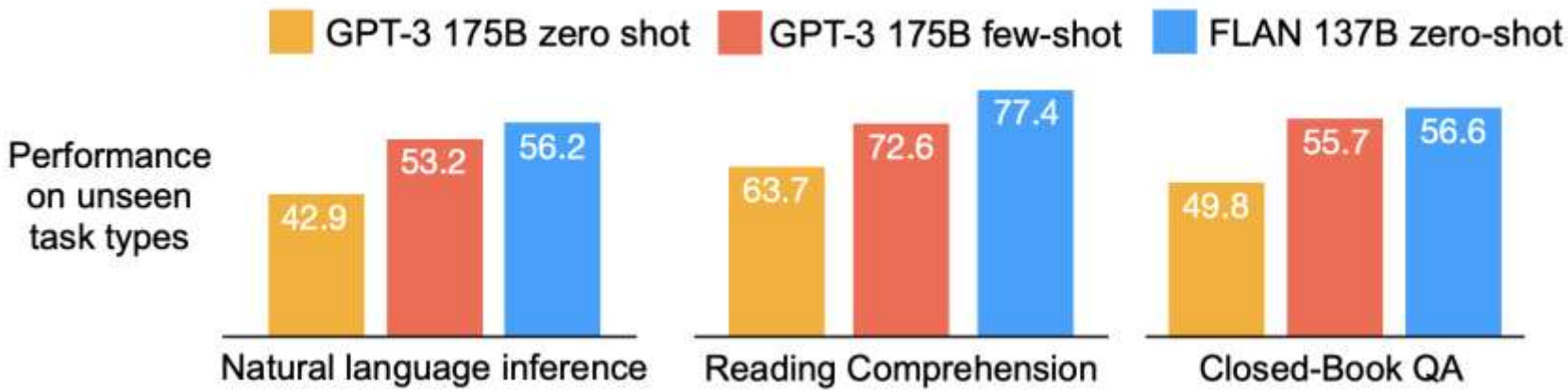
Take a 137B parameter pretrained language model and instruction tune it on over 60 NLP datasets verbalized via natural language instruction templates. Get the **F**inetuned **La**nguage **N**et, referred to as **FLAN**.



Jason Wei, et al. Finetuned language models are zero-shot learners

Finetuning Case

- Instruction tuning on GPT:
  - Finetune the language model on many tasks simultaneously described via instructions
  - The model will learn to follow instructions, generalizes to unseen tasks in zero-shot.



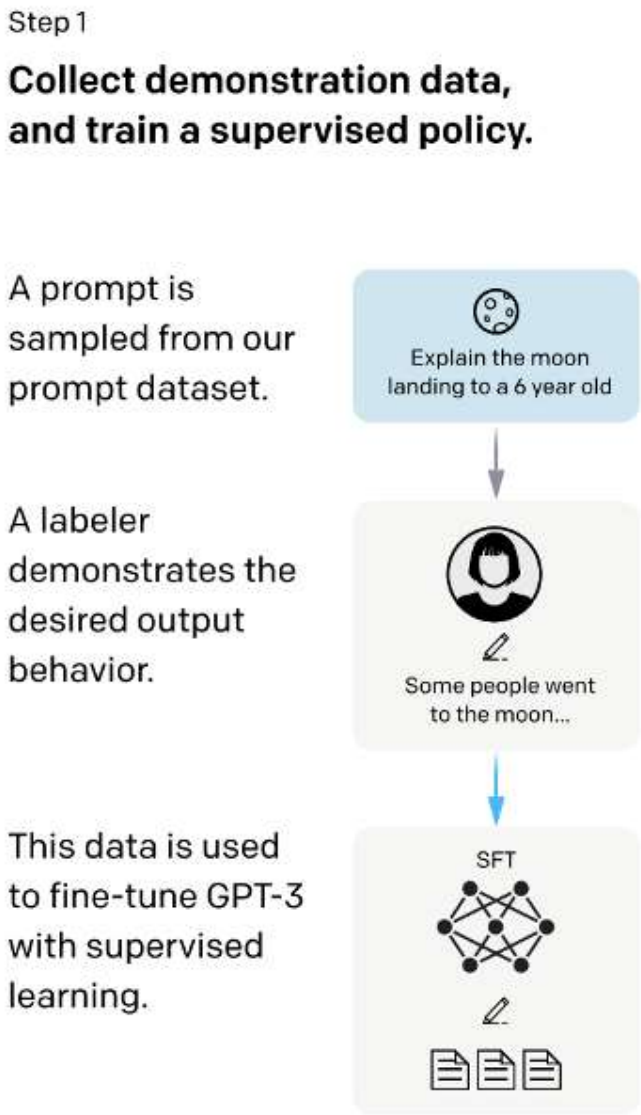
Jason Wei, et al. Finetuned language models are zero-shot learners

Finetuning Case

- Supervised Fine-Tuning Model (SFT)

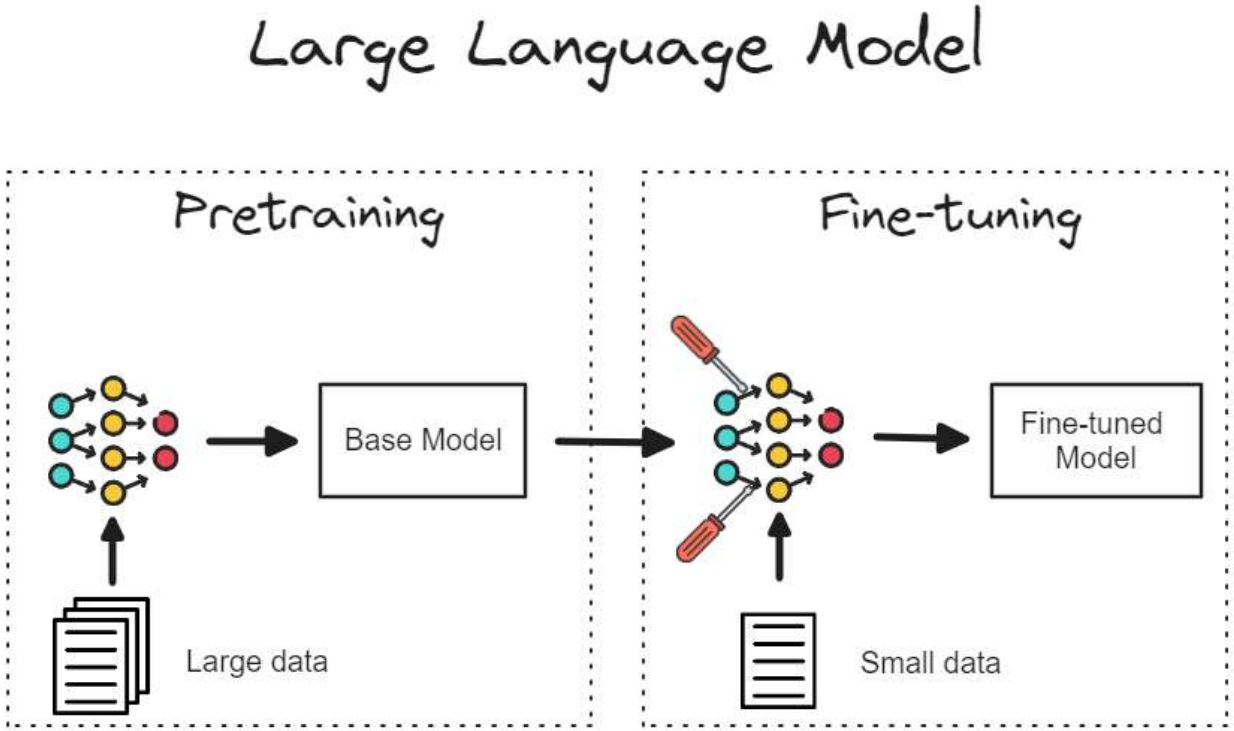
Example: 1-st stage of Instruct GPT:

  - Initiated by recruiting 40 contractors through a selective screening process
  - Compiled a dataset of *human-written demonstrations of the desired output behaviour* on prompts (mostly English) submitted to the OpenAI API and some labeler-written prompts
  - Used this dataset as source of supervised learning training



Why Finetuning Language Models Works

Pretrained models have learned a wide variety of knowledge. Finetuning can help language models concentrate on specific domain of knowledge



Parameter-Efficient Finetuning with Language Models

Motivation:The Cost of Finetuning

- Try to finetune GPT-3 with 175B parameters
  - Denote model parameters as Θ
  - float16 for model weights $\Theta_{\text{model}} = \Theta$
  - Float16 for gradients $\Theta_{grad} = \Theta$
  - Adam optimizer with float32 $\Theta_{optim} = 4\Theta$
- Full finetuning: $6\Theta = \Theta_{model} + \Theta_{grad} + \Theta_{optim}$
  - $\rightarrow$ **$\sim 2100G$ memory**
- Requires around 32×A100(80G) to start the training

Do we need to fine-tune all parameters?

- Large Scale
  - is necessary for training: better and more generalized optimization
  - may not be necessary for inference: a few of parameters may be enough for downstream tasks

- In language models, how many parameters we need to change for task transfer?

Intrinsic Dimensionality in Finetuning

- Pre-trained language models have a low “instrisic dimension” and can still learn efficiently despite a random projection to a smaller subspace
- Project parameters using an arbitrary random projection onto a much smaller space
  - $\Theta^D = \Theta_0^D + P(\Theta^d),\ d \ll D$
  - $P: \mathbb{R}^d \rightarrow \mathbb{R}^D$
- If we reach a satisfactory solution, we say the dimensionality of that subspace is the intrinsic dimension

- “For example, by optimizing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC.”

- Armen Aghajanyan et al, Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. ACL 2021. Meta.
- Chunyuan Li, Heerad Farkhoor, Rosanne Liu, Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. ICLR’18

Recall Dimension Reduction

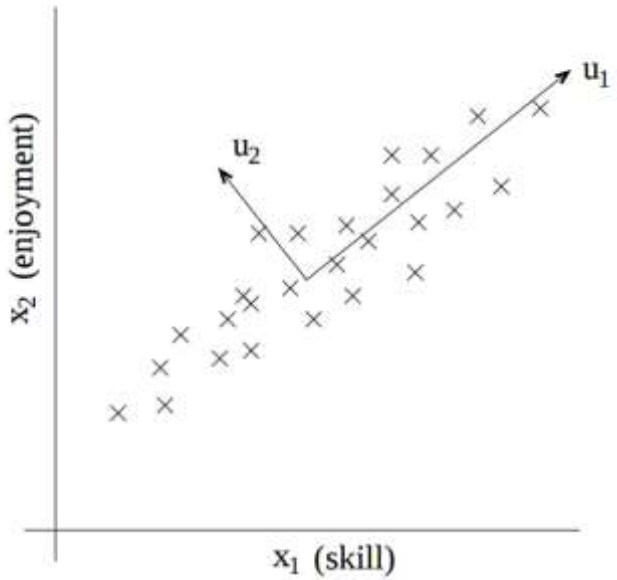
- Dimension Reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
  - Given some data $X \in R^{N \times m}$ (m might be huge), we would to find a new representation $\tilde{X} \in R^{N \times K}, K \ll$m.
  - Application: data visualization, data compression.

Projection of the MNIST dataset via PCA

PCA: Principal Components Analysis

- General Idea:
  - Given data points in *m*-dimensional space
  - Project into lower dimensional space while preserving as much information as possible --- minimize the square error for representing the original data

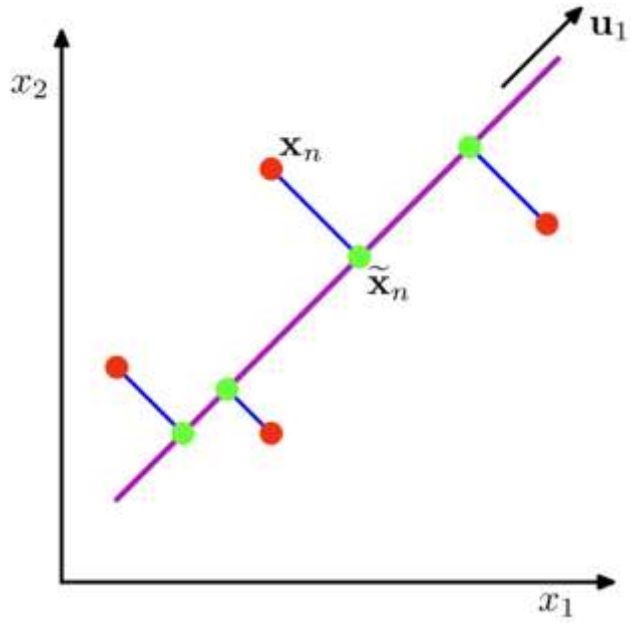
For example, given *N* data point $\{x_i; i=1,\ldots, N\}$, x_i \in $\mathbf{R}^m$, represent all data in two dimensions.



Principal Component Analysis (PCA)

- Principal Component Analysis is defined as an orthogonal linear transformation.
  - Linear transformation

$$\tilde{x}_n = x_n^T u_1, \ \left\| u_1 \right\|_2 = 1$$



- What is a good transformation?
  - Find directions of maximum variance.
    - The greatest variance of the data comes to lie on the first direction (first principal component), the second greatest variance on the second direction, and so on.

How many components?

- More generally, if we wish to project our data onto a k-dimensional subspace, we should choose $v_1,\cdots,v_k$, but how many should be enough?

- For each singular vector, check how much data variance is represented by this singular vector.
- Take enough many eigenvectors to cover e.g., 80-90% of the variance

$$\frac{\sum_i^k \sigma_i}{\sum_i^m \sigma_i} = 80\%$$

PCA Limitations

Problem

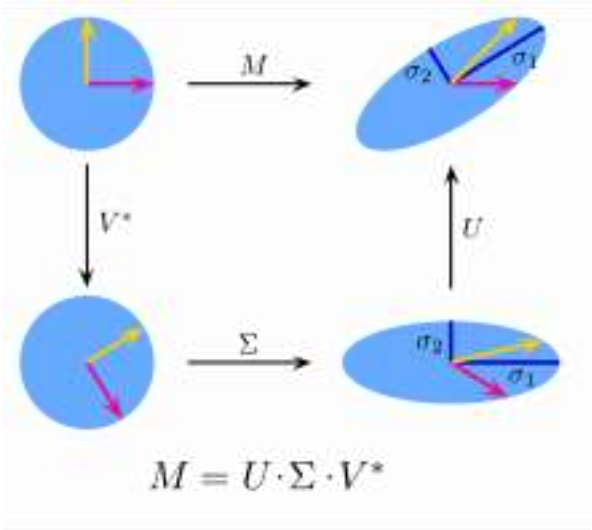
- The covariance matrix Σ is huge —$O(m^2)$, if the dimension of the original data is large

Singular Value Decomposition

- Definition: Singular Value Decomposition: For $X \in R^{n \times m}$, there exists a factorization of the form

$$X = UDV^T$$

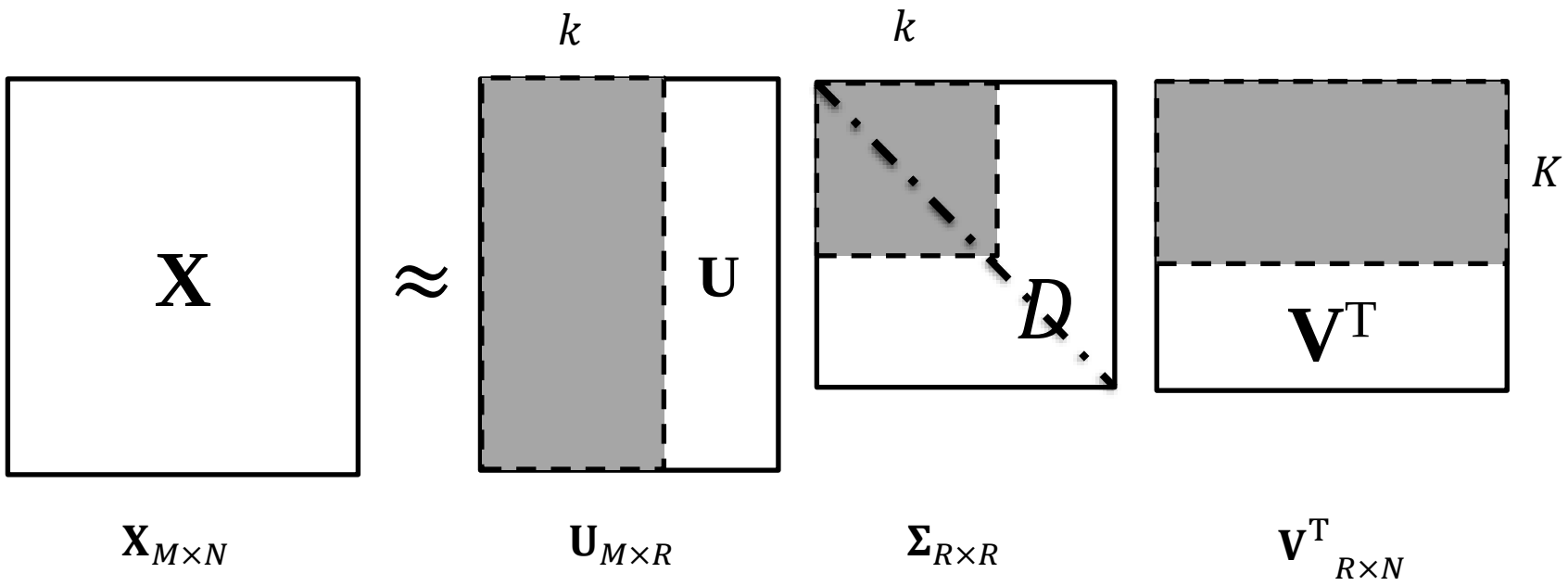
with U an $n \times k$ orthogonal matrix, D an $r \times k$ diagonal matrix of singular values, and V an $m \times k$ orthogonal matrix.



Truncated SVD

- TSVD factorizes the data matrix into the three matrices U, D, and V with the *k*-dimension.

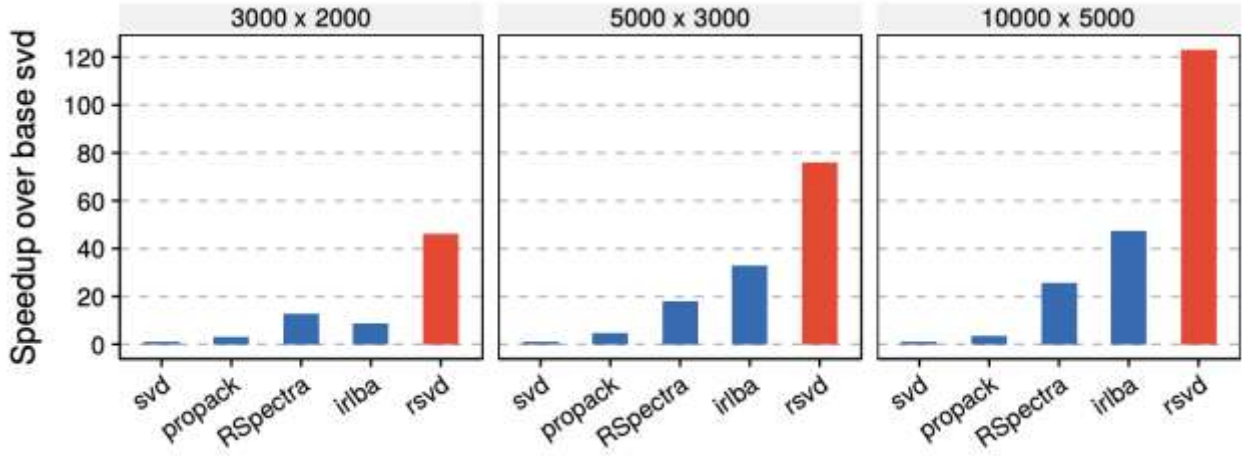
$$X \approx X_k = U_k D_k V_k^{\mathrm{T}}$$



The complexity is $O(\min(m^2n, mn^2))$, how to speed up?

Fast Random tSVD

- Randomness as a computational strategy
- 10^2 speedup!
- Brief Algorithm:



  - Stage A: Construct a low dimensional subspace to approximate the column space of X, with $X \approx QQ^TX$, Q is mxk, k<<min(m, n)
    - $Y=X\Omega$, Ω is randomly generated by gaussian;
    - $Y=QR$
  - Stage B: $B=Q^TX$, do tSVD for $B=SDV^T$, $QS=U$

$$X \approx Q \quad B$$
$$nxm \quad mxk \quad kxn$$

Back to Intrinsic Dimensionality

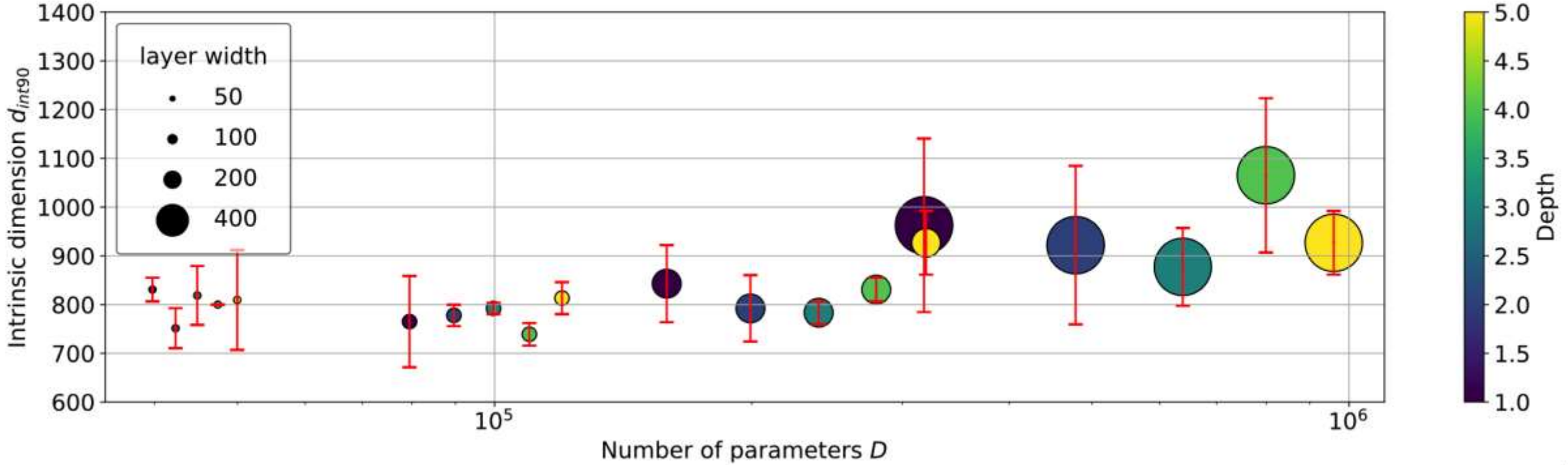
- Measured d_{int90} on various supervised and reinforcement learning problems.

Table 1: Measured d_{int90} on various supervised and reinforcement learning problems.

Dataset	**MNIST**		**MNIST (Shuf Pixels)**		**MNIST (Shuf Labels)**
Network Type	**FC**	**LeNet**	**FC**	**LeNet**	**FC**
Parameter Dim. D	199,210	44,426	199,210	44,426	959,610
Intrinsic Dim. d_{int90}	750	290	750	1,400	190,000

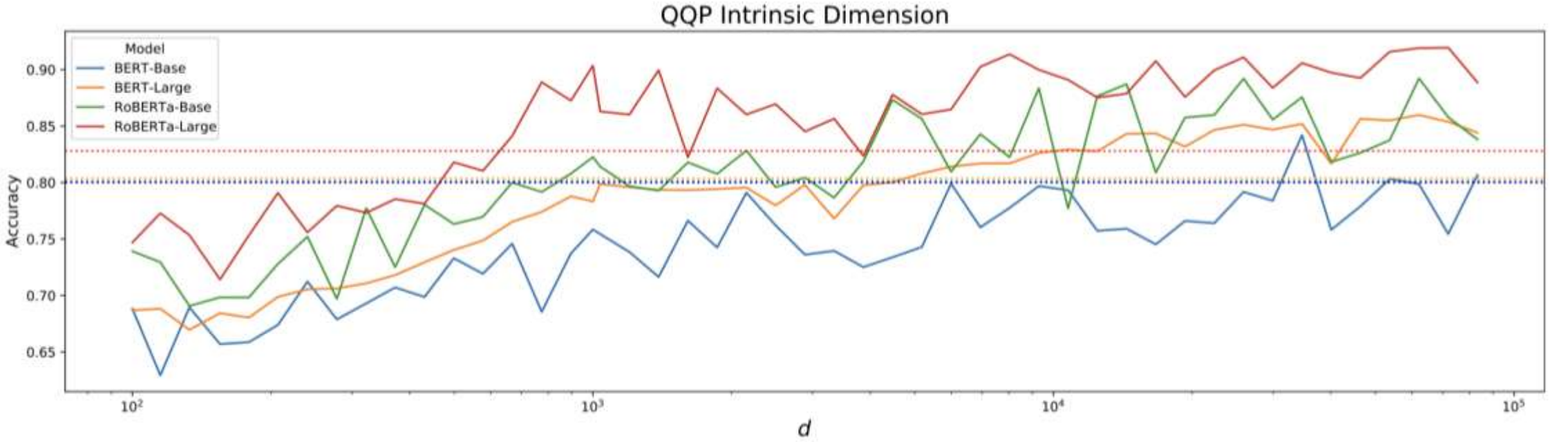
...	**CIFAR-10**		**ImageNet**	**Inverted Pendulum**	**Humanoid**	**Atari Pong**
...	**FC**	**LeNet**	**SqueezeNet**	**FC**	**FC**	**ConvNet**
...	656,810	62,006	1,248,424	562	166,673	1,005,974
...	9,000	2,900	$>$ 500k	4	700	6,000

Intrinsic dimension d_{int90} vs number of parameters D



Intrinsic Dimensionality in Finetuning

- 90% of the full fine-tuning solution using roughly 800 parameters
  - Incredible low dimensionality of viable solutions



- Finetuning a small-portion-amount of parameters can also specialize the pretrained to downstream tasks with almost equal performance.

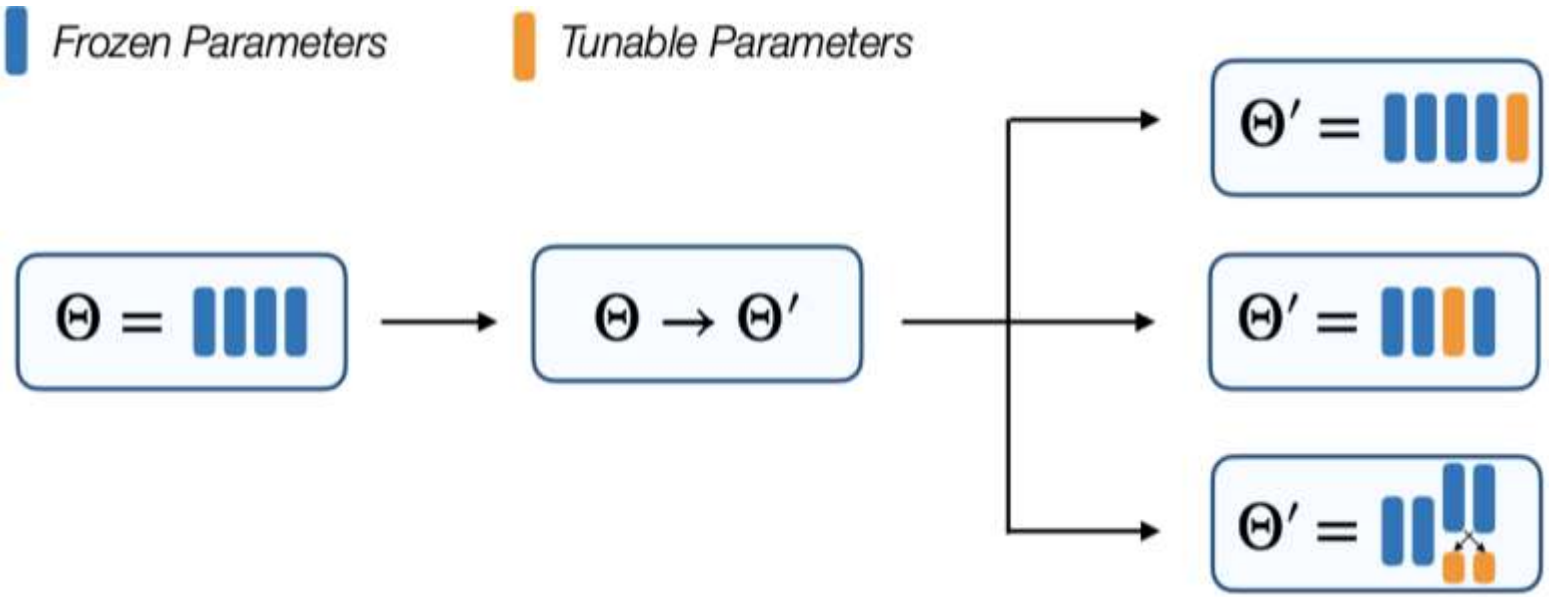
Armen Aghajanyan et al, Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. ACL 2021. Meta.

Do we need to fine-tune all parameters?

- Full-finetuning learns “weight updating”
  - $\max_{\Theta} \sum_{D} \log P_{\Theta}(y|x)$
  - $\Theta \leftarrow \Theta_0 + \Delta\Theta$

- Parameter efficient tuning
  - Learn the “updates” $\Delta\Theta$ directly rather than modifying $\nabla\Theta_0$
  - Directly learn the weight diff $\Delta\Theta$
    - $\max_{\Delta\Theta} \sum_{D} \log P_{\Theta_0+\Delta\Theta}(y|x), \quad |\Delta\Theta| \ll |\Theta_0|$
  - Learn to optimize prompts
    - $\max_{\Delta x} \sum_{D} \log P_{\Theta_0,\Delta x}(y|x,\Delta x), \quad |\Delta x| \ll |\Theta_0|$

Parameter Efficienct Tuning



Weight updating

1. Injection-based tuning: modify part of original paramaters
2. Introduce additional paramaters as updates $\Delta\Theta$
3. Input-based optimization, tuning prompts

Picture from Ding et al, Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

Paradigms: Parameter Efficienct Tuning

- Solution 1: Updating Part of Model Weights
  - Adapter tuning
  - Low rank adaptation (LoRA)
  - Quantization + LoRA (QLoRA)

- Solution 2: Optimizing Prompts
  - Prefix tuning
  - P-tuning

Adapter Tuning

- Fixed the original LM, and introduce new adapter modules (with a few parameters) to LLM

Adapter modules

– Only 3.6% additional parameters are needed for tuning

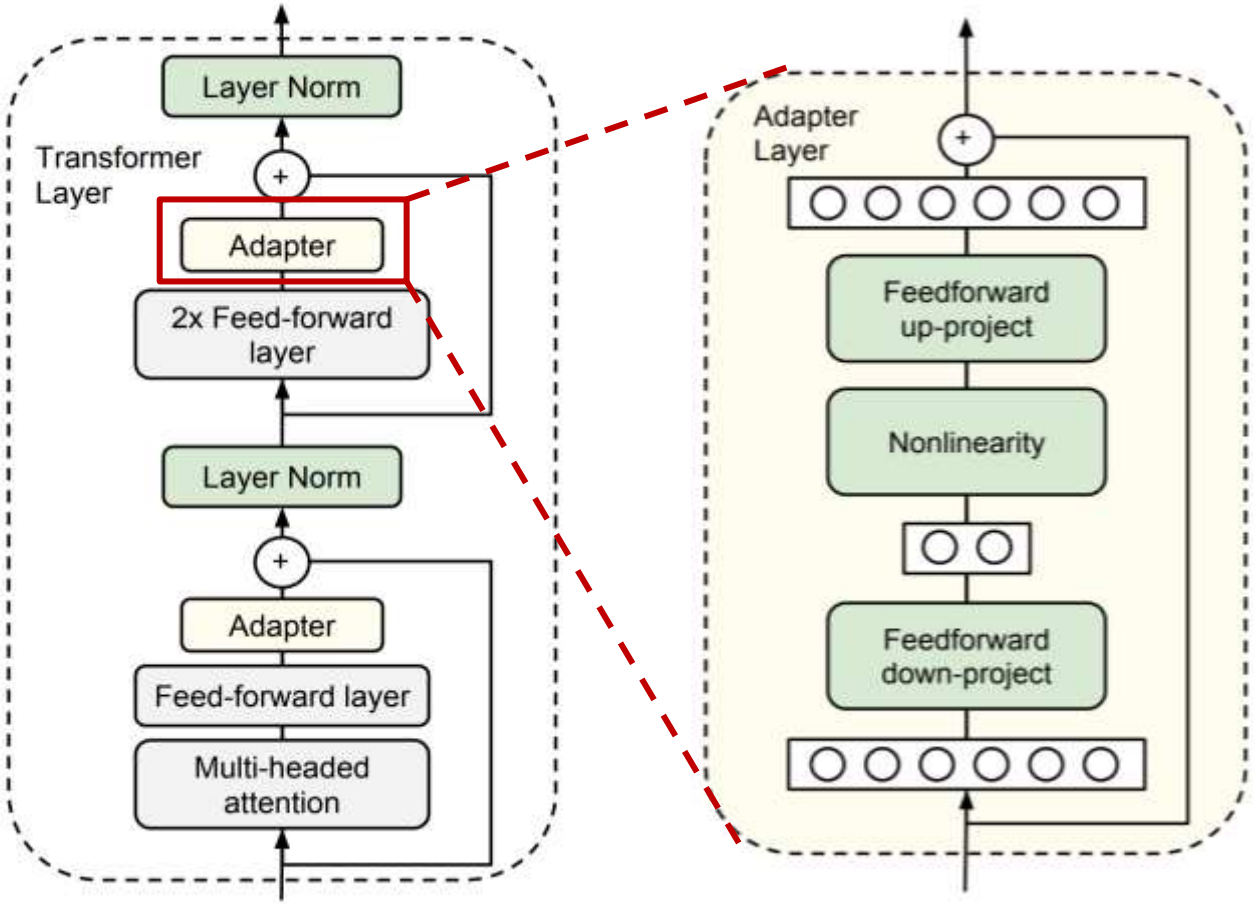
– Few parameters -> low memory/compute overheads

– Identity initialization -> stable training

1. Google. Parameter-Efficient Transfer Learning for NLP, ICML 2019.

Adapter Tuning

- Original Attention:
  - $H = Attn(X)W_o + b_o$
- FFN
  - $\mathrm{F} = \sigma(HW_1 + b_1)W_2 + b_2$
- Attention with Adapter
  - $H' = HW_{a1} + b_{a1}$
- FFN with Adapter
  - $F' = F(H')W_{a2} + b_{a2}$



- **Adapter modifies transformer architecture**

Adapter Tuning Results

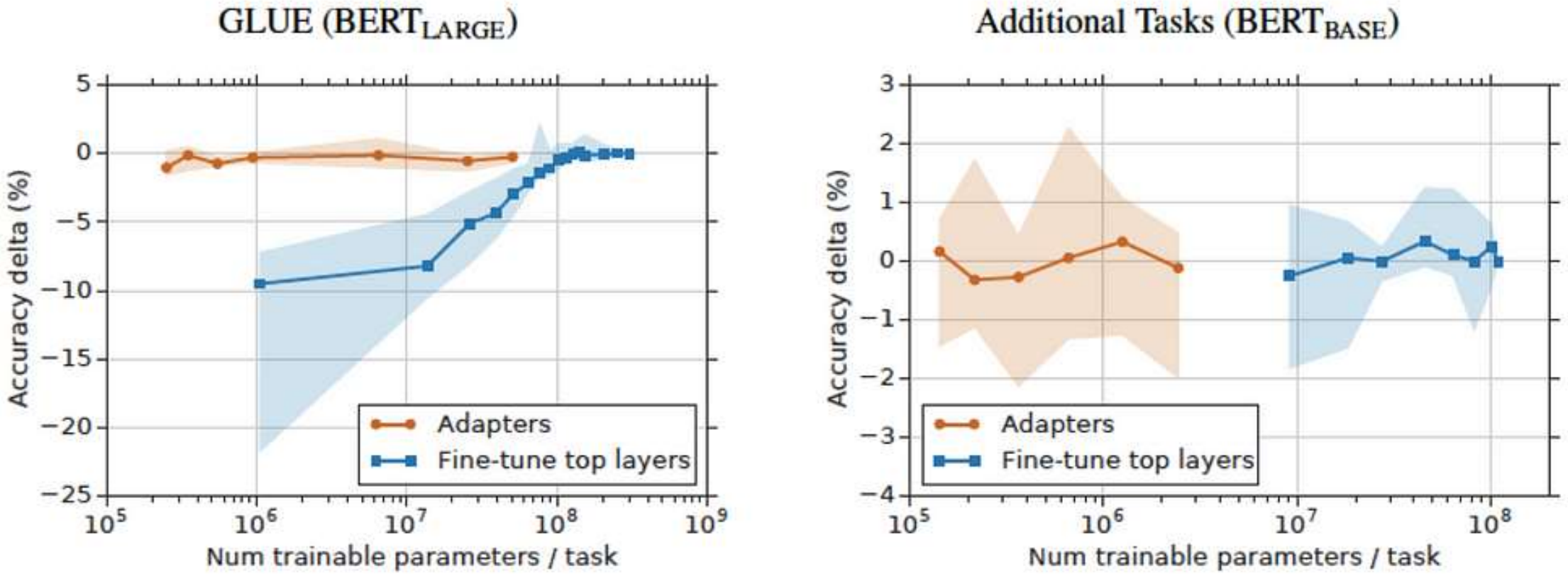
- fine-tuning requires 9×the total number of BERT parameters. In contrast, adapters require only 1.3× parameters.

Parameter-Efficient Transfer Learning for NLP

	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_mm	QNLI	RTE	Total
BERT_LARGE	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Adapter Tuning Results

Parameter-Efficient Transfer Learning for NLP



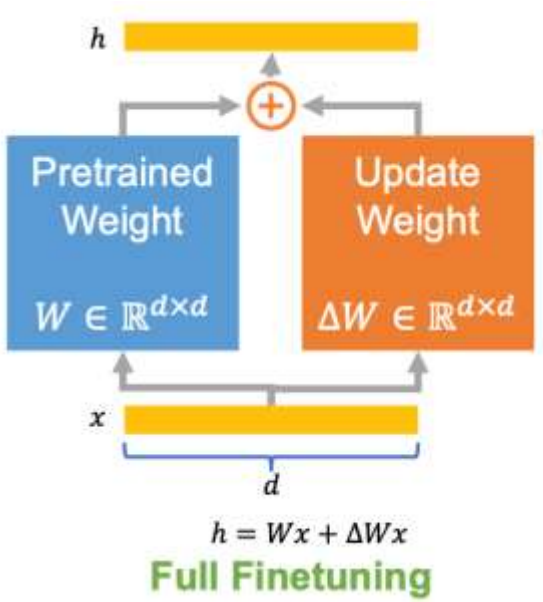
Low-rank Adaptation

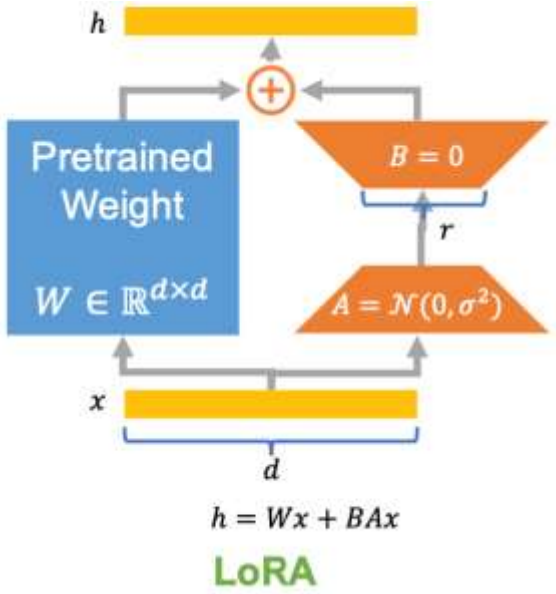
- Motivation
  - According to the previous section “Intrinsic Dimensionality in Finetuning”, Aghajanyan et al. (2020) shows that the pre-trained language models have a low "instrisic dimension" and can still learn efficiently despite a random projection to a smaller subspace.
  - Inspired by this, Low-rank Adaptation (LoRA) hypothesize the updates to the weights also have a low "intrinsic rank" during adaptation. This also explains why LoRA works.

Hu et al. LORA: Low-Rank Adaption of Large Language Models. ICLR’22. Microsoft.

Low-rank Adaptation

- Low-rank adaptation (LoRA) for weight matrices
  - Keep original architecture while introduce an additional equivalent "branch"
  - Make the branch as small as possible yet equivalent to original weight



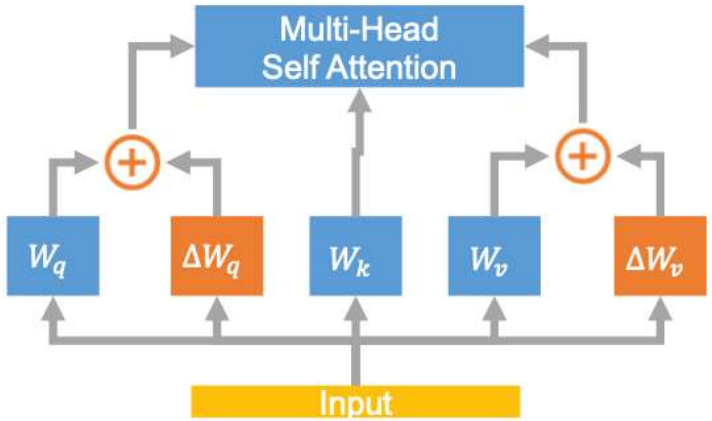


Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

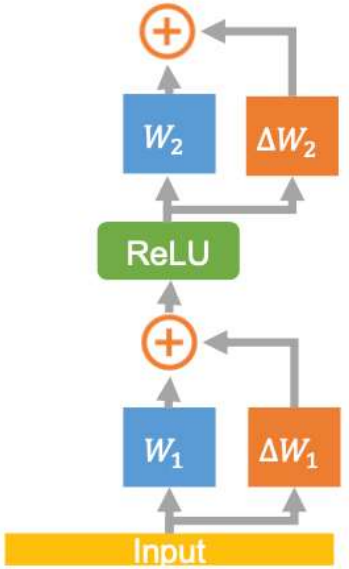
Low-rank Adaptation

- Decompose weight matrix to low-rank matrix
  - $W_0 x + \Delta W x = W_0 x + BAx$
  - $W_0 \in \mathbb{R}^{d \times d}$
  - $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times d}$
  - $r \ll d \longrightarrow |B|, |A| \ll |W_0|$

• Apply LoRA to Attention



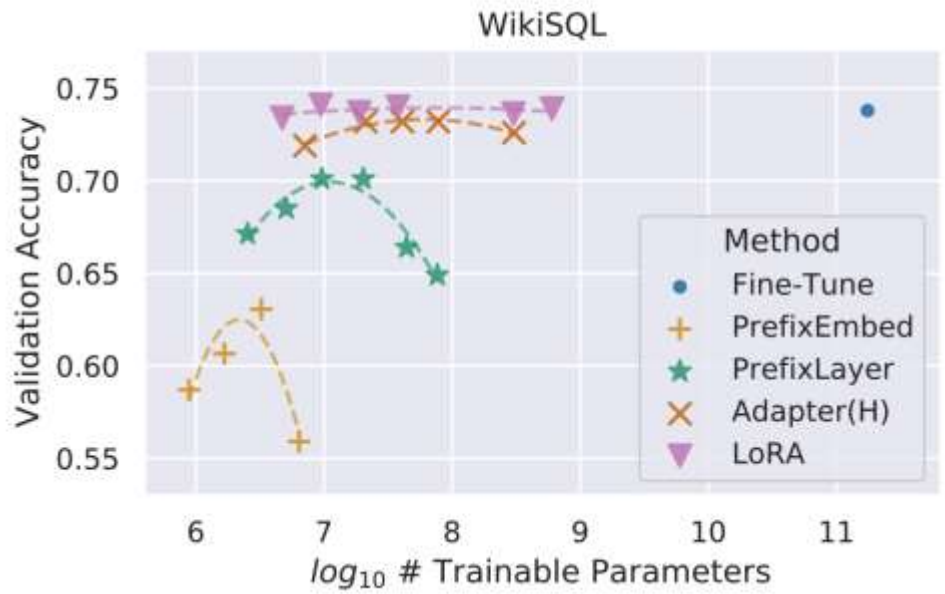
• Apply LoRA to MLP layer



Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

Low-rank Adaptation

- LoRA exhibits better scalability, stability, and task performance across different ranks and trainable parameters



	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL($\pm$0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm$0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

Low-rank Adaptation

- LoRA is proven to be one of the most effective parameter efficient tuning methods and widely adopted in open-source repos

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 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	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

Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

Low-rank Adaptation

- No additional inference latency
- Adapter / Reparameterization introduce extra latency

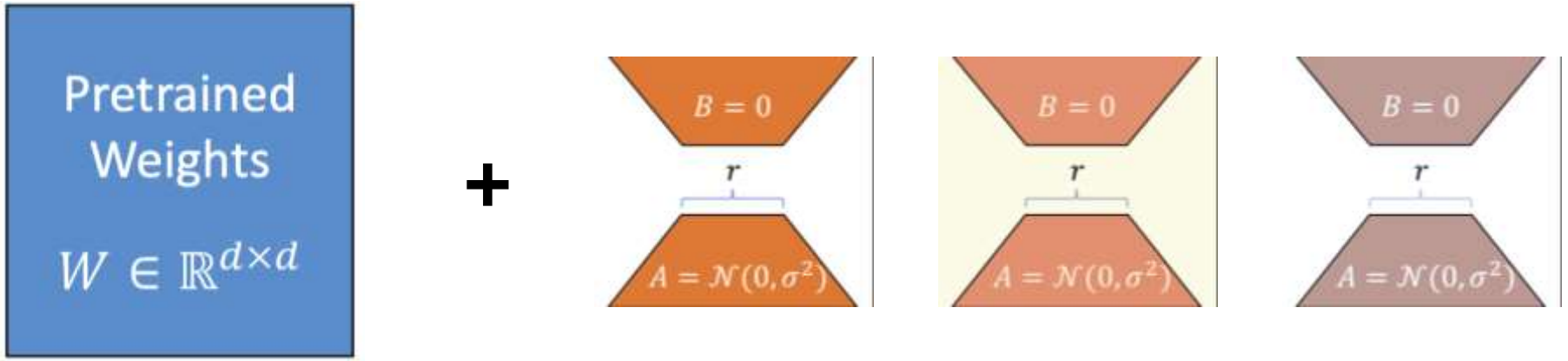
Batch Size Sequence Length $\|\Theta\|$	32 512 0.5M	16 256 11M	1 128 11M
Fine-Tune/LoRA	1449.4±0.8	338.0±0.6	19.8±2.7
$\text{Adapter}^{\text{L}}$	1482.0±1.0 (+2.2%)	354.8±0.5 (+5.0%)	23.9±2.1 (+20.7%)
$\text{Adapter}^{\text{H}}$	1492.2±1.0 (+3.0%)	366.3±0.5 (+8.4%)	25.8±2.2 (+30.3%)

Table 1: Infernece latency of a single forward pass in GPT-2 medium measured in milliseconds, averaged over 100 trials. We use an NVIDIA Quadro RTX8000. "$|\Theta|$" denotes the number of trainable parameters in adapter layers. $\text{Adapter}^{\text{L}}$ and $\text{Adapter}^{\text{H}}$ are two variants of adapter tuning, which we describe in Section 5.1. The inference latency introduced by adapter layers can be significant in an online, short-sequence-length scenario. See the full study in Appendix B.

Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

Low-rank Adaptation

- Easy to storing and deploying
- Dynamically switch among different tasks
  - Can learn a different set of $\Delta\Theta$



Pretrained Model Weights | Task A | Task B | Task C

Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

Low-rank Adaptation

- Applicable Scene
  - Limited computing source
  - Build many small LoRA modules for different tasks
- Inapplicable Scene
  - Tasks that require models with high parameter capacity to capture complex features
  - Extremely large amount of data

Hu et al. LORA: Low-Rank Adaption of Large Language Models. 2021

QLoRA

- QLoRA: less memory required
  - Example:

    finetune a 65B parameter model on a single 48GB GPU (originally 780GB) while preserving full 16-bit finetuning task performance

Model	Size	Elo
GPT-4	-	1348 ± 1
Guanaco 65B	41 GB	1022 ± 1
Guanaco 33B	21 GB	992 ± 1
Vicuna 13B	26 GB	974 ± 1
ChatGPT	-	966 ± 1
Guanaco 13B	10 GB	916 ± 1
Bard	-	902 ± 1
Guanaco 7B	6 GB	879 ± 1

Tim Dettmers et al. QLORA: Efficient Finetuning of Quantized LLMs. 2023

QLoRA

- Quantization: converting a data type into fewer bits
  - Example:

    FP32 tensor -> Int8 tensor with range [−128, 127]

    - Quantization:

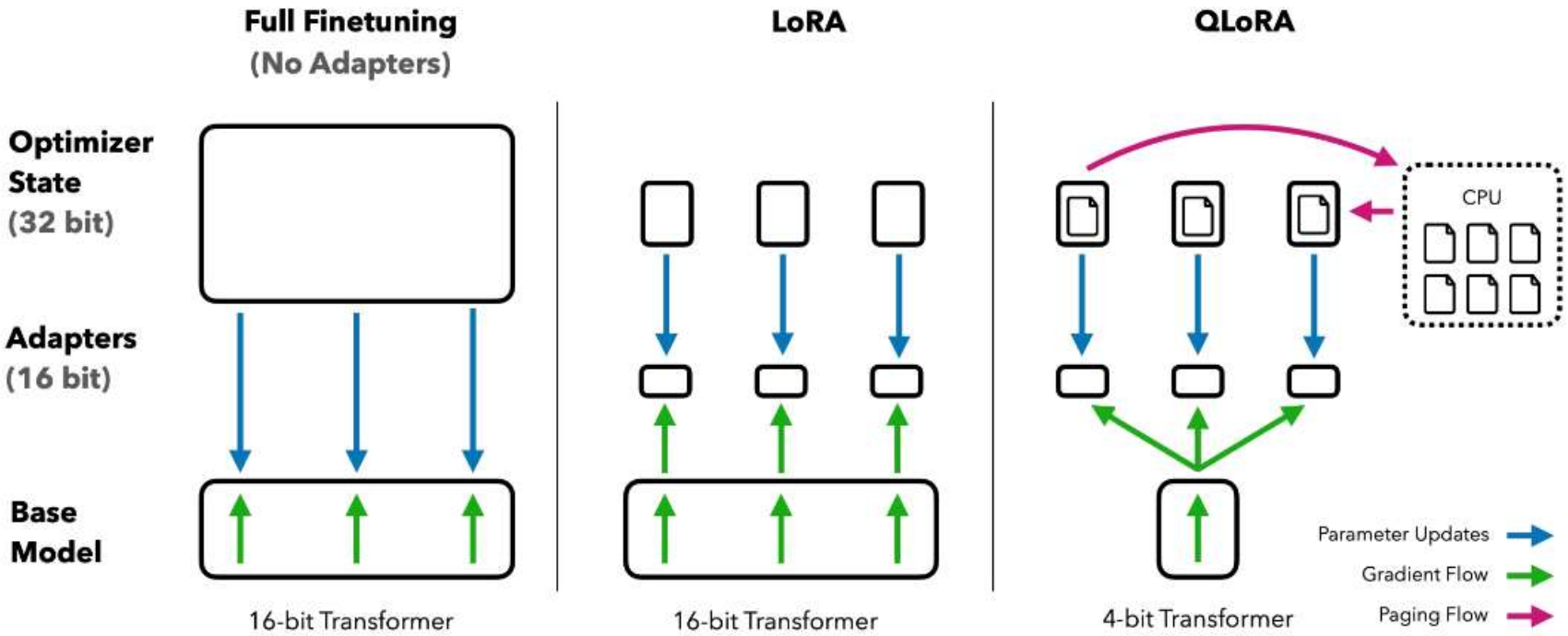
$$\mathbf{X}^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(\mathbf{X}^{\text{FP32}})}\mathbf{X}^{\text{FP32}}\right) = \text{round}(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}})$$

    - Dequantization:

$$\text{dequant}(c^{\text{FP32}}, \mathbf{X}^{\text{Int8}}) = \frac{\mathbf{X}^{\text{Int8}}}{c^{\text{FP32}}} = \mathbf{X}^{\text{FP32}}$$

Tim Dettmers et al. QLORA: Efficient Finetuning of Quantized LLMs. 2023

QLoRA



- Example: a linear layer

$$\underbrace{\mathbf{Y}^{\text{BF16}}}_{\text{16-bit activations}} = \mathbf{X}^{\text{BF16}}\text{doubleDequant}(c_1^{\text{FP32}}, c_2^{\text{k-bit}}, \underbrace{\mathbf{W}^{\text{NF4}}}_{\text{4-bit frozen weights}}) + \mathbf{X}^{\text{BF16}}\underbrace{\mathbf{L}_1^{\text{BF16}}\mathbf{L}_2^{\text{BF16}}}_{\text{16-bit LoRA weights}}$$

Tim Dettmers et al. QLORA: Efficient Finetuning of Quantized LLMs. 2023

QLoRA

- Save memory without sacrificing performance:

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

Parameter Efficienct Tuning v.s. Finetuning

- Speed up
  - do not need to calculate the gradient for the vast majority of the parameters.
    - Less computation cost in backward
  - Around 25% speedup during training on GPT-3 175B compared to full fine-tuning

Parameter Efficienct Tuning v.s. Finetuning

- Memory-efficient
  - Training setting:
    - float16 for model weights and gradients
    - Adam optimizer with float32
  - Full finetuning: $6\Theta = \Theta_{model} + \Theta_{\text{grad}} + 4\Theta_{\text{optim}}$
  - Parameter-efficient: $(1 + 6\delta)\Theta = \Theta_{\text{model}} + \delta\Theta_{pef} + \delta\Theta_{grad} + 4\delta\Theta_{optim}$
    - $\delta \sim 0.01$
- For GPT-3 175B
  - 2.1TB → ~350G

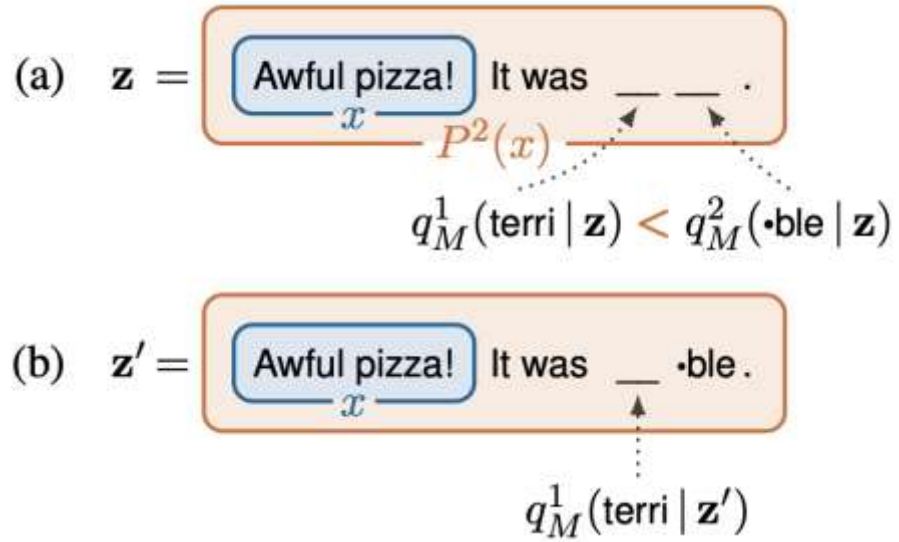
Paradigms: Parameter Efficienct Tuning

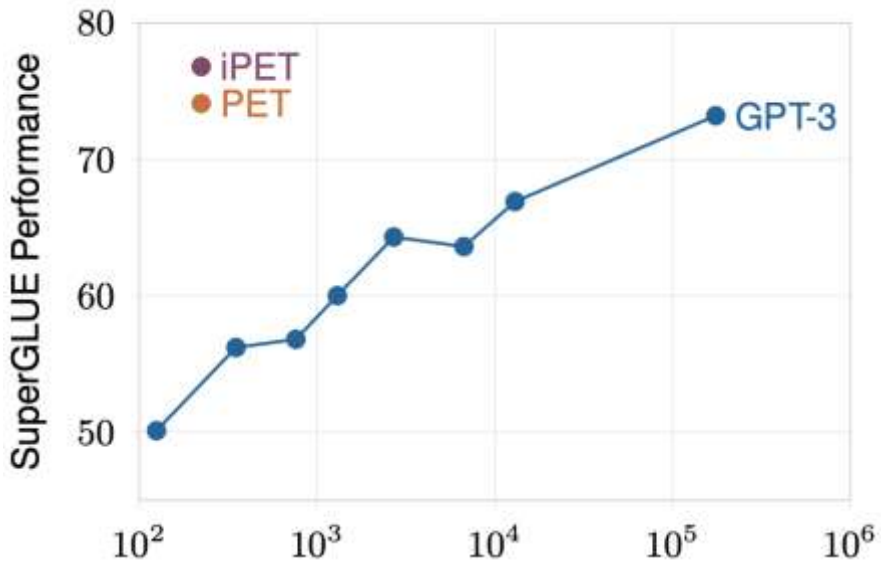
- Solution 1: Updating Part of Model Weights
  - BitFit
  - Adapter
  - Low rand adaptation (LoRA)

- Solution 2: Optimizing Prompts
  - Prefix tuning
  - P-tuning

Fine-tune with prompts

- Training model parameters is to adapt the model to downstream tasks, while training prompt is to adapt downstream tasks to the model.
- It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners (Schick, 2020)
  - Introducing prompts to the fine-tuning of smaller LMs



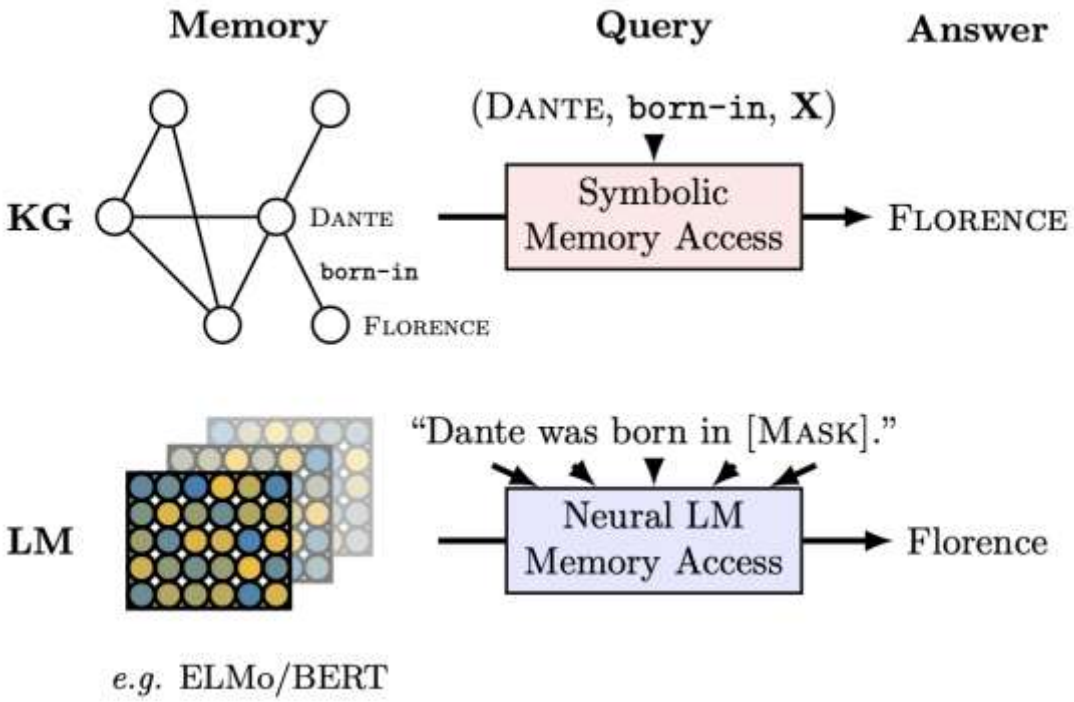


Motivation of Prompt Learning: Examples

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1][X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

Problem: Vulnerable Prompts

- Language Model as Knowledge Base
  - Human prompts are vulnerable
    - A single word's change in prompts results in drastic performance variation

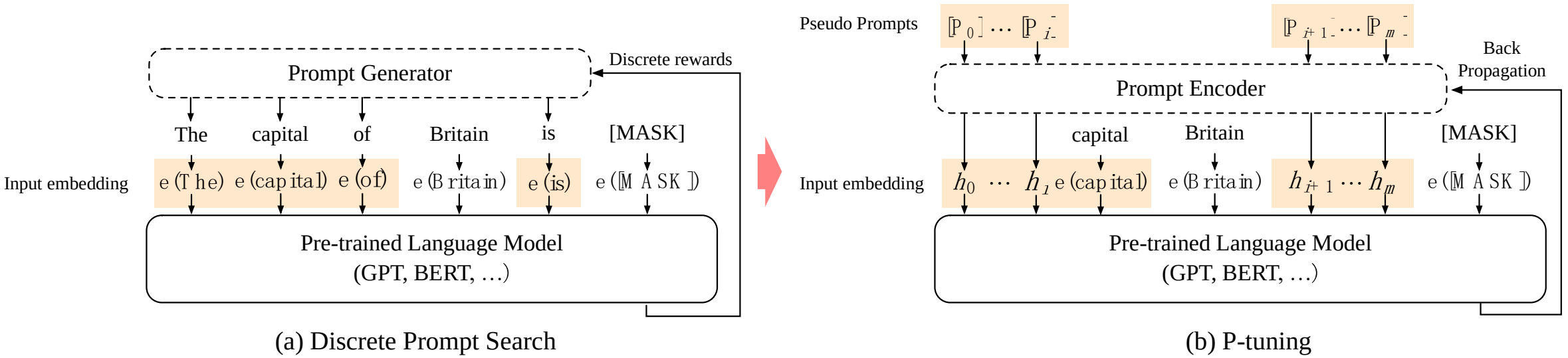


Prompt	P@1
[X] is located in [Y]. *(original)*	31.29
[X] is located in which country or state? [Y].	19.78
[X] is located in which country? [Y].	31.40
[X] is located in which country? In [Y].	51.08

Table 1: Case study on LAMA-TREx P17 with bert-base-cased. A single-word change in prompts could yield a drastic difference.

P-tuning

- Finding optimized prompt in the continuous space:
  - Fix LM parameters, add learnable vectors to input as soft prompts
  - train continuous prompt embeddings (i.e., soft prompts)



(a) Discrete Prompt Search

(b) P-tuning

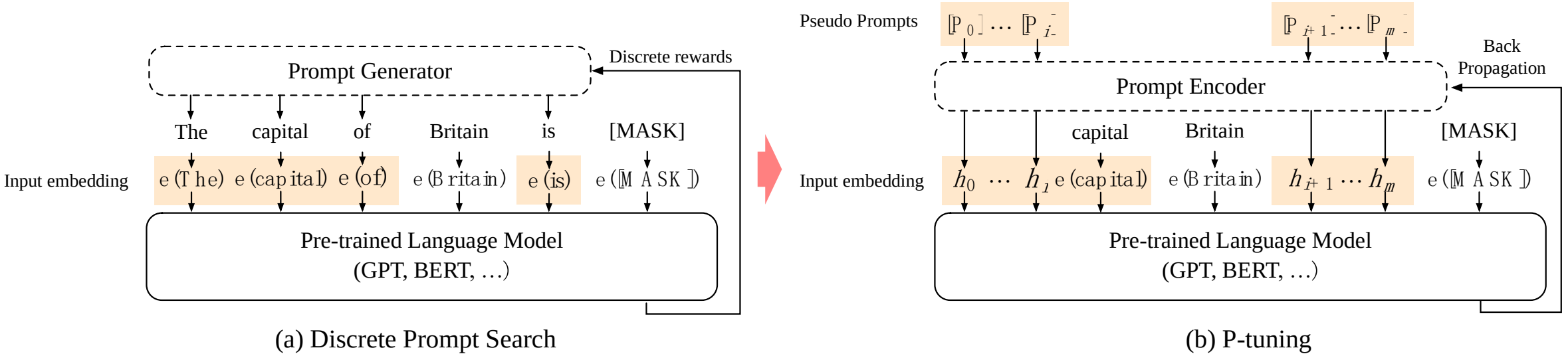
Liu X*, Zheng Y*, et al. GPT Understands, Too

P-tuning

- The prompt template for P-Tuning

$$T = \{[\mathrm{P}_{0:i}], \mathbf{x}, [\mathrm{P}_{(i+1):j}], \mathbf{y}, [\mathrm{P}_{(j+1):k}]\}$$

  - {(x,y)} is a labeled dataset. [Pi] is the i-th continuous prompt embedding.
  - A mapping function f maps trainable embeddings{Pi} to model inputs {hi}.
  - Finally, updating the embeddings {Pi} to optimize a task loss function.



(a) Discrete Prompt Search

(b) P-tuning

Liu X*, Zheng Y*, et al. GPT Understands, Too

P-tuning

- P-tuning
  - Significant improvement on LAMA by about 20 percent
  - LMs know more than we previously thought

Prompt type	Model	P@1
Original (MP)	BERT-base	31.1
	BERT-large	32.3
	E-BERT	36.2
Discrete	LPAQA (BERT-base)	34.1
	LPAQA (BERT-large)	39.4
	AutoPrompt (BERT-base)	43.3
P-tuning	BERT-base	48.3
	BERT-large	**50.6**

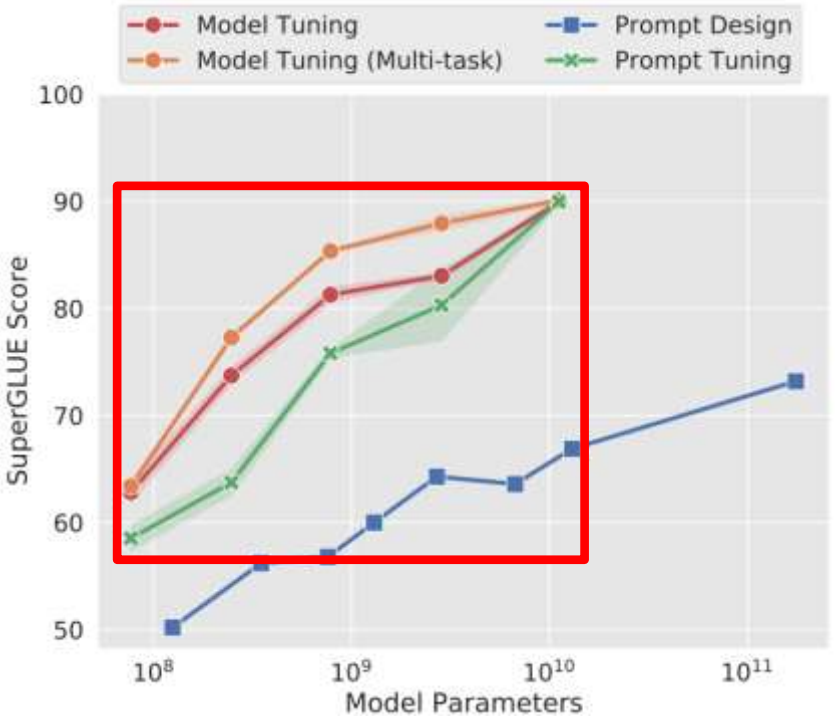
Model	MP	FT	MP+FT	P-tuning
BERT-base (109M)	31.7	51.6	52.1	52.3 (+20.6)
-AutoPrompt (Shin et al., 2020)	-	-	-	45.2
BERT-large (335M)	33.5	54.0	55.0	54.6 (+21.1)
RoBERTa-base (125M)	18.4	49.2	50.0	49.3 (+30.9)
-AutoPrompt (Shin et al., 2020)	-	-	-	40.0
RoBERTa-large (355M)	22.1	52.3	52.4	53.5 (+31.4)
GPT2-medium (345M)	20.3	41.9	38.2	46.5 (+26.2)
GPT2-xl (1.5B)	22.8	44.9	46.5	54.4 (+31.6)
MegatronLM (11B)	23.1	OOM*	OOM*	**64.2** (+41.1)

* MegatronLM (11B) is too large for effective fine-tuning.

Table 2. Knowledge probing Precision@1 on LAMA-34k (left) and LAMA-29k (right). P-tuning outperforms all the discrete prompt searching baselines. And interestingly, despite fixed pre-trained model parameters, P-tuning overwhelms the fine-tuning GPTs in LAMA-29k. (MP: Manual prompt; FT: Fine-tuning; MP+FT: Manual prompt augmented fine-tuning; PT: P-tuning).

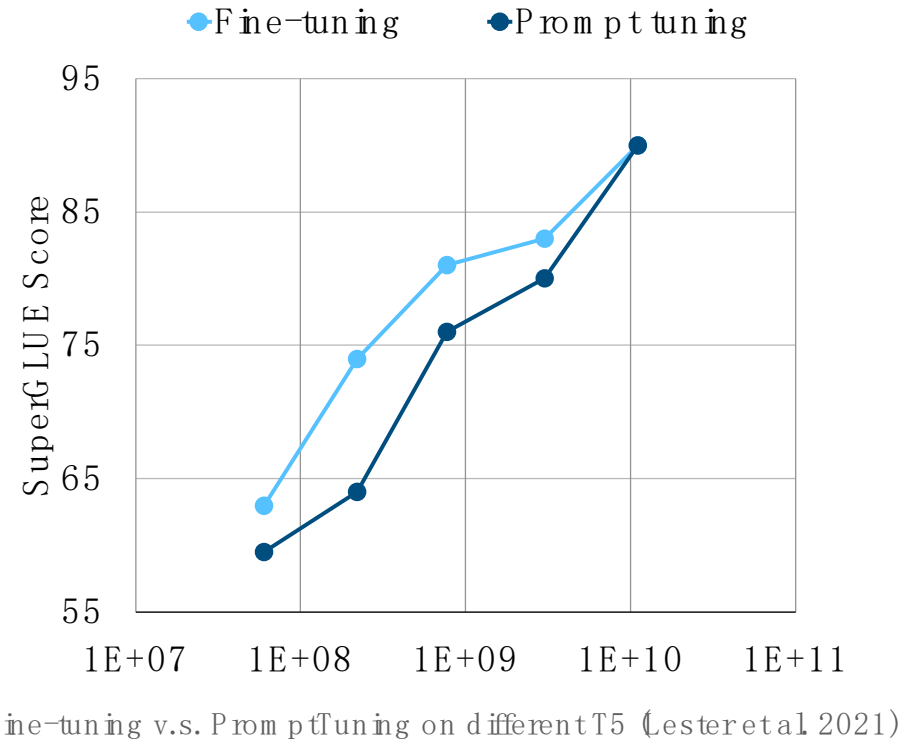
Prompt Tuning

- The Power of Scale for Parameter-Efficient Prompt Tuning (Lester, 2021)
  - Prompt tuning: only tuning prompts as P-tuning does for LAMA
  - As the model grows up to 11B, prompt tuning can be comparable to fine-tuning with only **0.01%** tuned parameters



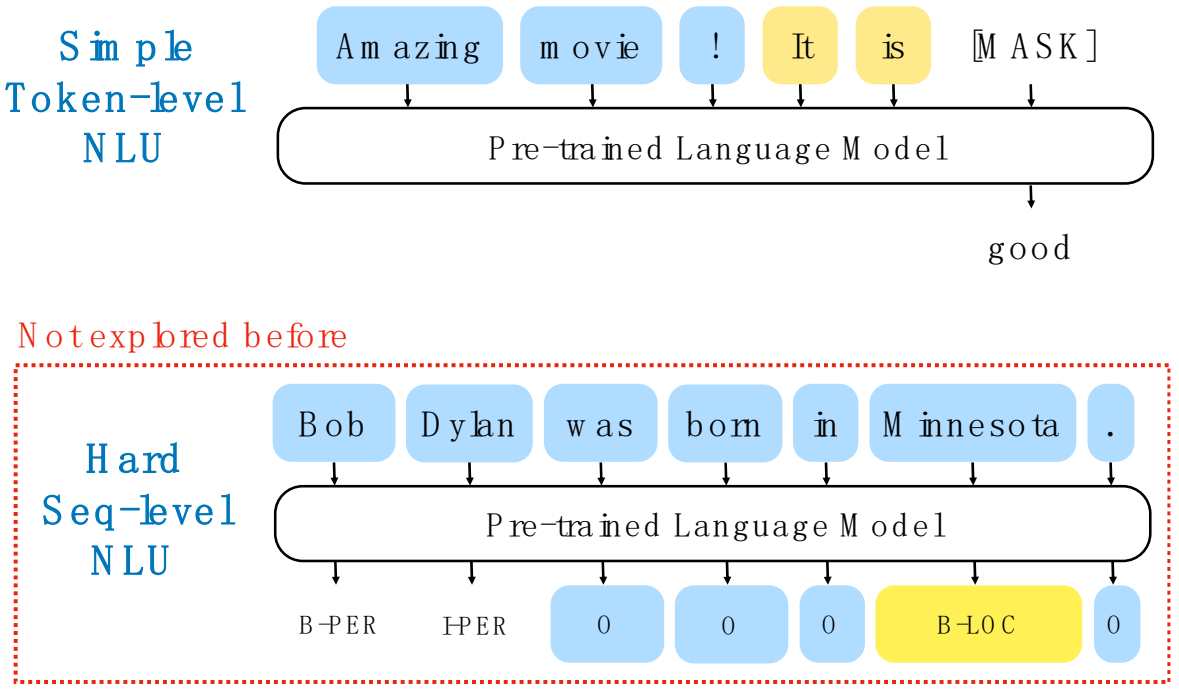
Limitation of Prompt Tuning

- Compared to fine-tuning, prompt tuning still lacks **universality across scales and tasks**



Fine-tuning v.s. Prompt Tuning on different T5 (Lester et al. 2021)

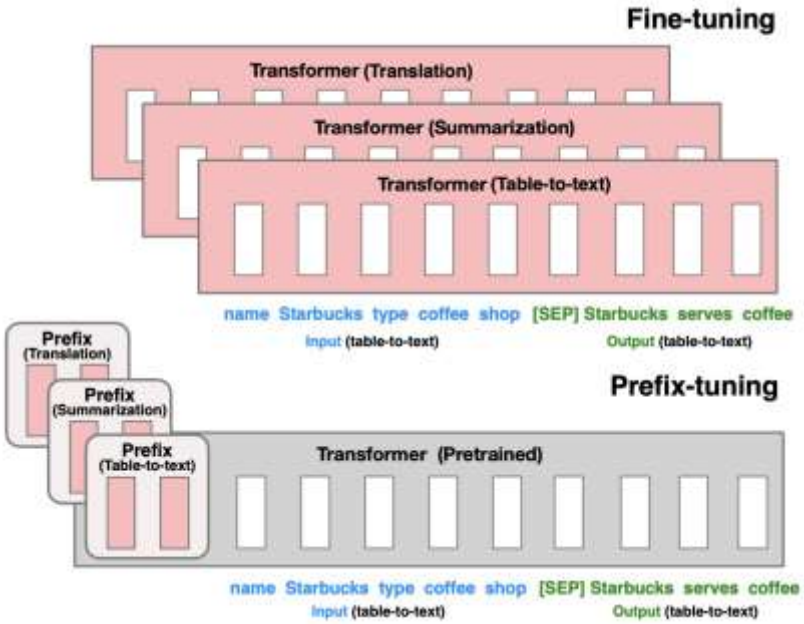
Lack of Universality Across Scales



Lack of Universality Across Tasks

Deep Prompt Tuning

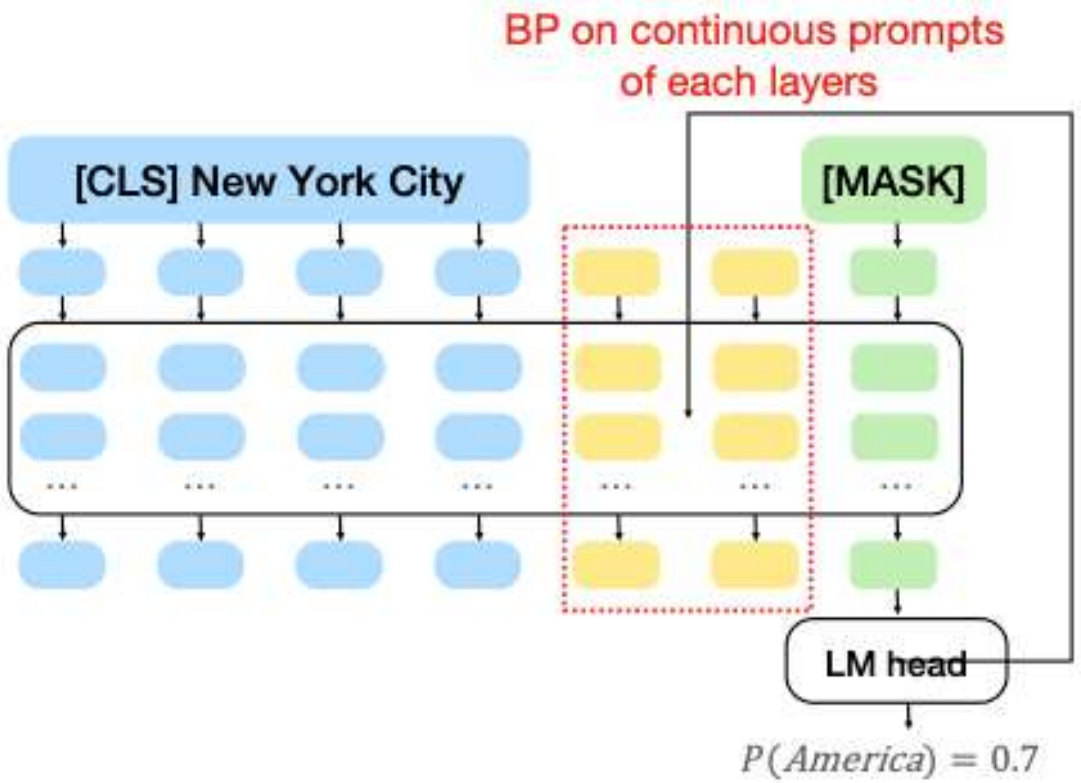
- Prefix-Tuning: Optimizing Continuous Prompts for Generation (Li & Liang, 2021)
  - On NLG, introducing continuous prompts on each transformer layer
  - Can even outperform fine-tuning in NLG tasks on GPTs



	E2E					WebNLG									DART					
	BLEU	NIST	MET	R-L	CIDEr	BLEU			MET			TER ↓			BLEU	MET	TER ↓	Mover	BERT	BLEURT
						S	U	A	S	U	A	S	U	A						
GPT-2$_{\text{MEDIUM}}$																				
FINE-TUNE	68.2	8.62	**46.2**	71.0	2.47	**64.2**	27.7	46.5	**0.45**	0.30	0.38	**0.33**	0.76	0.53	46.2	**0.39**	**0.46**	**0.50**	**0.94**	**0.39**
FT-TOP2	68.1	8.59	46.0	70.8	2.41	53.6	18.9	36.0	0.38	0.23	0.31	0.49	0.99	0.72	41.0	0.34	0.56	0.43	0.93	0.21
ADAPTER(3%)	68.9	8.71	46.1	71.3	2.47	60.4	**48.3**	54.9	0.43	**0.38**	**0.41**	0.35	**0.45**	**0.39**	45.2	0.38	**0.46**	**0.50**	**0.94**	**0.39**
ADAPTER(0.1%)	66.3	8.41	45.0	69.8	2.40	54.5	45.1	50.2	0.39	0.36	0.38	0.40	0.46	0.43	42.4	0.36	0.48	0.47	**0.94**	0.33
PREFIX(0.1%)	**69.7**	**8.81**	46.1	**71.4**	**2.49**	62.9	45.6	**55.1**	0.44	**0.38**	**0.41**	0.35	0.49	0.41	**46.4**	0.38	**0.46**	**0.50**	**0.94**	**0.39**
GPT-2$_{\text{LARGE}}$																				
FINE-TUNE	68.5	8.78	46.0	69.9	2.45	**65.3**	43.1	55.5	**0.46**	0.38	**0.42**	**0.33**	0.53	0.42	**47.0**	**0.39**	0.46	**0.51**	**0.94**	**0.40**
Prefix	**70.3**	**8.85**	**46.2**	**71.7**	**2.47**	63.4	**47.7**	**56.3**	0.45	**0.39**	**0.42**	0.34	**0.48**	**0.40**	46.7	**0.39**	**0.45**	**0.51**	**0.94**	**0.40**
SOTA	68.6	8.70	45.3	70.8	2.37	63.9	52.8	57.1	0.46	0.41	0.44	-	-	-	-	-	-	-	-	-

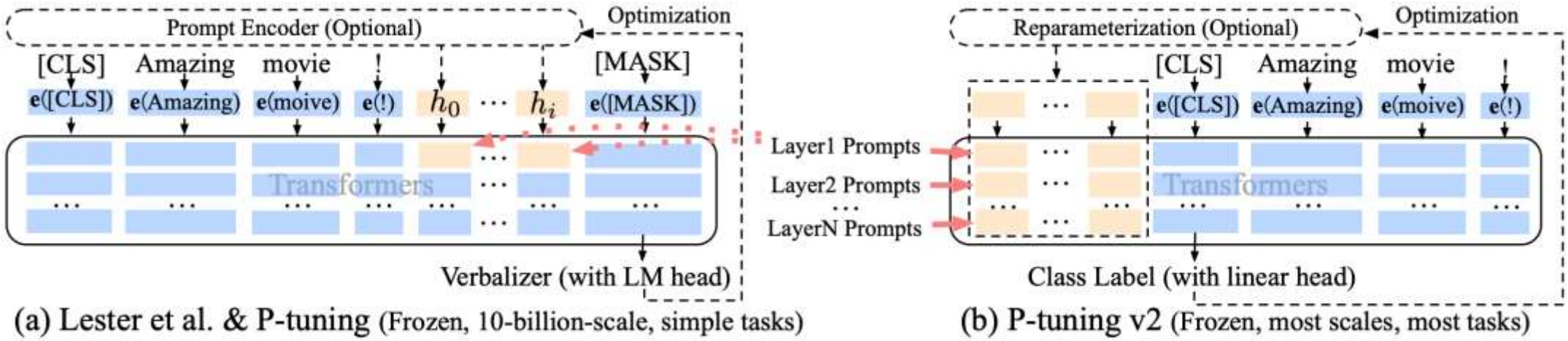
Deep Prompt Tuning

- Soft Prompts (Qin & Eisner, 2021)
  - Backpropagation on continuous prompts of each layer



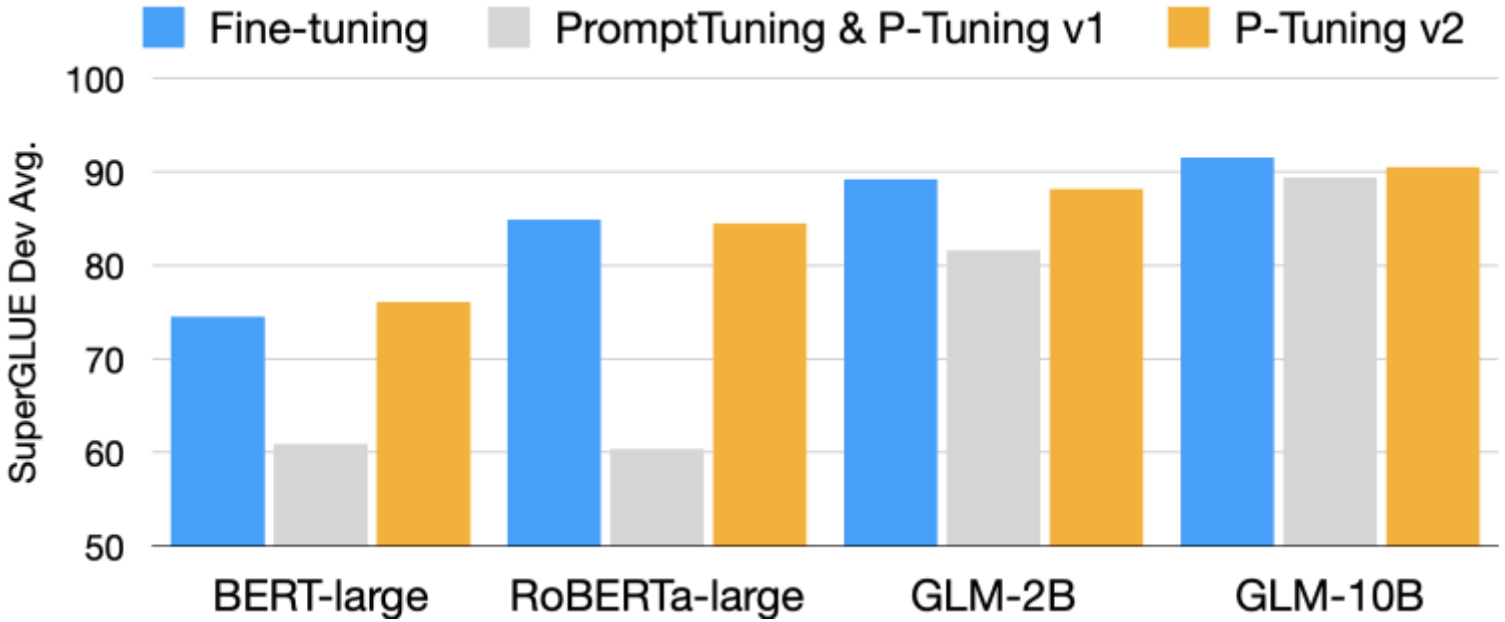
P-tuning v2：Deep Prompt Tuning

- P-tuning v2：Deep Prompt Tuning on NLU tasks
  - Removes reparameterization component
  - Give up verbalizer labels, and return to [CLS] (embedding-based)
  - Multi-task learning with P-tuning v2



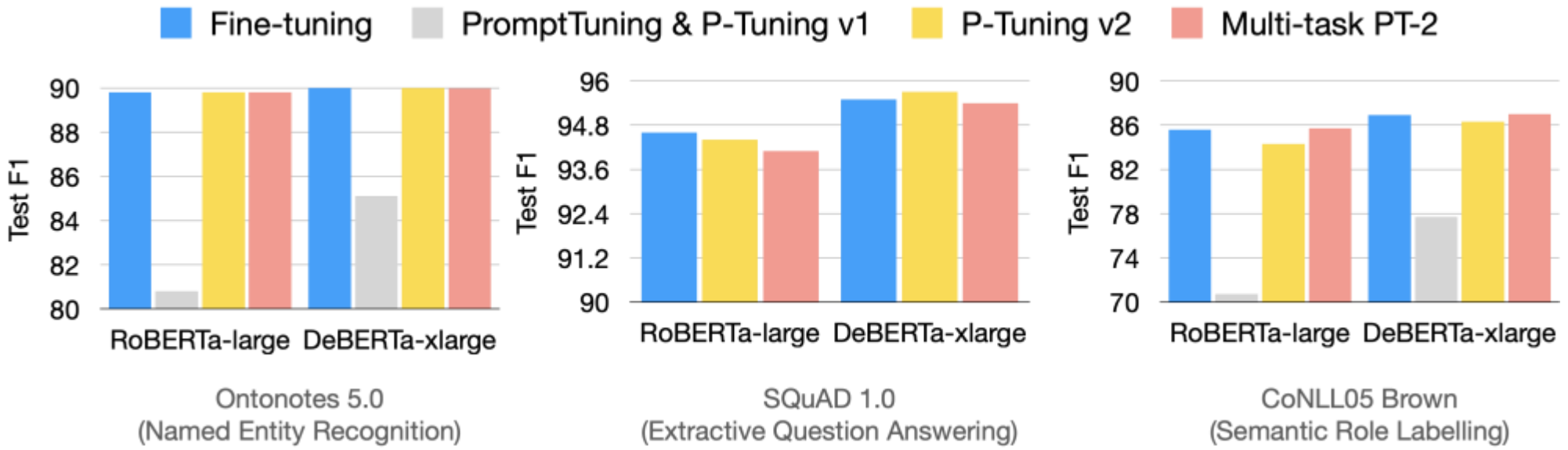
P-tuning v2: Across Scales

- Keep LM's parameter frozen
  - With only **0.1% tuneable parameters**, P-Tuning v2 can be comparable to 100% fine-tuning
  - P-Tuning v2 can be comparable to fine-tuning universally across model scales



P-tuning v2: Across Tasks

- Keep LM's parameter frozen
  - With only 3% tuneable parameters, P-Tuning v2 can be comparable to 100% fine-tuning
  - P-Tuning v2 can be also comparable to fine-tuning on hard semantic-level NLU tasks



Beyond Efficient Tuning & In-Context Learning

- Efficient tuning and In-context learning each have their own advantages and disadvantages
- Problem
  - **Can we further combine their strengths, enhancing the general ability of the pretrained model and making it more friendly to use?**

- Scaling up finetuning

Scaling up Finetuning

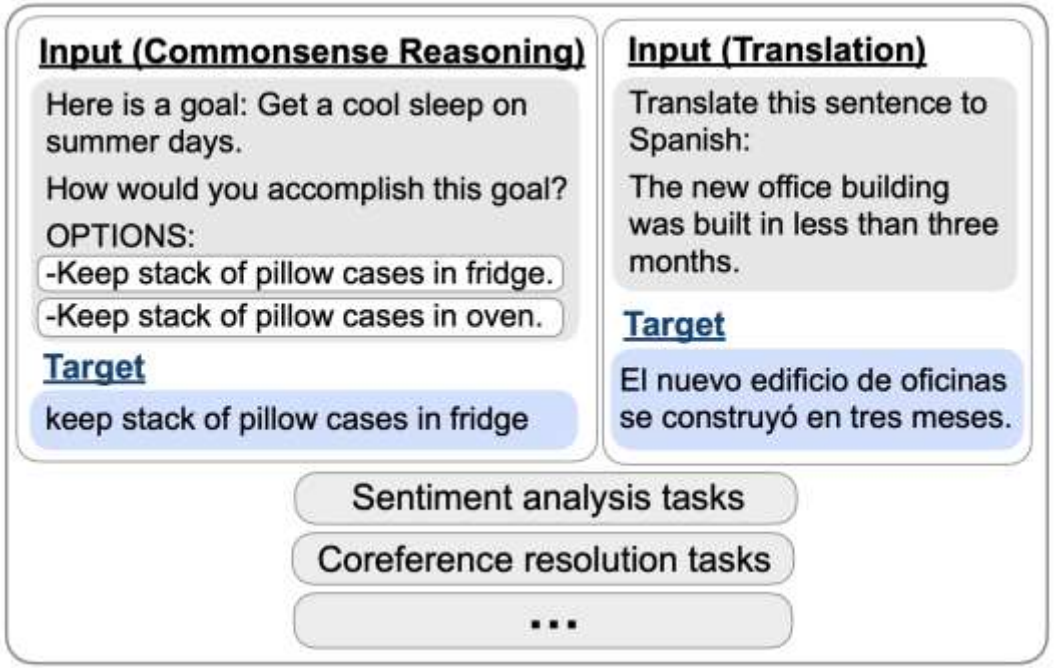
Scaling up Finetuning

- Parameter-efficient tuning
  - Efficiently specialize LLMs to specific tasks

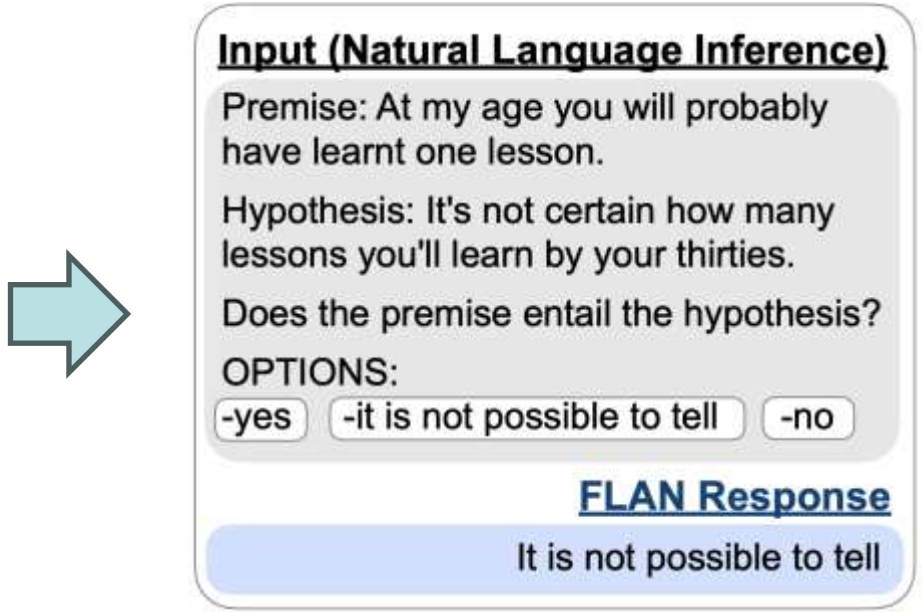
- Problems of traditional finetuning:
  - We have to finetune a specific model for each task. And the finetuned model suffers from poor generalization.
  - Untuned models are better at few-shot learning with proper prompts

Finetuning Towards better Zero-shot Ability

- Generalization to unseen tasks
  - In zero-shot manner



Training data



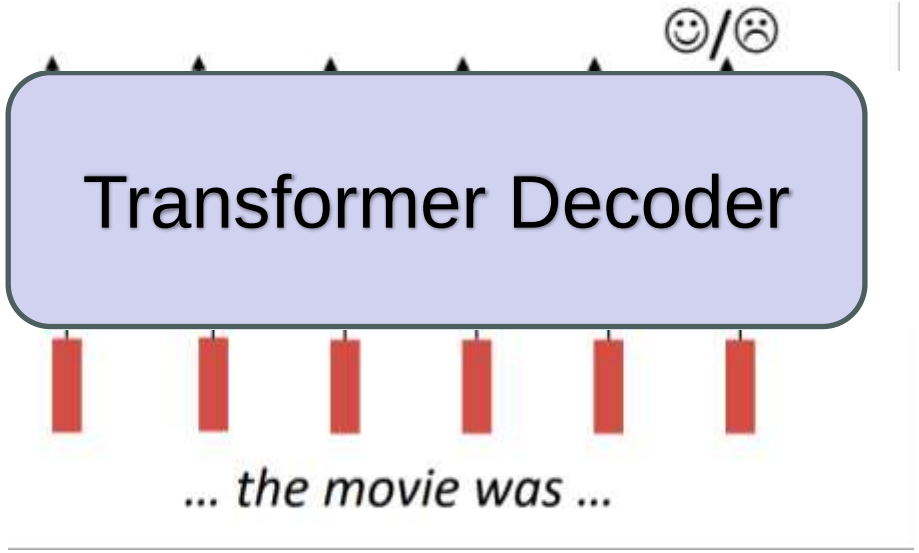
Unseen tasks

Instruction Finetuning

- Finetuning language models on a collection of datasets described via ***instructions***

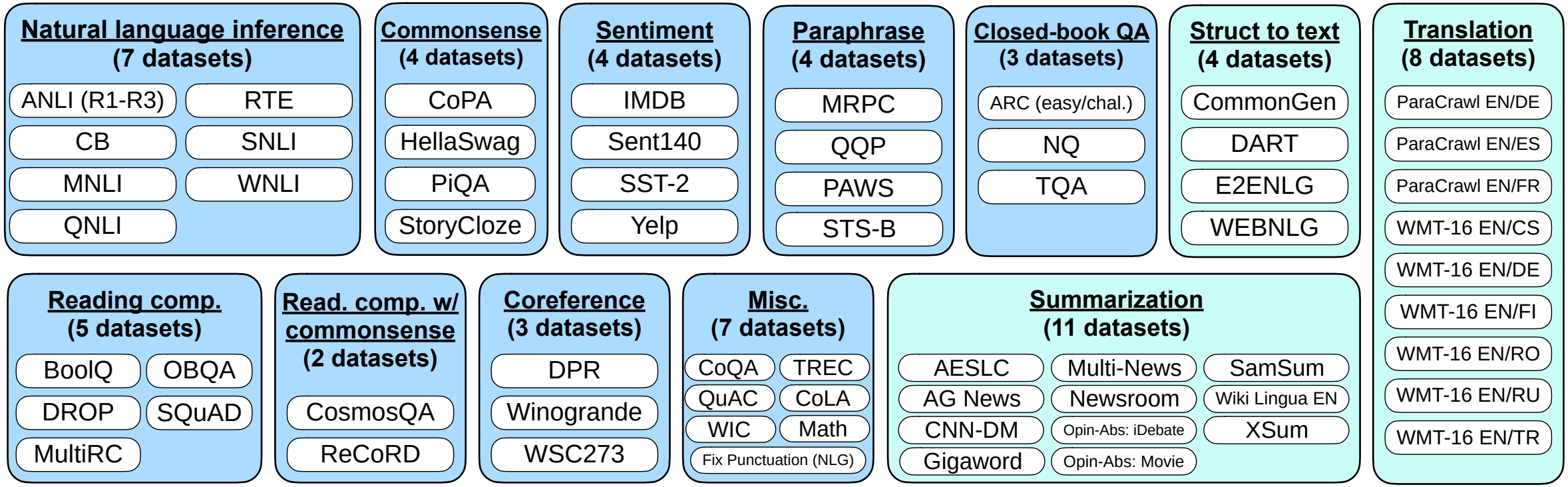
Finetune (on many instruction tasks)

Adapt to unseen various tasks



Finetune on Instruction Datasets

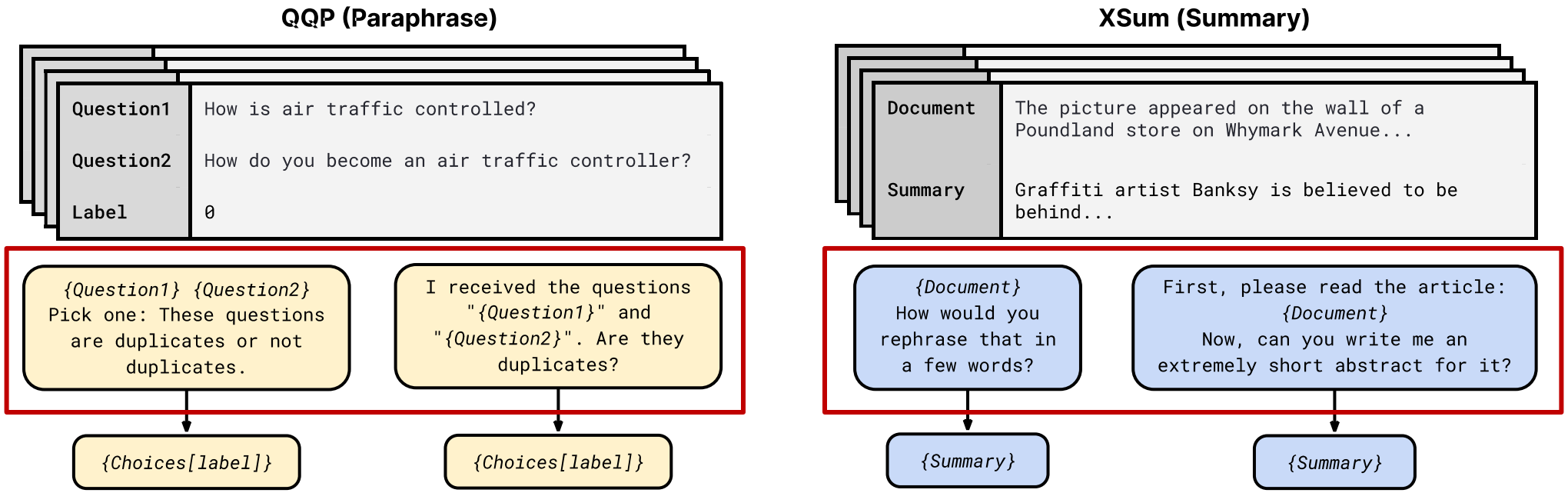
- Finetuning on instructions datasets
  - The fine-tuned model can better respond to instructions and perform on ***unseen test tasks***



Wei et al., Fine-tuned Language Models are Zero-shot Learner

Finetune on Multitask Prompted Datasets

- Generalize zero-shot ability via explicit multitask learning
  - Construct multiple prompts / instructions
  - Crowdsourcing 1939 different prompt templates for 171 datasets



Sanh et al., Multitask Prompted Training Enables Zero-Shot Task Generalization (2021)

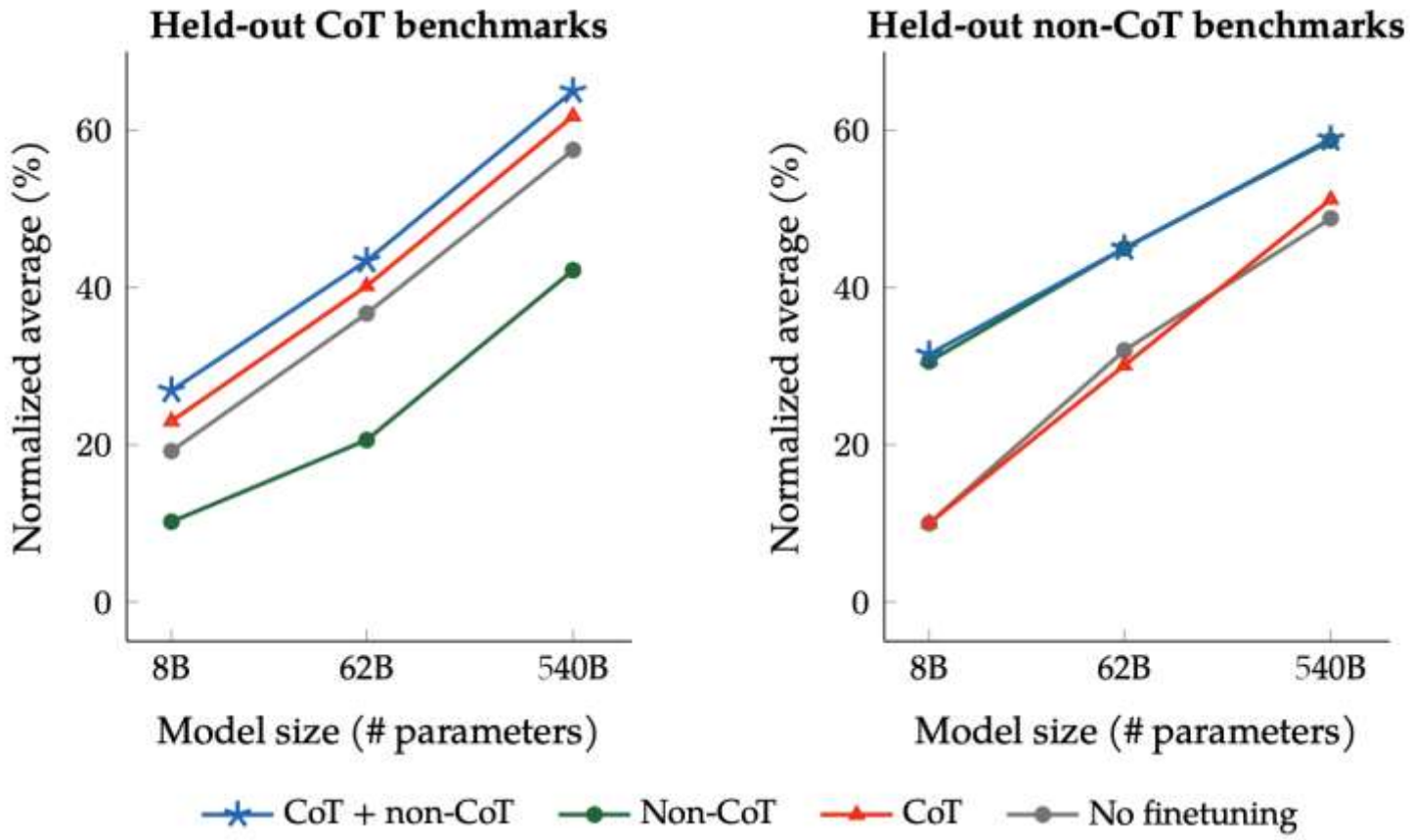
Finetune on Multitask Prompted Datasets

- Multitask Prompted Training Enables Zero-Shot Task Generalization
  - Examples on QQP dataset (identify same-meaning questions in Quora)

1. Can an answer to "**Why are African-Americans so beautiful?**" also be used to answer "**Why are hispanics so beautiful?**"?
2. I received the questions "**Why are African-Americans so beautiful?**" and "**Why are hispanics so beautiful?**". Are they duplicates?
3. **Why are African-Americans so beautiful? Why are hispanics so beautiful?** Pick one: These questions are "duplicates" or "not duplicates".
4. I'm an administrator on the website Quora. There are two posts, one that asks "**Why are African-Americans so beautiful?**" and another that asks "**Why are hispanics so beautiful?**". I can merge questions if they are asking the same thing. Can I merge these two questions?
5. Are the questions "**Why are African-Americans so beautiful?**" and "**Why are hispanics so beautiful?**" asking the same thing?

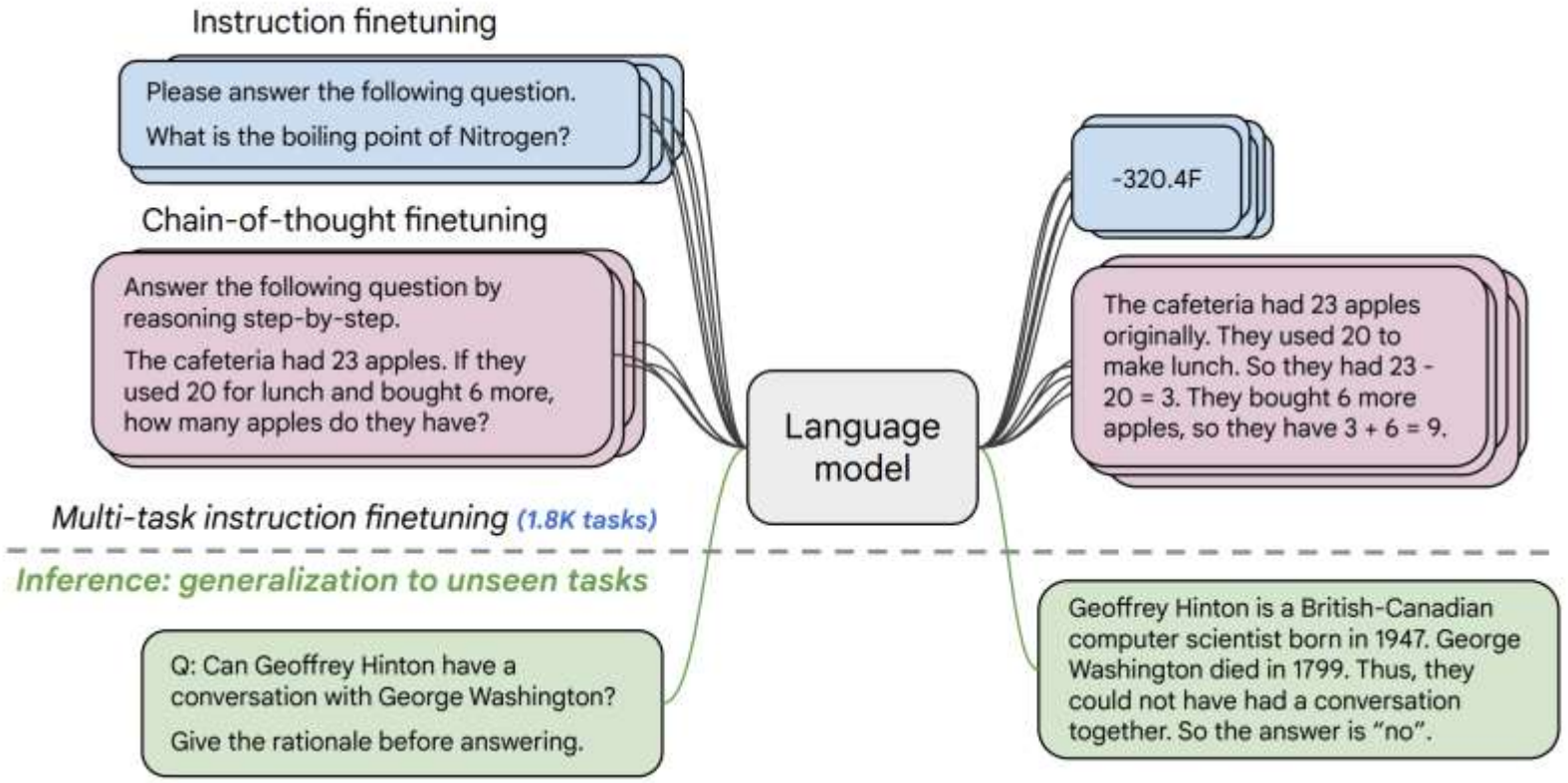
Fine-tune on Multitask Prompted Datasets

- Finetuning with chain-of-thought data is needed to maintain and improve reasonability.



Finetune on More Datasets

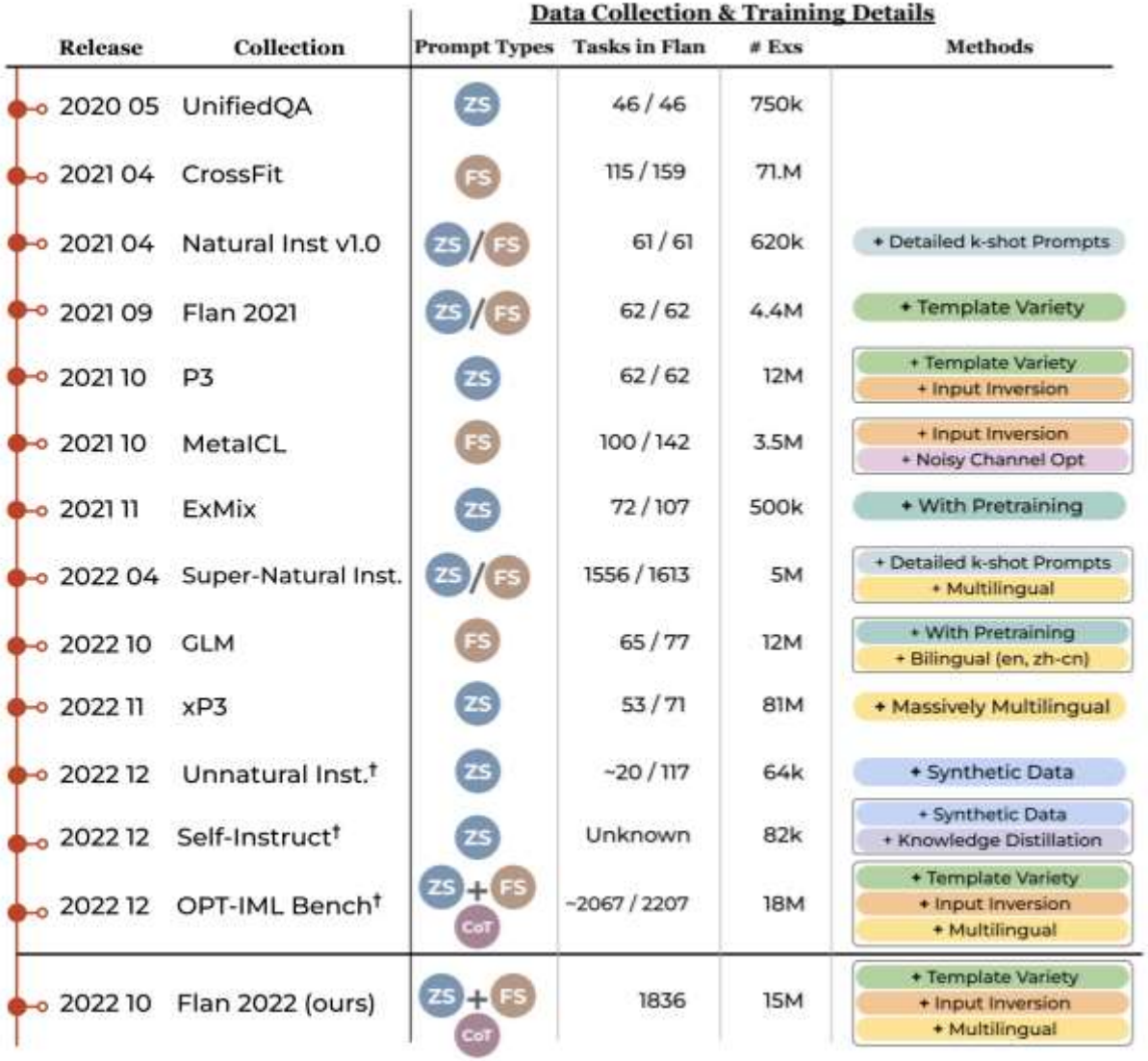
- Scaling Instruction-Finetuned Language
- Collecting (instruction, output) pairs across various types of data



Scaling Instruction-Finetuned Language Models (Chung et al., 2022)

Scaling up Finetuning

- Scaling up
  - Data types
  - Data size
  - Prompt diversity
  - Tasks
  - Instructions
- More and diver data can enable better finetuning performance



		Data Collection & Training Details			
Release	Collection	Prompt Types	Tasks in Flan	# Exs	Methods
2020 05	UnifiedQA	ZS	46 / 46	750k	
2021 04	CrossFit	FS	115 / 159	71.M	
2021 04	Natural Inst v1.0	ZS / FS	61 / 61	620k	+ Detailed k-shot Prompts
2021 09	Flan 2021	ZS / FS	62 / 62	4.4M	+ Template Variety
2021 10	P3	ZS	62 / 62	12M	+ Template Variety + Input Inversion
2021 10	MetaICL	FS	100 / 142	3.5M	+ Input Inversion + Noisy Channel Opt
2021 11	ExMix	ZS	72 / 107	500k	+ With Pretraining
2022 04	Super-Natural Inst.	ZS / FS	1556 / 1613	5M	+ Detailed k-shot Prompts + Multilingual
2022 10	GLM	FS	65 / 77	12M	+ With Pretraining + Bilingual (en, zh-cn)
2022 11	xP3	ZS	53 / 71	81M	+ Massively Multilingual
2022 12	Unnatural Inst.†	ZS	~20 / 117	64k	+ Synthetic Data
2022 12	Self-Instruct†	ZS	Unknown	82k	+ Synthetic Data + Knowledge Distillation
2022 12	OPT-IML Bench†	ZS + FS, CoT	~2067 / 2207	18M	+ Template Variety + Input Inversion + Multilingual
2022 10	Flan 2022 (ours)	ZS + FS, CoT	1836	15M	+ Template Variety + Input Inversion + Multilingual

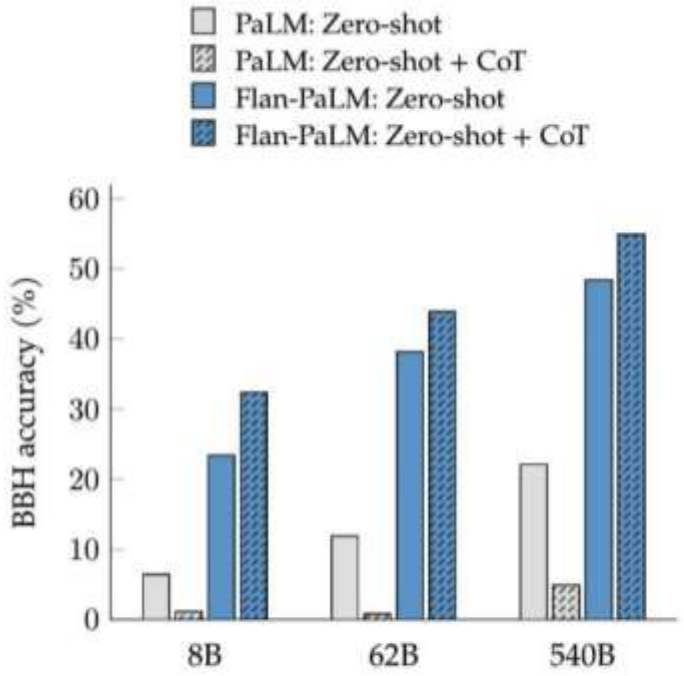
Instruction finetuning unlocks zero-shot reasoning

- Improve performance and generalization
  - Flan-xxx is the model after instruction tuning

	MMLU	BBH-nlp	BBH-alg	TyDiQA	MGSM
Prior best	69.3[a]	73.5[b]	**73.9**[b]	**81.9**[c]	55.0[d]
PaLM 540B					
- direct prompting	69.3	62.7	38.3	52.9	18.3
- CoT prompting	64.5	71.2	57.6	-	45.9
- CoT + self-consistency	69.5	78.2	62.2	-	57.9
Flan-PaLM 540B					
- direct prompting	72.2	70.0	48.2	67.8	21.2
- CoT prompting	70.2	72.4	61.3	-	57.0
- CoT + self-consistency	**75.2**	**78.4**	66.5	-	**72.0**

Instruction finetuning unlocks zero-shot reasoning

- Instruction finetuning unlocks zero-shot reasoning



Instruction-finetuning with CoT data unlocks zero-shot reasoning via "let's think step-by-step".

Model input (Snarks)

Q: Which statement is sarcastic?
Options:
(A) Wow thats crazy, I've never had a takeout meal that used hardwood containers before!
(B) Wow thats crazy, I've never had a takeout meal that used plastic containers before!
A: Let's think step by step.

PaLM 540B output

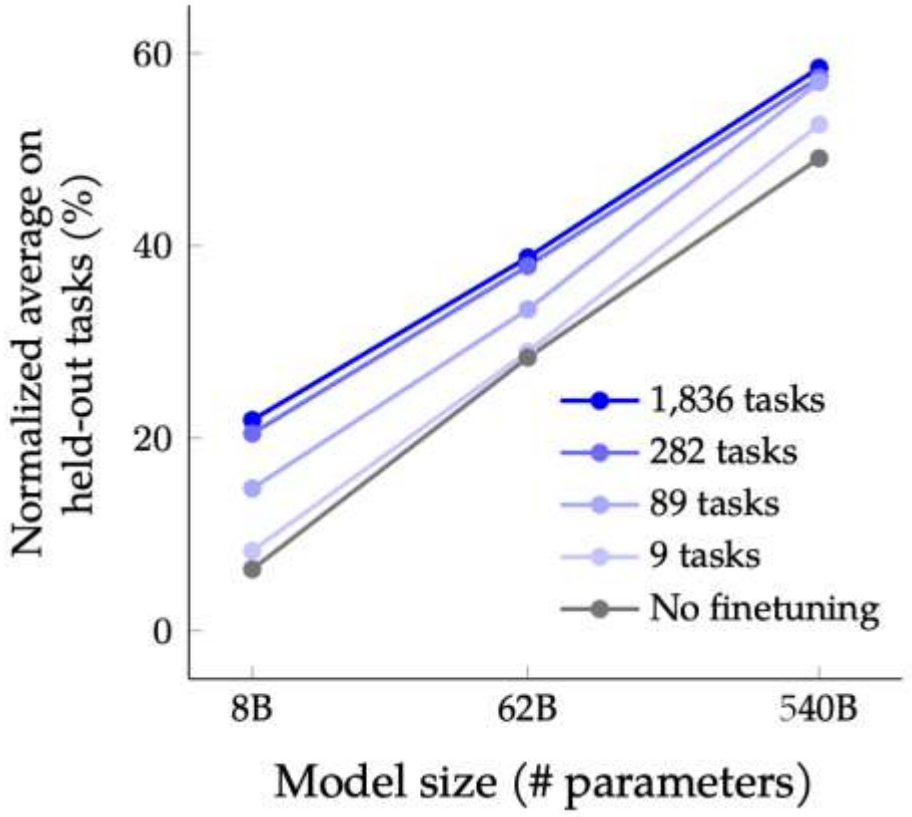
What is the speaker trying to say?
The speaker is trying to say that he has never had a takeout meal that used plastic containers before.
What is the definition of sarcasm?
the use of irony to mock or convey contempt.
What is the definition of irony?
the expression of one's meaning by using language that normally signifies the opposite [...]
✖ (never stops generating)

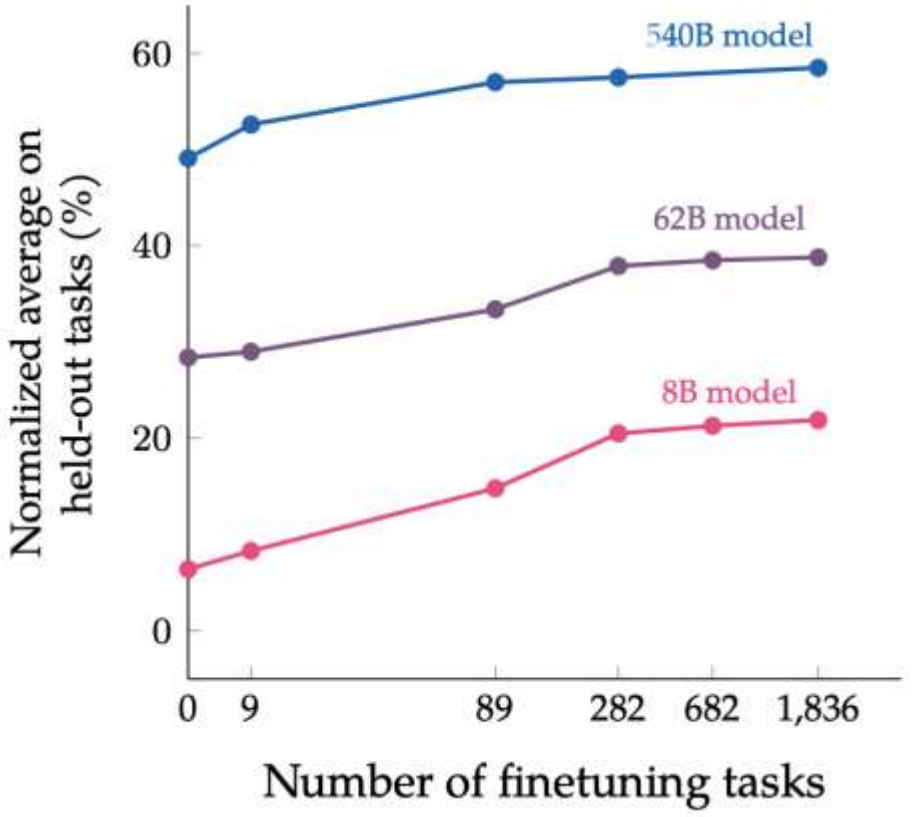
Flan-PaLM 540B output

Plastic containers are very common for takeout meals. So, the answer is (B). ✔

Scaling Property in Instruction Finetuning

- Scaling behavior of multi-task instruction finetuning with respect to model size and number of finetuning tasks



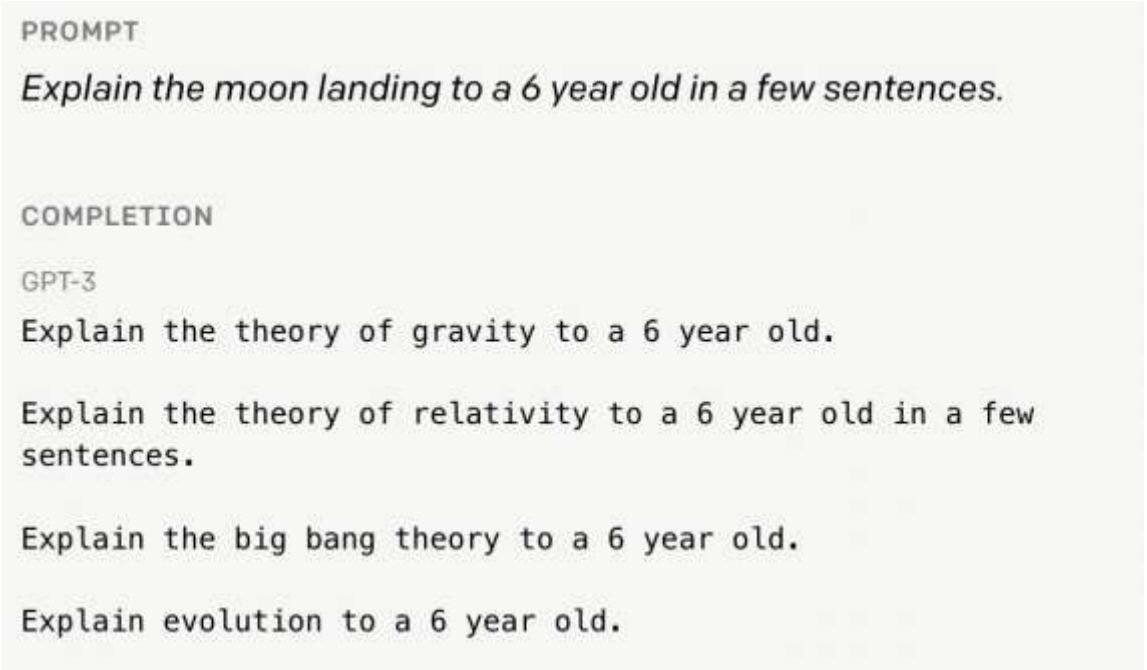


One Step Further

- Instruction finetuning unlocks zero-shot reasoning
- Also elicits a more user-friendly model

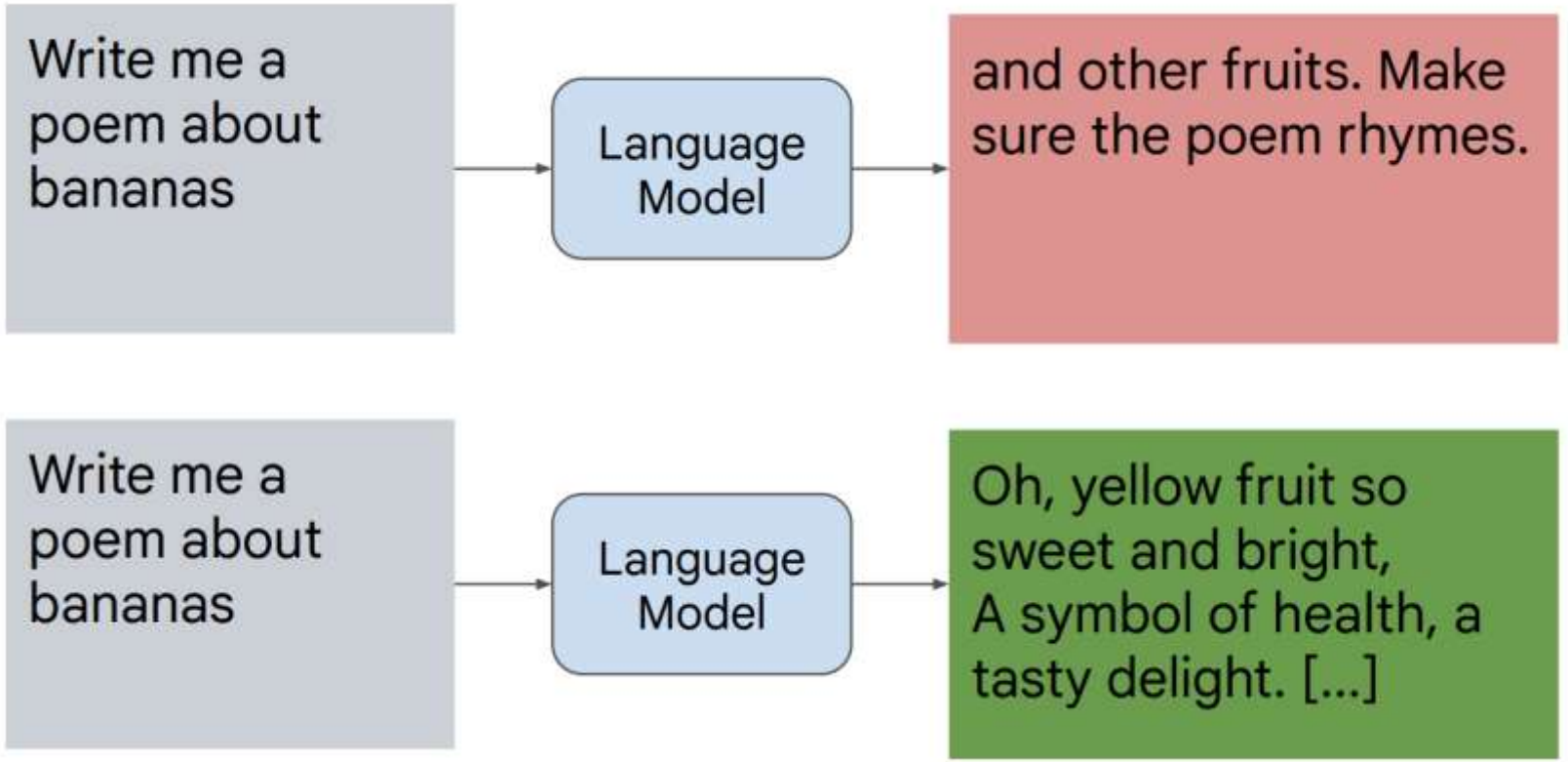
Recall: what does pretraining learn

- Explain the moon landing to a 6 year old in a few sentences.
  - Without proper ``prompt engineering'', GPT-3 won't return satisfying answers yet output "similar" sentences.



Towards User-friendly Models

- Follow human instructions and respond to prompts



https://ai4comm.media.mit.edu/slides/emergence.pdf

Inspirations From Instruction Finetuning

- Instruction enables language models to respond to / follow human instructions

- Can we construct more complex instruction data and responses so that language model can better align with human values
  - From GPT to ChatGPT

Summary

- Multitask instruction tuning can significantly improve zero-shot generalization across many tasks

- And enables language models to respond to human instructions

Finetuning Language Models

- Efficienct tuning
  - Parameter efficienct tuning
  - → specific model

- Strong zero-shot generalization and instruction-following
  - Instruction tuning
  - → powerful & general model

Finetuning Language Models

Jie Tang, KEG, Tsinghua University

http://keg.cs.tsinghua.edu.cn/jietang
https://github.com/THUDM