ICLR 2024 Poster

When Do Prompting and Prefix-Tuning Work?
A Theory of Capabilities and Limitations

Aleksandar Petrov, Philip Torr, Adel Bibi

Department of Engineering Science, University of Oxford, United Kingdom

24.2.29
Minkyu Kim
EffL@POSTECH

Index

- **Introduction**
- **Background**
- **Analyze & Delineate**
  - Soft prompting has more capacity than prompting
  - Prefix-tuning can only bias the output of an attention head
  - The bias can elicit skills from the pretrained model
  - Effects of prefix-tuning beyond the single attention layer
- **Limitation**
- **Conclusion**

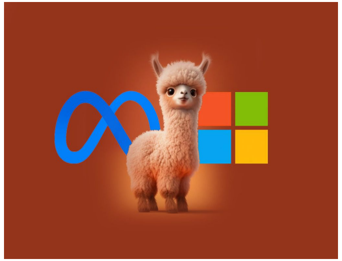
Introduction

Training the cutting-edge language model. needs huge cost

- GPT4. more than $100M
- LLaMA-2-7b. $85K

Note. Without cost to buy hardware (~ $30M)

👉 Out of reach for most academic researchers (even fine-tuning)

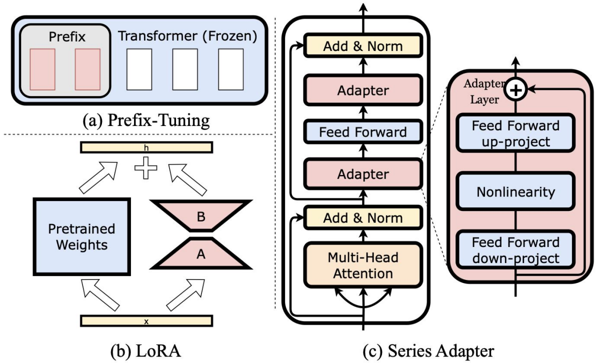


https://web.archive.org/web/20230418190335/https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/
https://news.ycombinator.com/item?id=35391469

Introduction

Alternative. ‘Efficient’ fine-tuning

- Sparsely modifying the parameters of the model
  - Adapter modules, Low-rank updates
- **Modifying its input context** (Context-based fine-tuning)
  - In-context learning (GPT3)
  - Prompt tuning(Hard prompt tuning, Soft prompt tuning, Prefix-tuning)



Hu et al., “LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models”,EMNLP, 2023.
Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models”, ICLR, 2022.

Introduction

Modifying its input context (Context-based fine-tuning).

- The most popular approach: prompting
  - Generation is conditioned on either human-crafted or automatically optimized tokens
  - Have witnessed impressive empirical successes and widespread adoption
- Type
  - Tokens(word sequence): Hard Prompt 👉 discrete information
  - Vectors(parameters): Soft Prompt 👉 continuous information
    - greater expressiveness due to the expansive nature of continuous space

Introduction

However. we have little theoretical understanding of how they work

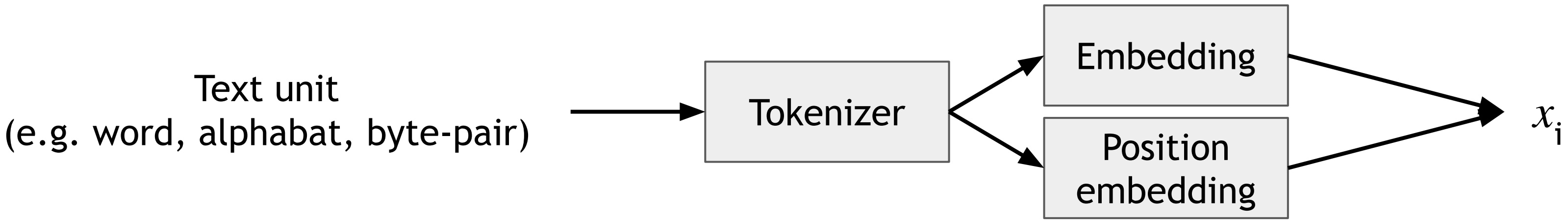
- In this work, we…
  - Analyse the influence of prompts and prefixes on the computations of a pretrained model
  - Delineate their limitations
- Specifically, we address the following questions:
  - Can a transformer utilize the soft prompt or prefix vector?
  - Since prefix-tuning is more expressive than prompting, is it as expressive as full fine-tuning?
  - If context-based fine-tuning methods suffer from such structural limitations, how come they have high empirical performance?

(Authors answered these in next sections)

Background

Input data.

- **Input sequences.** (x_1,...,x_p), x_i ∈ {1,...,V}
  - V: size of vocabulary
- **Embedding matrix (token → vector).** $E \in R^{d\_e*v}$
  - d_e: dimension of embedding
- **One-hot i-th position encoding.** $e_N(i)$
- **Embedding for i-th position.** $x_i = [E^T_{:,x\_i}, e_N^T(i)]^T$



Background

Transformer architecture. Consists of alternating attention blocks

- **Attention block.** consists of H heads
  - **Attention matrix** for head 'h': $\mathbf{A}^h \in R^{p \times p}$
- **Linear.** applied to all positions of the sequence
  - followed by ReLU activation layer

$$\boldsymbol{A}^h_{ij} = \frac{\exp\left(T/\sqrt{k}(\boldsymbol{W}^h_Q \boldsymbol{x}_i)^\top(\boldsymbol{W}^h_K \boldsymbol{x}_j)\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}(\boldsymbol{W}^h_Q \boldsymbol{x}_i)^\top(\boldsymbol{W}^h_K \boldsymbol{x}_r)\right)}$$

[Attention matrix]

$$\mathcal{A}[(\boldsymbol{W}^1_Q, \ldots, \boldsymbol{W}^H_Q), (\boldsymbol{W}^1_K, \ldots, \boldsymbol{W}^H_K), (\boldsymbol{W}^1_V, \ldots, \boldsymbol{W}^H_V)](\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p) = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_p)$$
$$\boldsymbol{t}_i = \sum_{h=1}^{H} \sum_{j=1}^{p} \boldsymbol{A}^h_{ij} \boldsymbol{W}^h_V \boldsymbol{x}_j.$$

[Attention block]

$$\mathcal{L}[\boldsymbol{M}, \boldsymbol{b}](\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x} + \boldsymbol{b}$$
$$\hat{\mathcal{L}}[\boldsymbol{M}, \boldsymbol{b}](\boldsymbol{x}) = \mathrm{ReLU}(\boldsymbol{M}\boldsymbol{x} + \boldsymbol{b})$$

[Linear layer]

$$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p) = \left(\mathcal{A}_1 \fatsemi \hat{\mathcal{L}}_{1,1} \fatsemi \mathcal{L}_{1,2} \fatsemi \mathcal{A}_2 \fatsemi \hat{\mathcal{L}}_{2,1} \fatsemi \mathcal{L}_{2,2} \fatsemi \mathrm{softmax}\right)\left(\begin{bmatrix}\boldsymbol{E}_{:,\mathrm{x}_1} \\ \boldsymbol{e}_N(1)\end{bmatrix}, \ldots, \begin{bmatrix}\boldsymbol{E}_{:,\mathrm{x}_p} \\ \boldsymbol{e}_N(p)\end{bmatrix}\right)$$

[Output of the transformer]

Background

Context-based fine-tuning of a pretrained model.

- Hard **Prompting.** prefixing the input with a token sequence
  - Guide the model response
- Soft **prompting.** learnable input embeddings

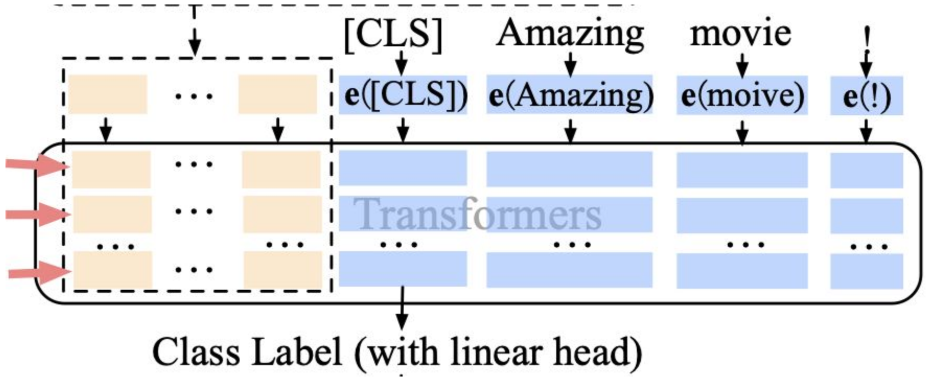
Constant: hard prompting
Learnable: soft prompting

$$\left(\boxed{\begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{e}_N(1)\end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{s}_{n_S} \\ \boldsymbol{e}_N(n_S)\end{bmatrix}}, \begin{bmatrix} \boldsymbol{E}_{:,\mathrm{x}_1} \\ \boldsymbol{e}_N(n_S+1)\end{bmatrix}, \ldots, \begin{bmatrix} \boldsymbol{E}_{:,\mathrm{x}_{n_X}} \\ \boldsymbol{e}_N(n_S+n_X)\end{bmatrix}\right)$$

Background

Context-based fine-tuning of a pretrained model.

- **Prefix tuning.** applied soft prompt across the depth of the model
  - The first n_s positions for all attention blocks are learnable parameters
    - Optimize the inputs of every attention layer
  - Successful at fine-tuning models

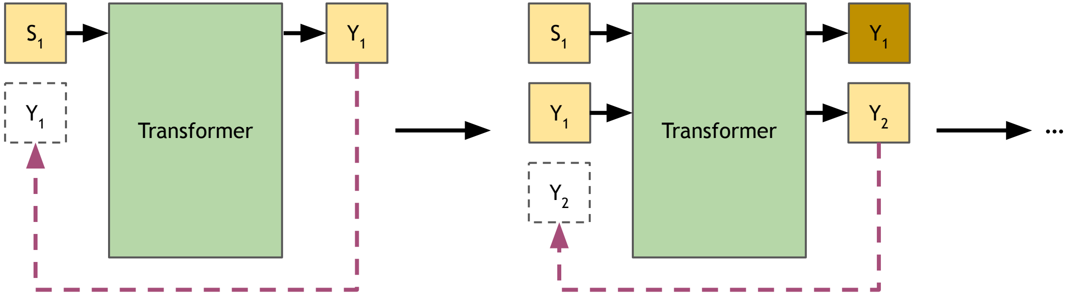


Liu et al., “P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks”, ACL, 2022.

Soft prompting has more capacity than prompting

Case: Unconditional generation with a single system token

- Vocabulary size: V
- Deterministic autoregressive function
- **If we use hard prompt.** Enable to get number of V outputs
    - [f(s_1), f(s_2),..., f(s_V)] (Finite number('V') of case)
- **If we use soft prompt.**
    - Real-value → Infinite number of cases



$$(\mathrm{Y}_1, \ldots, \mathrm{Y}_N) = f(\mathrm{S}_1) = f_{\mathrm{S}_1}$$

[using hard prompt]

$$(\mathrm{Y}_1, \ldots, \mathrm{Y}_N) = f(\boldsymbol{s}_1) = f_{\boldsymbol{s}_1}$$

[using soft prompt]

Soft prompting has more capacity than prompting

Guessing the successful result of soft prompt (comparing with hard prompt)

- Unconditional generation with a single system token
  - If we use soft prompt.
    - Note: it does not mean that the transformer architecture can represent a function that achieves any type of output in practice
      - it is not obvious if there is a surjective map from soft prompt to word in vocab.
  - **Theorem 1.** Condition that enable to create 'soft prompt + transformer'
    , which generates infinite unique outputs

Theorem 1 (Exponential unconditional generation capacity of a single virtual token). *For any $V, N>0$, there exists a transformer with vocabulary size V, context size N, embedding size $d_e=N$, one attention layer with two heads and a three-layer MLP such that it generates any token sequence $(Y_1, \dots, Y_N) \in \{1, \dots, V\}^N$ when conditioned on the single virtual token* $\boxed{\boldsymbol{s}_1 = ((Y_1-1)/V, \dots, (Y_N-1)/V).}$

Soft prompting has more capacity than prompting

Guessing the successful result of soft prompt (comparing with hard prompt)

- Conditional generation with a single system token
  - Prompt + User input (X_1, …, X_{n_x}), Output (Y_1, …, Y_{n_y})
  - **If we use hard prompt.**
    - $X_1 \rightarrow Y_1$: V^V cases ($1 \rightarrow [1,\ldots,V]$, $2 \rightarrow [1,\ldots,V]$,…,$V \rightarrow [1,\ldots,V]$ 👉 $[1,\ldots,V] \rightarrow [1,\ldots,V]$)
      - $S_1 \in \{1, \ldots, V\}$: V cases = less than V^V
        - → Can't control the output by modifying system token
          (S_1 cannot be used to specify an arbitrary map from input to output)
  - **If we use soft prompt.**
    - S_1 is 'real value' = infinite cases
      - Can control the all cases of output(V^V)

$$(Y_1 = f(\boxed{S_1, X_1}) = f_{S_1}(X_1))$$

Soft prompting has more capacity than prompting

Guessing the successful result of soft prompt (comparing with hard prompt)

- Conditional generation with a single system token
  - **Theorem 2.** Soft prompting is also more expressive for the conditional behavior of a transformer model

Theorem 2 (Conditional generation capacity for a single virtual token ($n_X = n_Y = 1$)). *For any $V > 0$, there exists a transformer with vocabulary size V, context size $N = 2$, embedding size $d_e = V$, one attention layer with two heads and a three-layer MLP that reproduces any map $m : [1, ..., V] \rightarrow [1, ..., V]$ from a user input token to a model response token when conditioned on a single virtual token $\boxed{\boldsymbol{s}_1 = \left(\frac{m(1)}{V}, ..., \frac{m(V)}{V}\right)}$. That is, by selecting $\boldsymbol{s}_1$ we control the model response to any user input.*

Soft prompting has more capacity than prompting

Conclusion.

- Soft prompting, Prefix-tuning posses greater expressiveness than prompting
- We can fully determine the map from user input to output using virtual tokens
  - Soft prompting is as powerful as full fine-tuning

Prefix-tuning can only bias the attention head's output

Strength of soft prompting & prefix-tuning.

- Needs specific network's parameters (Theorem 1, 2)

So, we can ask question,

- Given a fixed pre-trained model, Can prefix-tuning be considered equally powerful to full fine-tuning?
  - **Answer.** prefix can't change the relative attention over the content X, Y
    - Can only bias the attention block outputs in a subspace of rank n_s
      👉 Making it strictly less powerful than full fine-tuning

Prefix-tuning can only bias the attention head's output

Prefix-tuning can't alter the attention pattern **of an attention head**

- A_{ij}: Attention that i-th position token gives to j-th position token

$$\boldsymbol{A}_{ij} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{W}_Q^\top \boldsymbol{W}_K \boldsymbol{x}_j\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{W}_Q^\top \boldsymbol{W}_K \boldsymbol{x}_r\right)} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_j\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r\right)}$$

- **Full fine-tuning.** Can enact arbitrary changes to W_Q and W_K
  - Change the attention patterns arbitrarily

$(W_Q)^T W_K$

$$\boldsymbol{A}_{ij}^{\text{ft}} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boxed{\boldsymbol{H}} \boldsymbol{x}_j + T/\sqrt{k}\, \boldsymbol{x}_i^\top \boxed{\Delta \boldsymbol{H}} \boldsymbol{x}_j\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r + T/\sqrt{k}\, \boldsymbol{x}_i^\top \boxed{\Delta \boldsymbol{H}} \boldsymbol{x}_r\right)}$$

Prefix-tuning can only bias the attention head's output

Prefix-tuning can't alter the attention pattern of an attention head

- **Prefix-tuning.** If we have a prefix of length one at position 0,
  - Only increase denominator of attention where to get attention not at 0
    👉 prefix can't change the attention pattern (can't modify **what an attention head attends to**)
  - Re-writing the A_{ij}^{pt} makes this more evident

$$\boldsymbol{A}_{ij}^{\text{pt}} = \boldsymbol{A}_{ij} \sum_{r=1}^{p} \boldsymbol{A}_{ir}^{\text{pt}} = \boldsymbol{A}_{ij} \boxed{(1 - \boldsymbol{A}_{i0}^{\text{pt}})}$$

just scaling

$$\boldsymbol{A}_{i0}^{\text{pt}} = \frac{\exp\left(T/\sqrt{k}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1\right)}{\exp\left(\frac{T}{\sqrt{k}}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1\right) + \sum_{r=1}^{p} \exp\left(\frac{T}{\sqrt{k}}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r\right)}, \quad \boldsymbol{A}_{ij}^{\text{pt}} = \frac{\exp\left(T/\sqrt{k}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_j\right)}{\boxed{\exp\left(\frac{T}{\sqrt{k}}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1\right)} + \sum_{r=1}^{p} \exp\left(\frac{T}{\sqrt{k}}\ \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r\right)} \text{ for } j \geq 1.$$

Attention to prefix

Attention to other tokens

(Added value to denominator)

Prefix-tuning can only bias the attention head's output

Prefix-tuning only adds a bias to the attention block output

- **Term containing prefix.** Independent of the content (x_1 ,..., x_p)
  - Act as bias
- In contrast with full fine-tuning
  - Allow for content-dependent change of attention and value computation.

$$\boldsymbol{t}_i = \sum_{j=1}^{p} \boldsymbol{A}_{ij} \boldsymbol{W}_V \boldsymbol{x}_j, \qquad \boldsymbol{t}_i^{\text{ft}} = \sum_{j=1}^{p} \boldsymbol{A}_{ij}^{\text{ft}} (\boldsymbol{W}_V + \boxed{\Delta \boldsymbol{W}_V}) \boldsymbol{x}_j,$$

$$\boldsymbol{t}_i^{\text{pt}} = \boldsymbol{A}_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1 + \sum_{j=1}^{p} \boldsymbol{A}_{ij}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{x}_j \overset{(6)}{=} \boldsymbol{A}_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1 + \sum_{j=1}^{p} \boldsymbol{A}_{ij} (1 - \boldsymbol{A}_{i0}^{\text{pt}}) \boldsymbol{W}_V \boldsymbol{x}_j = \boxed{\boldsymbol{A}_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1} + (1 - \boldsymbol{A}_{i0}^{\text{pt}}) \boldsymbol{t}_i.$$

Prefix-tuning can only bias the attention head's output

Conclusion,

- In real-world scenario, prefix-tuning has some limitations,
    - It can be competitive with full fine-tuning only in very limited circumstances
    - Transformers do not behave like we want
        - Models are typically trained with token inputs rather than soft prompts

The bias can elicit skills from the pre-trained model

Target. Explain when and why prefix-tuning can work in practice.

- **Hypothesis.** prefix-tuning cannot be used to gain new knowledge but can bring to the surface latent knowledge present in the pretrained model
- **Test.** Constructing small transformers trained on one or few tasks
  - Task: Sort numbers into ascending/descending order
    - Training strategy 1: Pre-training task = Fine-tuning task
    - Training strategy 2: Pre-training task != Fine-tuning task

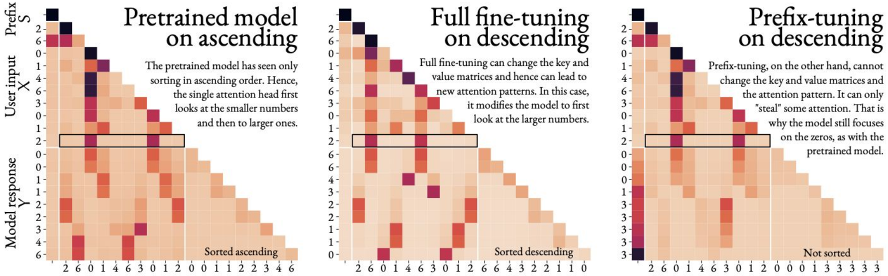
The bias can elicit skills from the pre-trained model

Test results. Prefix-tuning can’t learn a new task requiring a different attention pattern

- 1-layer, 1-head transformer

	Ascending	Descending
Pretrain on asc.	91±5%	0±0%
Full fine-tune on desc.	0±0%	85±5%
Prefix-tune on desc.	0±0%	0±0%

Accuracy (10 random seeds)



: the attention when the first response Y_1 is being generated

The bias can elicit skills from the pre-trained model

Test results. Prefix-tuning can elicit a skill from the pre-trained model

- Pretrain a 1-layer, 4-head model with solutions…
  - Sorted in ascending (↗) order (1st task)
  - Sorted in descending (↘) order (2nd task)
  - Adding one (+1) to each element (3th task)
  - Adding two (+2) to each element (4th task)

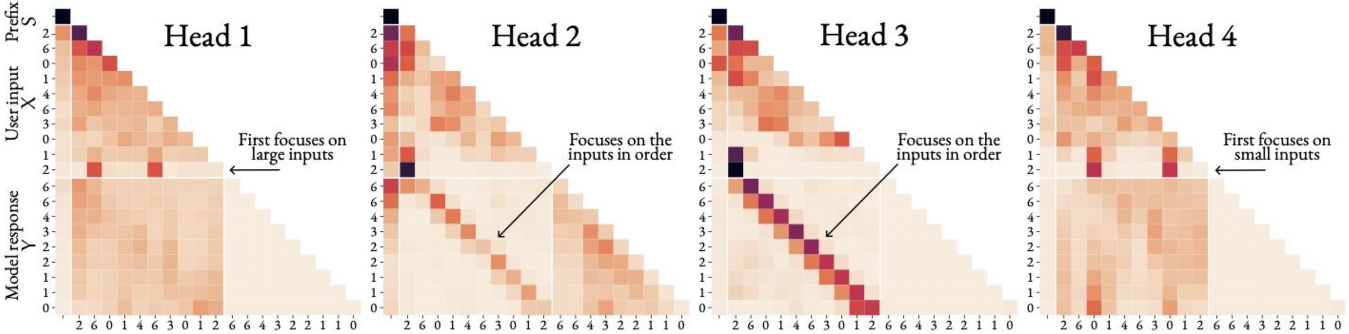
  👉 (Each head learns a different task)

The bias can elicit skills from the pre-trained model

Test results. Prefix-tuning can elicit a skill from the pre-trained model

- Compared to the previous case, prefix-tuning is more successful
  - because the pretrained model contains the attention mechanisms for solving the four tasks

Accuracy on:	↗	↘	+1	+2
Pretrained	25±13%	25±12%	24±11%	22±7%
Prefix-tune on ↗	95± 2%	0± 0%	0± 0%	0±0%
Prefix-tune on ↘	0± 0%	90± 3%	1± 1%	1±1%
Prefix-tune on +1	0± 0%	1± 3%	95± 6%	0±1%
Prefix-tune on +2	0± 0%	0± 0%	1± 2%	98±5%



The bias can elicit skills from the pre-trained model

Conclusion. Prefix-tuning can combine knowledge from pre-training tasks to solve new tasks

- "Skill" required to solve the new task = "Skills" the pre-trained model has seen
- **Test again.** 40-layer 4-head model with the same four tasks
  - length of prefixes: 10 (n_S =10)
  - New task H: mapping each element to the number of elements in the sequence with the same value

Accuracy on:	$\nearrow$	$\searrow$	+1	+2	$\nearrow$+1	$\mathbb{H}$
Pretrained	17%	23%	34%	25%	0%	0%
Prefix-tune on $\nearrow$	100%	0%	0%	0%	0%	0%
Prefix-tune on $\searrow$	0%	100%	0%	0%	0%	0%
Prefix-tune on +1	0%	0%	100%	0%	0%	0%
Prefix-tune on +2	0%	0%	0%	100%	0%	0%
Prefix-tune on $\nearrow$+1	0%	0%	0%	0%	$\boxed{93\%}$	0%
Prefix-tune on $\mathbb{H}$	0%	0%	0%	0%	0%	1%

Combine the knowledge

Effects of prefix-tuning beyond the single attention layer

Multi layer case. Bias made by prefix can exhibit increasingly complex behaviors

- Strength.
  - Prefix-tuning can change the attention pattern
- Weakness.
  - The representational capacity of the prefix-tuning is severely limited

⬇️

Prefix-tuning appears to be less expressive than full fine-tuning

even when both methods are given the same number of learnable parameters

Effects of prefix-tuning beyond the single attention layer

Prefix-tuning can change the attention

- Output of first attention(including bias): affect the attention with content
- **Limitation.** Each depends on one input position(i or j) only
  - Full fine-tuning can achieve
    $\tilde{\boldsymbol{A}}_{ij}^{\mathrm{ft}(2)} = {}^{T}/\sqrt{k}\, \boldsymbol{t}_i^{\mathrm{ft}(1)\top} (\boldsymbol{H}^{(2)} + \Delta \boldsymbol{H}^{(2)}) \boldsymbol{t}_j^{\mathrm{ft}(1)}$

$$\boldsymbol{t}_i^{\mathrm{pt}(1)} = \boldsymbol{A}_{i0}^{\mathrm{pt}(1)} \boldsymbol{W}_V \boldsymbol{s}_1^{(1)} + \sum_{j=1}^{p} \boldsymbol{A}_{ij}^{\mathrm{pt}(1)} \boldsymbol{W}_V^{(1)} \boldsymbol{x}_j^{(1)} \overset{(7)}{=} \underbrace{\boldsymbol{A}_{i0}^{\mathrm{pt}(1)}}_{\alpha_i} \boxed{\underbrace{\boldsymbol{W}_V \boldsymbol{s}_1^{(1)}}_{\boldsymbol{\mu}}} + (1 - \boldsymbol{A}_{i0}^{\mathrm{pt}(1)}) \boldsymbol{t}_i^{(1)},$$

$$\begin{aligned} \tilde{\boldsymbol{A}}_{ij}^{\mathrm{pt}(2)} &= \frac{T}{\sqrt{k}}\, \boldsymbol{t}_i^{\mathrm{pt}(1)\top} \boldsymbol{H}^{(2)} \boldsymbol{t}_j^{\mathrm{pt}(1)}, \\ &= \frac{T}{\sqrt{k}} \Big(\alpha_i \alpha_j \underbrace{\boldsymbol{\mu}^\top \boldsymbol{H}^{(2)} \boldsymbol{\mu}}_{\text{constant}} + \alpha_j (1 - \alpha_i) \underbrace{\boldsymbol{t}_i^{(1)\top} \boldsymbol{H}^{(2)} \boxed{\boldsymbol{\mu}}}_{\text{depends only on } \boldsymbol{t}_i^{(1)}} + \alpha_i (1 - \alpha_j) \underbrace{\boxed{\boldsymbol{\mu}}^\top \boldsymbol{H}^{(2)} \boldsymbol{t}_j^{(1)}}_{\text{depends only on } \boldsymbol{t}_j^{(1)}} + (1 - \alpha_i)(1 - \alpha_j) \underbrace{\boldsymbol{t}_i^{(1)\top} \boldsymbol{H}^{(2)} \boldsymbol{t}_j^{(1)}}_{\text{pretrained attention } \tilde{\boldsymbol{A}}_{ij}^{(2)}} \Big). \end{aligned}$$

Effects of prefix-tuning beyond the single attention layer

The representational capacity of the prefix-tuning is severely limited

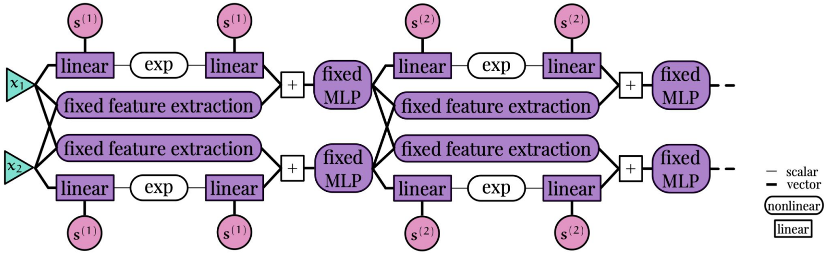
- Consider two inputs x_1, x_2 and a single prefix s (learnable parameters)
  - H, W: pre-trained parameters
  - prefix interacts with only one of the inputs at a time
    and only by computing a single scalar value
- Computational graph.
  - Only learnable interaction between the inputs is indirect
    - Interaction between the inputs happens via non-learnable residual connections

$$\mathcal{A}_{\boldsymbol{s}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle$$

$$\boldsymbol{y}_1 = \frac{\boxed{\exp(\boldsymbol{x}_1{}^\top \boldsymbol{H}\boldsymbol{s})\boldsymbol{W}_V\boldsymbol{s}} + \exp(\boldsymbol{x}_1{}^\top \boldsymbol{H}\boldsymbol{x}_1)\boldsymbol{W}_V\boldsymbol{x}_1 + \exp(\boldsymbol{x}_1{}^\top \boldsymbol{H}\boldsymbol{x}_2)\boldsymbol{W}_V\boldsymbol{x}_2}{C_1}$$

$$\boldsymbol{y}_2 = \frac{\boxed{\exp(\boldsymbol{x}_2{}^\top \boldsymbol{H}\boldsymbol{s})\boldsymbol{W}_V\boldsymbol{s}} + \exp(\boldsymbol{x}_2{}^\top \boldsymbol{H}\boldsymbol{x}_1)\boldsymbol{W}_V\boldsymbol{x}_1 + \exp(\boldsymbol{x}_2{}^\top \boldsymbol{H}\boldsymbol{x}_2)\boldsymbol{W}_V\boldsymbol{x}_2,}{C_2}$$

learnable parameter participates



Computational graph

Limitation

Not analyze about Suffix-tuning.

- Representations for prompt and soft prompt suffixes would depend on the previous positions
    - Whether suffixing is more expressive than prefixing remains an open question

Open question.

- Formal analysis of the conditions under which they may be universal approximators

Training scale. only conducting toy experiments

- In practice, language models are pre-trained with very large datasets
    - can pick up very complex behaviors
    - the limitations when using large-scale pretrained transformers with soft prompt = future work

Conclusion.

- Soft prompting are strictly more expressive than hard prompting
- Despite this expressivity, prompt tuning suffers from structural limitations
  - Have difficult learning new attention patterns
  - Only bias the output of the attention layer
- Prefix-tuning can easily elicit a skill the pretrained model already seen
  - can even learn a new task
- The effect of the prefix-induced bias is more complicated and powerful
  - still strictly less expressive than full fine-tuning