北京大學
PEKING UNIVERSITY

Label Words are Anchors：
An Information Flow Perspective for Understanding In-Context Learning

@article{wen2023hard,
title={Label Words are Anchors:
An Information Flow Perspective for Understanding In-Context Learning},
author={Wang, Lean and Li, Lei and Dai, Damai and Chen, Deli and Zhou, Hao and Meng,
Fandong and Zhou, Jie and Sun, Xu},
journal={2305.14160},
year={2023}
Conference={ENNLP}
Cites={2}
Best Paper Award

Background and Motivation

LLMs如何通过ICL来提升它们理解和生成文本的能力。LLMs可以在给定适当上下文的情况下生成有意义和相关的文本。这些模型通常需要大量的数据来训练，而且不总是能够充分理解上下文中提供的信息是如何影响输出的。

1.资源效率：希望找到更有效率的方法来利用上下文信息，减少资源消耗。
2.上下文利用：当前LLMs对上下文信息的利用还不够充分。
3. **学习过程的优化**：模型的学习过程有改进的空间，特别是在如何从示例中学习和提取有用信息的方面。

Information Flow with Labels as Anchors
$\mathcal{H}_1$: In shallow layers, label words gather the information of demonstrations to form semantic representations for deeper layers.
$\mathcal{H}_2$: In deep layers, the model extracts the information from label words to form the final prediction.

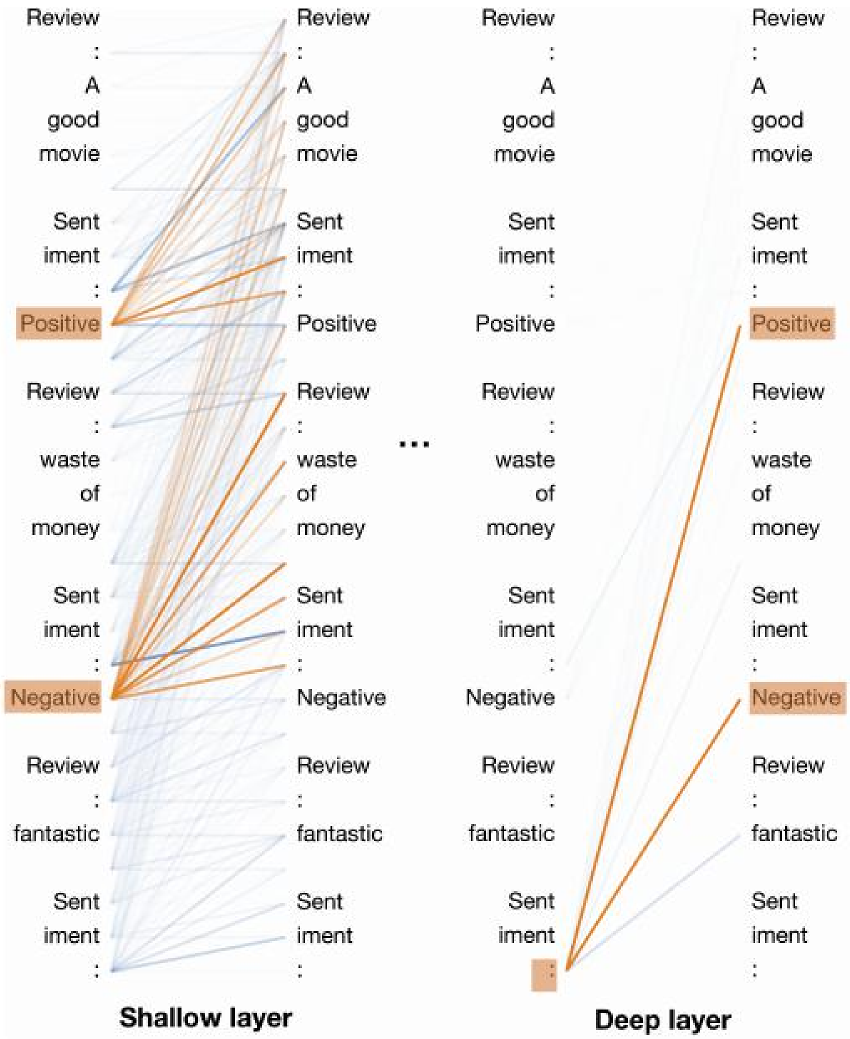


Figure 1: Visualization of the information flow in a GPT model performing ICL. The line depth reflects the significance of the information flow from the right word to

Method

作者利用显著性分数来揭示GPT模型中标记之间注意力交互的内在模式

$$I_l = \left| \sum_h A_{h,l}^\top \frac{\partial \mathcal{L}(x)}{\partial A_{h,l}} \right| .$$

S_{wp}, the mean significance of information flow from the text part to label words:

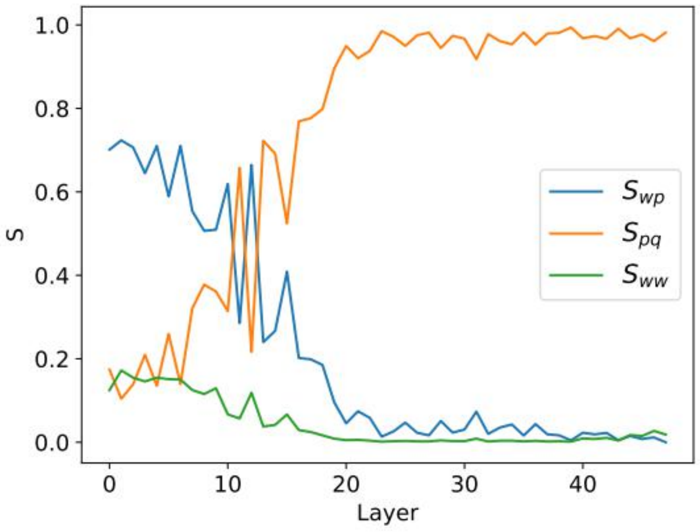
$$S_{wp} = \frac{\sum_{(i,j)\in C_{wp}} I_l(i,j)}{|C_{wp}|}, \tag{2}$$
$$C_{wp} = \{(p_k, j) : k \in [1, C], j < p_k\}.$$

S_{pq}, the mean significance of information flow from label words to the target position:

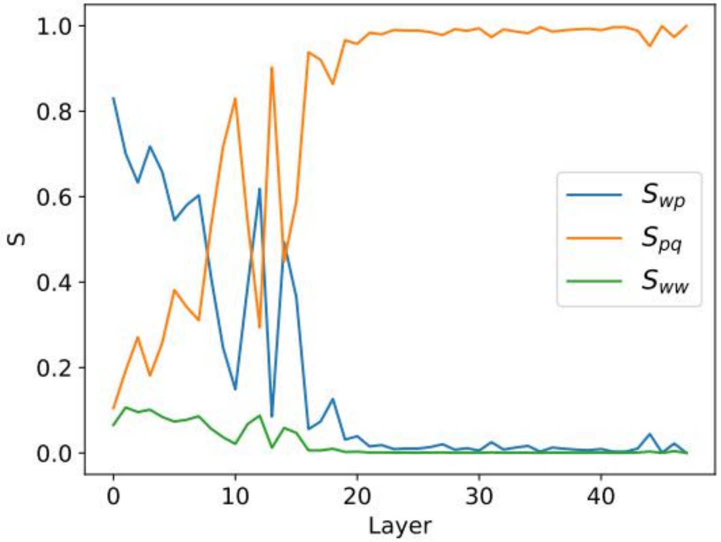
$$S_{pq} = \frac{\sum_{(i,j)\in C_{pq}} I_l(i,j)}{|C_{pq}|}, \tag{3}$$
$$C_{pq} = \{(q, p_k) : k \in [1, C]\}.$$

S_{ww}, the mean significance of the information flow amongst all words, excluding influences represented by S_{wp} and S_{pq} :

$$S_{ww} = \frac{\sum_{(i,j)\in C_{ww}} I_l(i,j)}{|C_{ww}|}, \tag{4}$$
$$C_{ww} = \{(i,j) : j < i\} - C_{wp} - C_{pq}.$$



(a) Results on the SST-2 dataset



(b) Results on the AGNews dataset

这三个指标用来评估模型中不同类型信息流动的强度和重要性。*S_wp* 评估了输入文本到标签词的信息聚合程度，*S_pq* 评估了标签词对最终决策的影响程度，而 *S_ww* 提供了一个除去上述两种特定信息流动的全局信息流动的基准

Proposed Hypothesis Based on this, we propose the hypothesis that label words function as anchors in the ICL information flow. In shallow layers, label words gather information from demonstration examples to form semantic representations for deeper layers, while in deep layers, the model extracts the information from label words to form the final prediction. Figure 2 gives an illustration for our hypothesis.

Method

Shallow Layers: Information Aggregation

验证信息聚合在ICL中的角色，特别是在模型浅层中标签词的作用。目的是展示当阻止了标签词从文本中获取信息时，模型行为的变化。

阻断信息流:通过操纵注意力矩阵 *A*,阻断到标签词的信息流 。将注意力矩阵 *A* 的第 *l* 层中，标签词 *p* 位置之前的所有位置 *i* 的注意力值设置为0，*A*(*p*,*i*) 对于所有 *i*<*p*。在第 *l* 层中，标签词无法从先前的文本中聚合信息

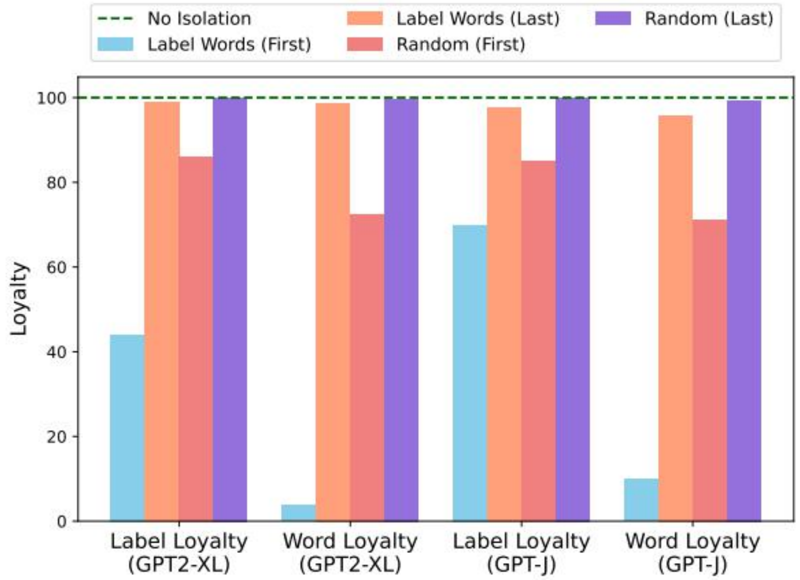


Figure 4: The impact of isolating label words versus randomly isolating non-label words within the first or last 5 layers. Isolating label words within the first 5 layers exerts the most substantial impact, highlighting the importance of shallow-layer information aggregation via label words.

当在模型的前5层隔离标签词时，模型输出的一致性显著下降，这表明浅层的信息聚合对模型预测非常重要。相比之下，隔离后5层的标签词或随机隔离非标签词对模型预测的影响较小。这进一步证实了标签词在浅层中作为信息聚合锚点的作用。

Method

Deep Layers: Information Extraction

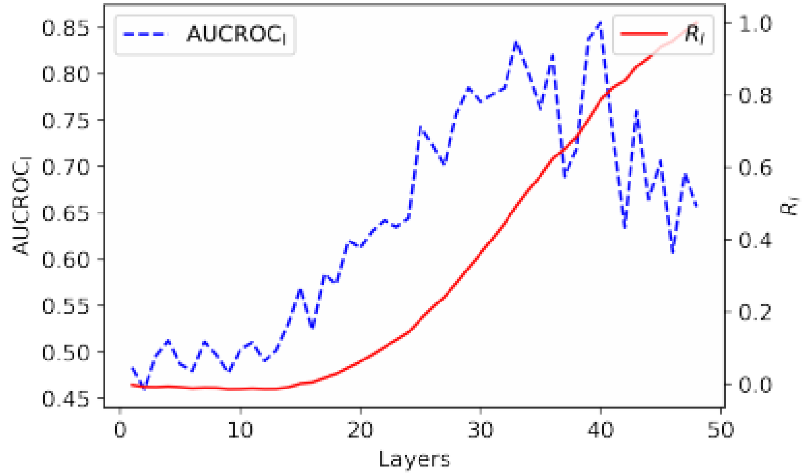
验证信息聚合在ICL中的角色，特别是在模型浅层中标签词的作用。目的是展示当阻止了标签词从文本中获取信息时，模型行为的变化。

$(A_l(q, p_1), \dots, A_l(q, p_C))$ 对最后结果的影响

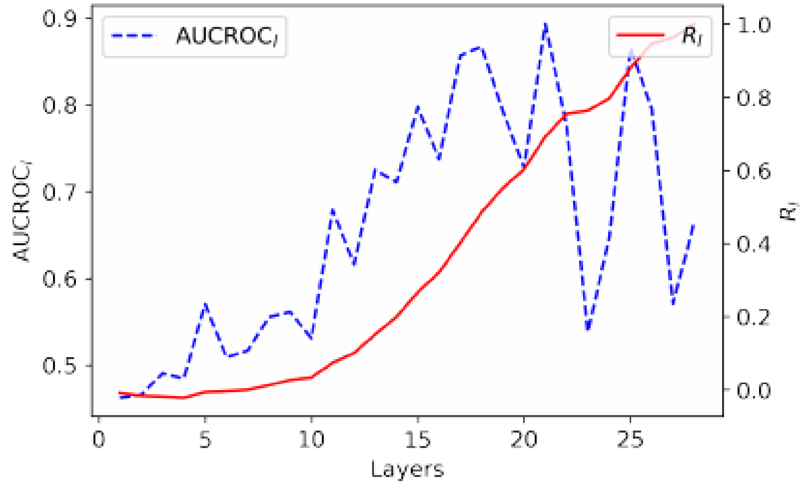
We utilize the AUC-ROC score to quantify the correlation between $A_l(q, p_i)$ and model prediction, which we denote as AUCROC_l for the l-th layer.

$$R_l = \frac{\sum_{i=1}^{l}(\mathrm{AUCROC}_i - 0.5)}{\sum_{i=1}^{N}(\mathrm{AUCROC}_i - 0.5)}.$$

量化直到第 *l* 层为止所有层对模型预测的累积贡献



(a) GPT2-XL (total 48 layers).



(b) GPT-J (total 28 layers).

Applications of Our Anchor-Based Understanding

Anchor Re-weighting:提高ICL的准确性

$$\begin{aligned} &\Pr_{f}(Y=i|X=x) \\ \approx &A(q,p_i) \\ = &\frac{\exp(\mathbf{q}_q\mathbf{k}_{p_i}^T/\sqrt{d})}{\sum_{j=1}^{N}\exp(\mathbf{q}_q\mathbf{k}_j^T/\sqrt{d})}. \end{aligned} \quad (6)$$

By setting $\mathbf{q}_q/\sqrt{d} = \hat{\mathbf{x}}$ and $\mathbf{k}_{p_i} - \mathbf{k}_{p_C} = \boldsymbol{\beta}_i$, we deduce:

$$\log\frac{\Pr_{f}(Y=i|X=x)}{\Pr_{f}(Y=C|X=x)} = \boldsymbol{\beta}_i^T\hat{\mathbf{x}}. \quad (7)$$

This approximates a logistic regression model where:

$$\log\frac{\Pr_{f}(Y=i|X=x)}{\Pr_{f}(Y=C|X=x)} = \beta_0^i + \boldsymbol{\beta}_i^T\mathbf{x}. \quad (8)$$

$$\hat{A}(q,p_i) = \exp(\beta_0^i)A(q,p_i) \quad (9)$$

To train the re-weighting vector $\boldsymbol{\beta} = \{\beta_0^i\}$, we utilize an auxiliary training set $(\boldsymbol{X}_{train}, \boldsymbol{Y}_{train})$. Here, we perform ICL with normal demonstrations and optimize $\boldsymbol{\beta}$ with respect to the classification loss $\mathcal{L}$ on $(\boldsymbol{X}_{train}, \boldsymbol{Y}_{train})$:

$$\boldsymbol{\beta}^{\star} = \arg\min_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{X}_{train}, \boldsymbol{Y}_{train}). \quad (10)$$

Method	SST-2	TREC	AGNews	EmoC	Average
Vanilla In-Context Learning (1-shot per class)	61.28	57.56	73.32	15.44	51.90
Vanilla In-Context Learning (5-shot per class)	64.75	60.40	52.52	9.80	46.87
Anchor Re-weighting (1-shot per class)	**90.07**	**60.92**	**81.94**	**41.64**	**68.64**

Table 1: The effect after adding parameter β_0^i. For AGNews, due to the length limit, we only use three demonstrations per class. Our Anchor Re-weighting method achieves the best performance overall tasks.

Method

Applications of Our Anchor-Based Understanding

Anchor-Only Context Compression:， 模型推理时会生成一系列隐藏状态，每个状态对应于输入文本中的一个词。通常情况下，模型在生成文本或推理时，会考虑所有隐藏状态。
只考虑与“锚点”相关的隐藏状态，而不是全部状态。在进行推理时只需要处理这些锚点对应的状态，加快推理速度。

pendent of subsequent words. This allows for the calculation and caching of the label word hidden states $\boldsymbol{H} = \{\{\boldsymbol{h}_l^i\}_{i=1}^{C}\}_{l=1}^{N}$ ($\boldsymbol{h}_l^i$ is the l-th layer's hidden state of the i-th label word in the demonstration). By concatenating $\boldsymbol{h}_l^1, ..., \boldsymbol{h}_l^C$ at the front in each layer during inference, instead of using the full demonstration, we can speed up inference.

Text$_{\text{anchor}}$: This method concatenates the formatting and label text with the input, as opposed to concatenating the hidden states at each layer.

Hidden$_{\text{random}}$: This approach concatenates the hidden states of formatting and randomly selected non-label words (equal in number to Hidden$_{\text{anchor}}$).

Hidden$_{\text{random-top}}$: To establish a stronger baseline, we randomly select 20 sets of non-label words in Hidden$_{\text{random}}$ and report the one with the highest label loyalty.

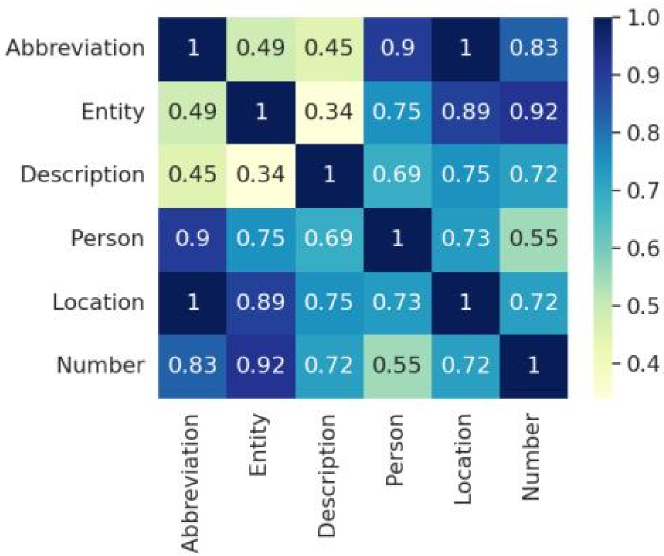
Method	Label Loyalty	Word Loyalty	Acc.
ICL (GPT2-XL)	100.00	100.00	51.90
Text$_{\text{anchor}}$	51.05	36.65	38.77
Hidden$_{\text{random}}$	48.96	5.59	39.96
Hidden$_{\text{random-top}}$	57.52	4.49	41.72
Hidden$_{\text{anchor}}$	**79.47**	**62.17**	**45.04**
ICL (GPT-J)	100.00	100.00	56.82
Text$_{\text{anchor}}$	53.45	43.85	40.83
Hidden$_{\text{random}}$	49.03	2.16	31.51
Hidden$_{\text{random-top}}$	71.10	11.36	52.34
Hidden$_{\text{anchor}}$	**89.06**	**75.04**	**55.59**

Method

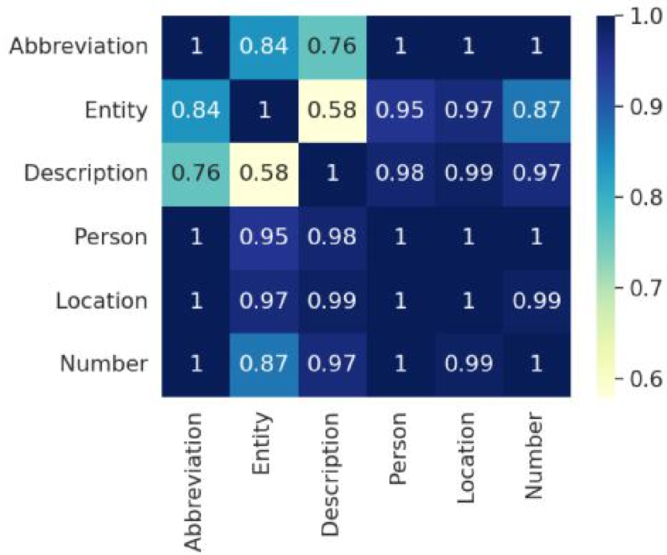
Anchor Distances for Error Diagnosis
用于诊断ICL错误的方法，是通过检查注意力模块中对应于标签词的关键向量之间的距离

$$\text{Confusion}_{ij}^{\text{pred}} = \frac{\|\hat{\mathbf{k}_{\mathbf{p_i}}} - \hat{\mathbf{k}_{\mathbf{p_j}}}\|}{\max_{s \neq t} \|\hat{\mathbf{k}_{\mathbf{p_s}}} - \hat{\mathbf{k}_{\mathbf{p_t}}}\|}, \qquad (11)$$

用了类似主成分分析（PCA）的方法来从键向量中提取与最大变异量对应的成分



(a) Confusion matrix of $\text{Confusion}_{ij}^{\text{pred}}$.



(b) Confusion matrix of Confusion_{ij}.

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

@article{
title={When Do Prompting and Prefix-Tuning Work?
A Theory of Capabilities and Limitations},
Conference={ICLR 2024}
Cites={0}
6 6 6 8

fine-tuning, soft prompt and prefix-tuning
motivations, abilities and limitations

1. **Soft prompting and prefix-tuning are motivated by the embedding space being larger than the token space. However, can a transformer utilize the additional capacity?** We show that with a careful choice of transformer weights, controlling a single embedding can generate any

2. **Since prefix-tuning is more expressive than prompting, is it as expressive as full fine-tuning?** Despite the expressiveness of continuous space, prefix-tuning has structural limitations. A prefix cannot change the relative attention over the content tokens and can only bias the output of the attention block in a constant direction. In contrast, full fine-tuning can learn new attention patterns and arbitrarily modify attention block outputs, making it strictly more powerful.

3. **If context-based fine-tuning methods suffer from such structural limitations, how come they have high empirical performance?** We show that the prefix-induced bias can steer the model towards a pretraining task. Prefix-tuning can also combine skills picked up during pretraining to solve some new tasks similar to pretraining tasks. However, it cannot learn a completely new task. This is not simply because of the small number of learnable parameters: fine-tuning the same number of parameters can be sufficient to learn the novel task. Hence, context-based fine-tuning can elicit or combine pretrained model skills but cannot learn completely new behaviors.

Method

3 Soft prompting has more capacity than prompting

Soft prompt和prefix turning是由embedding space的广度激发的，这个空间比单个token的可能完成项的空间要大。其成功的原因，主要归因于embedding space的容量大于固定token空间的容量。嵌入空间是不可数无限的，相比之下，token空间是有限的。

- 无条件生成（unconditional generation）指的是没有输入的情况下，系统token生成序列的能力。例如，使用一个单一的系统token S_1 生成序列 $(Y_1, ..., Y_N) = f_{S_1}$。
- 如果使用确定性自回归函数，一个系统token S_1 只能生成V种不同的序列，因为第一个token决定了序列的其余部分。
- 但是，如果我们使用一个由实数向量索引的虚拟单一系统token s_1，理论上可以生成所有V^N可能的输出序列，因为实数向量是无限的。

V是词汇表的大小
*N*代表生成序列的长度

有一个简单的自回归模型，它的词汇表大小是*V*，其中包含了如 “apple”, “banana”, “cherry” 等词汇。如果我们只改变序列的第一个词（第一个系统token _*S*1），并且这个词从词汇表中选取，那么整个序列的变化只能基于这个第一个词的选择。如果模型决定在 "apple" 之后总是跟着 "is red"，那么每次选择 "apple" 作为第一个词时，生成的序列就总会是 "apple is red"。既然我们只能从*V*个词汇中选择第一个词，模型最多只能生成*V*种不同的序列。

Method

这部分的结论是soft prompt和prefix turning具有比传统prompt更大的表现力。

尽管可以用virtual token完全决定从用户输入到模型响应的映射，这可能给人一种错觉，认为soft prompt的能力与full fine-tuning一样强大。

但是。 soft prompt和full fine-tuning存在结构上的限制，它们无法促进学习一个全新任务的能力。接下来的部分将阐明差异。

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 $(Y_1, \ldots, Y_N) \in \{1, \ldots, V\}^N$ when conditioned on the single virtual token $\boldsymbol{s}_1 = ((Y_1-1)/V, \ldots, (Y_N-1)/V)$.*

from X_1 to Y_1, but S_1 can take on only V values: $|\{f_{S_1} : S_1 \in 1, \ldots, V\}| = V < V^V$. Hence, tokens cannot be used to specify an arbitrary map from user input to model output. However, a single virtual token can specify any of the V^V maps, i.e., there exists a transformer $f_{\boldsymbol{s}_1}(X_2)$ for which there is a surjective map from $\{f_{\boldsymbol{s}_1} : \boldsymbol{s}_1 \in \mathbb{R}^{d_e}\}$ to $\{1, \ldots, V\}^{\{1, \ldots, V\}}$.

定理1说明了使用单一virtual token的transformer模型在理论上具有强大的生成能力，即使在条件生成的场景下，这个能力仍然非常强大，能够映射出复杂的输入到输出序列的关系。

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 $m:[1, \ldots, V] \to [1, \ldots, V]$ from a user input token to a model response token when conditioned on a single virtual token $\boldsymbol{s}_1 = (m(1)/V, \ldots, m(V)/V)$. That is, by selecting $\boldsymbol{s}_1$ we control the model response to any user input.*

定理2扩展了定理1的概念。不仅是对于单个响应token，而且对于更长的响应序列，通过增加soft prompt的长度来增加用户输入的长度， soft prompt显示出更大的表现力。

4 PREFIX-TUNING CAN ONLY BIAS THE OUTPUT OF AN ATTENTION HEAD

While full fine-tuning can alter the attention pattern of an attention head, prefix-tuning cannot. Recall the attention A_{ij} position i gives to position j for a trained transformer (Equation (1)):

$$A_{ij} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{W}_Q^\top \boldsymbol{W}_K \boldsymbol{x}_j\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{W}_Q^\top \boldsymbol{W}_K \boldsymbol{x}_r\right)} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_j\right)}{\sum_{r=1}^{p} \exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r\right)}, \tag{5}$$

where $\boldsymbol{W}_Q^\top \boldsymbol{W}_K = \boldsymbol{H}$. Full fine-tuning can enact arbitrary changes to $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ and hence, assuming the input does not change (e.g., at the first attention layer), we get the following attention:

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

where the changes to $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ are folded into $\Delta\boldsymbol{H}$. It is clear that by varying $\Delta\boldsymbol{H}$ full fine-tuning can change the attention patterns arbitrarily. However, let us see how is attention affected by

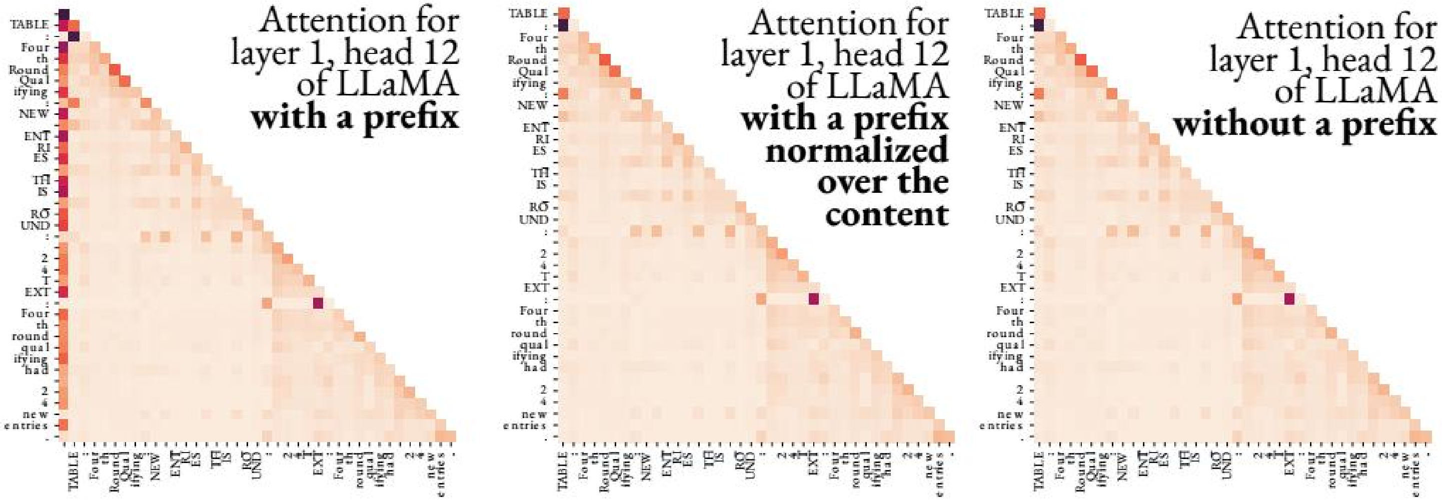
$$A_{i0}^{\text{pt}} = \frac{\exp\left(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1\right)}{\exp\left(\frac{T}{\sqrt{k}}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1\right) + \sum_{r=1}^{p} \exp\left(\frac{T}{\sqrt{k}}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{x}_r\right)},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

The numerator of A_{ij}^{pt} is the same as in Equation (5), i.e., the prefix does not affect it. It only adds the term $\exp(T/\sqrt{k}\, \boldsymbol{x}_i^\top \boldsymbol{H} \boldsymbol{s}_1)$ to the denominator. Therefore, the attention position i gives to the content positions $j \geq 1$ is simply scaled down by the attention it now gives to the prefix. If *tomato* attends the most to *salad* in a particular context, no prefix can change that. This becomes evident by rewriting A_{ij}^{pt} as the attention of the pretrained model scaled by the attention "stolen" by the prefix:

前缀调整相对于完整微调，它只能在模型现有的注意力框架内引入偏差，而不能创建新的注意力模式，只能在原有的注意力模式上加权或减权

Method

4 PREFIX-TUNING CAN ONLY BIAS THE OUTPUT OF AN ATTENTION HEAD



Prefix-tuning only adds a bias to the attention block output. Let us see how this attention scaling down affects the output of the attention block. Following Equation (2), the output at position i for the pretrained ($\boldsymbol{t}_i$), the fully fine-tuned ($\boldsymbol{t}_i^{\text{ft}}$) and the prefix-tuned ($\boldsymbol{t}_i^{\text{pt}}$) models are as follows:[5]

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

$$\boldsymbol{t}_i^{\text{pt}} = A_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1 + \sum^{p} A_{ij}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{x}_j \overset{(6)}{=} A_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1 + \sum^{p} A_{ij} (1 - A_{i0}^{\text{pt}}) \boldsymbol{W}_V \boldsymbol{x}_j = A_{i0}^{\text{pt}} \boldsymbol{W}_V \boldsymbol{s}_1 + (1 - A_{i0}^{\text{pt}}) \boldsymbol{t}_i. \tag{7}$$

Longer prefixes define larger subspaces for the bias but are not fully utilized in practice

有更长的前缀和偏置空间:
更长的前缀会定义更大的偏置空间，但实际上并未完全利用。偏置空间并不是被前缀完全占据的。尽管理论上前缀可以定义一个很大的空间，实际中前缀的注意力分布并不依赖于输入内容，它们主要作为偏置存在

Method

5 THE BIAS CAN ELICIT SKILLS FROM THE PRETRAINED MODEL

Prefix-tuning cannot learn a new task requiring a different attention pattern,
prefix-tuning indeed cannot learn a new task if it requires new attention patterns.

Table 1: A transformer pretrained on sorting in ascending order cannot be prefix-tuned to sort in descending order. 10 random seeds.

	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%

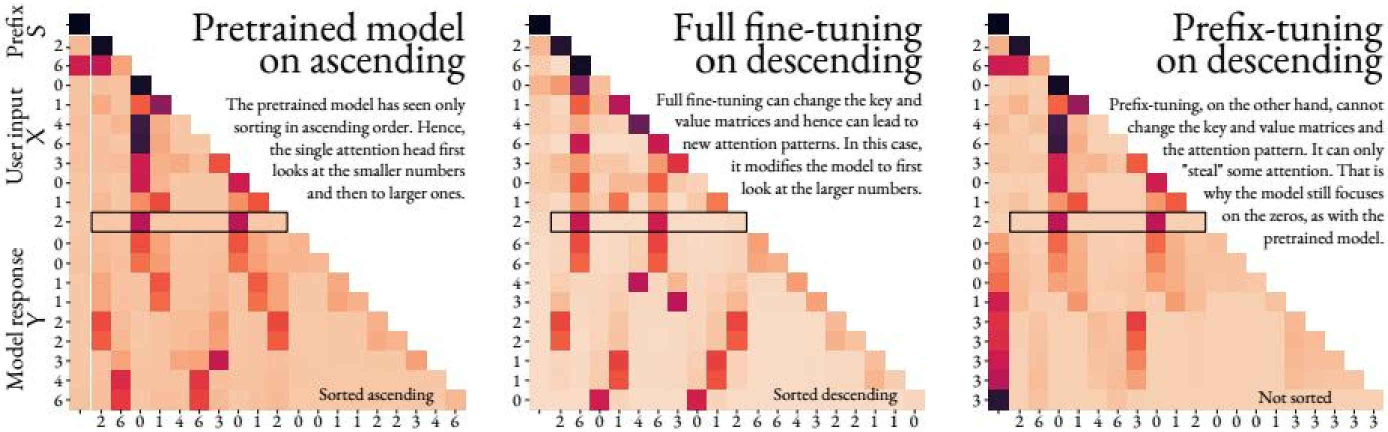


Figure 1: Attention patterns of a small transformer pretrained on sorting in ascending order. The model is given the prefix S and user input X and generates Y autoregressively. We have highlighted the attention when the first response $\mathtt{Y}_1$ is being generated. Full fine-tuning sorts in descending order but prefix-tuning cannot as it cannot update the learned attention. Note how the relative attention of X to X in the left and right plots is exactly the same: the prefix cannot change the attention pattern for the same inputs. The relative attention of X to X in the center plot is very different because full fine-tuning can arbitrarily change $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$.

Method

5 THE BIAS CAN ELICIT SKILLS FROM THE PRETRAINED MODEL

Prefix-tuning can elicit a skill from the pretrained model.

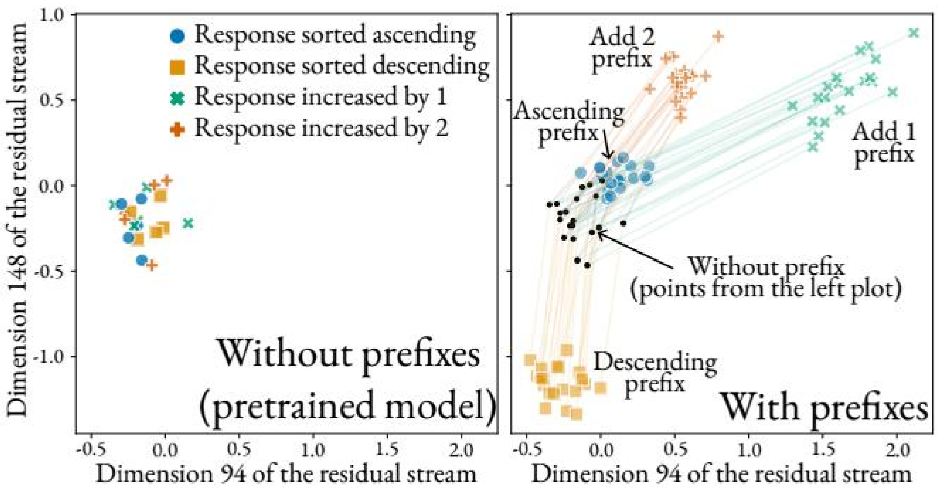


Table 2: A transformer pretrained on several tasks can be prefix-tuned for one of them. 10 random seeds.

Accuracy on:	$\nearrow$	$\searrow$	+1	+2
Pretrained	25±13%	25±12%	24±11%	22±7%
Prefix-tune on $\nearrow$	95± 2%	0± 0%	0± 0%	0±0%
Prefix-tune on $\searrow$	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%

Table 3: Prefix tuning can learn a new task requiring only pretraining skills ($\nearrow$+1) but cannot learn a completely new task ($\mathbb{H}$). Average accuracy over 3 seeds.

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%	93%	0%
Prefix-tune on $\mathbb{H}$	0%	0%	0%	0%	0%	1%

Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery

@article{wen2023hard,
title={Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery},
author={Wen, Yuxin and Jain, Neel and Kirchenbauer, John and Goldblum},
journal={arXiv:2302.03668},
year={2023}
Conference={Nips}
Cites={47}

Background and Motivation

黑盒prompt 优化

Hard prompts can be automatically discovered that are effective in tuning LMs for classification.

Algorithm 1 Hard **P**rompts made **E**a**Z**y: PEZ Algorithm

Input: Model θ, vocabulary embedding $\mathbf{E}^{|V|}$, projection function Proj, broadcast function $\mathcal{B}$, optimization steps T, learning rate γ, Dataset D

Sampled from real embeddings:

$\mathbf{P} = [\mathbf{e_i}, ... \mathbf{e_M}] \sim \mathbf{E}^{|V|}$

for $1, ..., T$ **do**

Retrieve current mini-batch $(X, Y) \subseteq D$.

Forward Projection:

$\mathbf{P}' = \text{Proj}_{\mathbf{E}}(\mathbf{P})$

Calculate the gradient w.r.t. the *projected* embedding:

$g = \nabla_{\mathbf{P}'} \mathcal{L}_{\text{task}}(\mathcal{B}(\mathbf{P}', X_i), Y_i, \theta)$

Apply the gradient on the *continuous* embedding:

$\mathbf{P} = \mathbf{P} - \gamma g$

end for

Final Projection:

$\mathbf{P} = \text{Proj}_{\mathbf{E}}[\mathbf{P}]$

return P

$$\mathcal{L} = (1 - \lambda_{\text{fluency}})\mathcal{L}_{\text{task}} + \lambda_{\text{fluency}}\mathcal{L}_{\text{fluency}}.$$

Method	GPT-2 Large (755M, **Source**)	GPT-2 XL (1.3B)	T5-LM-XL (3B)	OPT (2.7B)	OPT (6.7B)
$\text{Empty}_{\text{Template}}$	80.84	73.85	52.75	72.48	58.72
$\text{AutoPrompt}_{\text{SGD}}$	$87.56_{\pm 0.35}$	$78.19_{\pm 2.68}$	$56.01_{\pm 1.67}$	$73.69_{\pm 1.63}$	$65.28_{\pm 1.75}$
FluentPrompt	$\mathbf{88.33}_{\pm 0.35}$	$78.53_{\pm 2.82}$	$55.64_{\pm 0.59}$	$70.39_{\pm 2.08}$	$61.74_{\pm 1.25}$
$\text{PEZ}_{\text{No Fluency}}$ (Ours)	$88.12_{\pm 0.15}$	$77.8_{\pm 3.45}$	$61.12_{\pm 2.94}$	$76.93_{\pm 1.29}$	$71.72_{\pm 3.16}$
$\text{PEZ}_{\text{Fluency}}$ (Ours)	$88.05_{\pm 0.55}$	$\mathbf{79.72}_{\pm 3.26}$	$\mathbf{63.30}_{\pm 2.30}$	$\mathbf{77.18}_{\pm 3.82}$	$\mathbf{72.39}_{\pm 1.82}$