Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

> Submitted on **28 Jul 2021**

综述文章太长不看版→知乎讲解，下面仅记录一些比较重要的综述内容

1 四种范式：

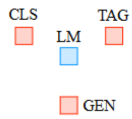
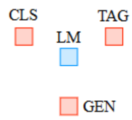
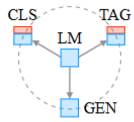
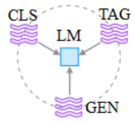
Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The "**engineering**" column represents the type of engineering to be done to build strong systems. The "**task relation**" column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). : fully unsupervised training. : fully supervised training. : Supervised training combined with unsupervised training. indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. "LM→Task" represents *adapting LMs (objectives) to downstream tasks* while "Task→LM" denotes *adapting downstream tasks (formulations) to LMs*.

1. **Feature-Engineering(特征工程)：**纯有监督学习为主，需要一定规模的标注数据，然后学习模型参数，再基于模型对新的句子进行解码inference
2. **Architecture-Engineering(架构工程)：**以设计新的神经网络模型为主的有监督学习
3. **Objective-Engineering(目标工程)：**以设计新的预训练任务为代表
4. **Prompt-Engineering(提示工程)：**比如**填空**、**前缀**等等，可以诱发/检索出大模型中所含有的实际任务所需要的

2 **Prompting Methods：**

Name	Notation	Example	Description
Input	$\boldsymbol{x}$	I love this movie.	One or multiple texts
Output	$\boldsymbol{y}$	++ (very positive)	Output label or text
Prompting Function	$f_{\text{prompt}}(\boldsymbol{x})$	[X] Overall, it was a [Z] movie.	A function that converts the input into a specific form by inserting the input $\boldsymbol{x}$ and adding a slot [Z] where answer $\boldsymbol{z}$ may be filled later.
Prompt	$\boldsymbol{x}'$	I love this movie. Overall, it was a [Z] movie.	A text where [X] is instantiated by input $\boldsymbol{x}$ but answer slot [Z] is not.
Filled Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot [Z] is filled with any answer.
Answered Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot [Z] is filled with a true answer.
Answer	$\boldsymbol{z}$	"good", "fantastic", "boring"	A token, phrase, or sentence that fills [Z]

Table 2: Terminology and notation of prompting methods. $\boldsymbol{z}^*$ represents answers that correspond to true output $\boldsymbol{y}^*$.

1 **Prompting Function(提示函数)**：$f_{prompt}(\cdot)$负责把一个输入文本x变换为一个prompt x'，即为$x' = f_{prompt}(x)$

首先应用一个“模板”，其中该模板应包含一个输入 slot[X] 和一个答案 slot[Z] ， [Z] 最后会被映射给最后的输出y

然后用输入文本x填充 slot[X]

> [Z] 不在句子末尾的称为**cloze prompt(填空型提示)**； [Z] 在末尾的称为**prefix prompt(前缀型提示)**

2 **填充函数**：$f_{fill}(x', z)$负责用一个候选答案z来填充prompt x'中的 [Z] ，得到的prompt称为filled prompt

3 **Answer Search(答案搜索)**：$\hat{z} = search_{z \in Z} P(f_{full(x',z);\theta})$，其中$P$即为PLM对prompt打分得到的概率。

4 **Answer Mapping**：最后将得到的$\hat{z}$映射到任务定义的y

3 代表性**NLP**任务：

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. ...

Table 3: Examples of *input*, *template*, and *answer* for different tasks. In the **Type** column, "CLS" is an abbreviation for "classification". In the **Task** column, "NLI" and "NER" are abbreviations for "natural language inference" (Bowman et al., 2015) and "named entity recognition" (Tjong Kim Sang and De Meulder, 2003) respectively.

1 **Text CLS(文本分类任务)**：该任务还可细分成**sentiment(情感分析)、Topics(文本"主题"分类任务)、Intention(意图识别)**

2 **Text-span CLS(文本片段分类任务)**

3 **Text-pair CLS(文本对自然语言推理分类任务)**

4 **Tagging(序列标注任务)**

5 **Text Generation(文本生成任务)**：该任务可细分为**Summarization(总结)、Translation(翻译)**

4 **Prompt Engineering(提示工程)**：分为**discrete prompts(离散提示)、continuous prompts(连续提示)**

1 **discrete prompts(离散提示)**：使用具体的words/tokens

prompt mining(提示挖掘)

prompt paraphrasing(提示改述)，例如$English \to Chinese \to English$这种

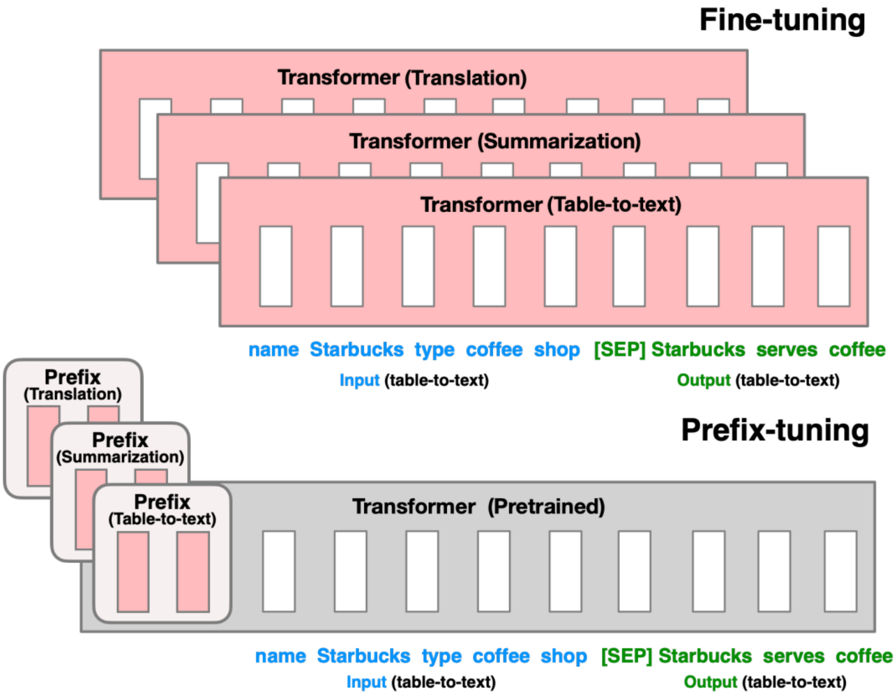
Gradient-based Search(基于梯度的搜索)

Prompt Generation(提示生成)

Prompr Scoring(提示打分)

2 **continuous prompts(连续提示)**：基于embeddings来表示prompts，prompts拥有自己的参数可以微调

Prefix-Tuning(前缀微调)



在每个句子上加若干前缀，遇到新下游任务就修改prefix

Tuning Initialized with Discrete Prompts(先离散后连续)

Hard-Soft Prompts Hybrid Tuning(离散+连续)，例如将可微调的embeddings放入一个hard(离散的)prompt template

Extensible Prompts for Language Models on Zero-shot Language Style Customization

Submitted on **1 Dec 2022**, last revised **30 Nov 2023**

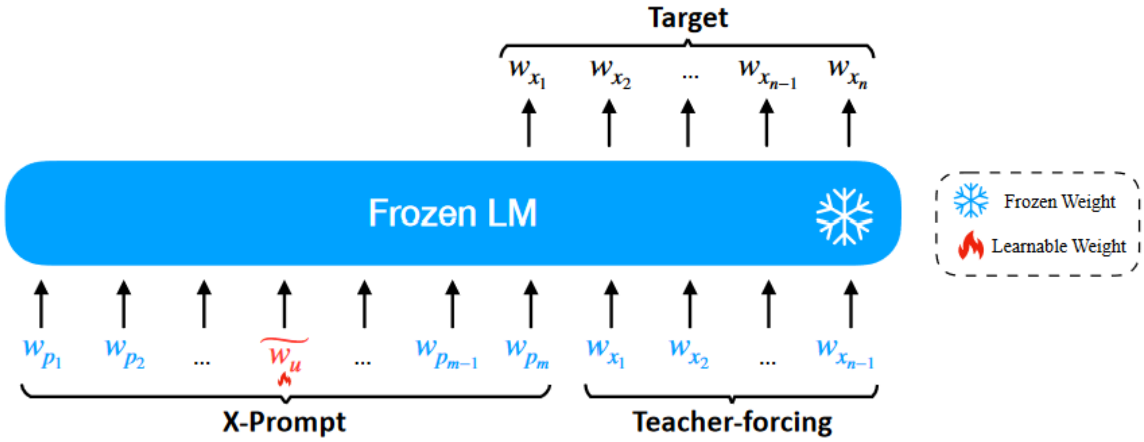
1 Introduction

提出了**eXtensible Prompt(X-Prompt)**，将**imaginary words**$\rightarrow \tilde{w}$注入**NL(natural language)**中构成**X-Prompt**，可以用于解决OOD robustness问题。同时提出**context-augmented learning(CAL)**的概念，更好地学习**imaginary words**$\rightarrow \tilde{w}$，保证其general usability

task的类型是language style customization

2 eXtensible Prompt

X-Prompt: $(w_{p_1}, \cdots, w_{p_m})$，每个$w_{p_i}$可以来自NL vocabulary V或者extensible imaginary word vocabulary $\tilde{V}$

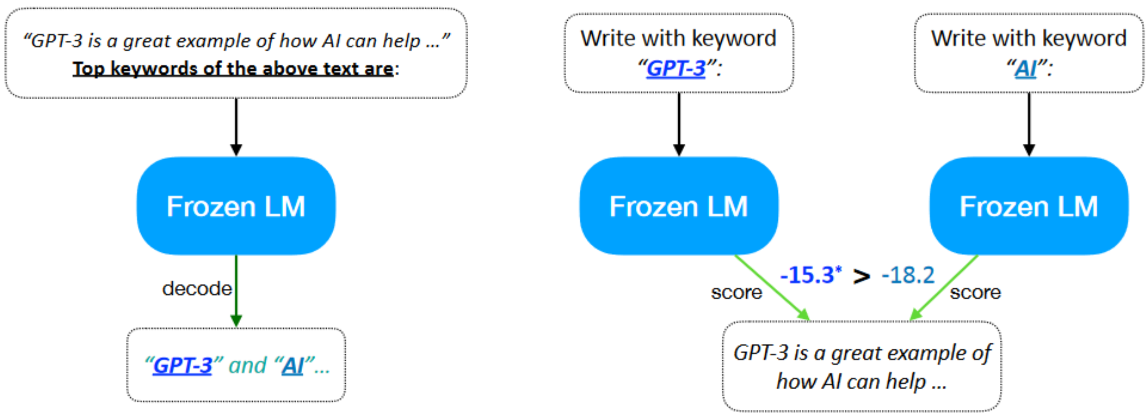


Context-augmented learning：假设**X-Prompt**为$(w_{p_1}, \cdots, \widetilde{w_{p_u}}, \cdots w_{p_m})$，那么学习**imaginary words**$\widetilde{w_{p_u}}$也即最大化$\log P(\vec{x}|w_{p_1}, \cdots, w_{p_m})$，其中$\vec{x}$为training example$(w_{x_1}, \cdots, w_{x_n})$

Template augmentation：给定T个**X-Prompt**，$\{(w_{p_1}^{(t)}, \cdots, \widetilde{w_u}, \cdots, w_{p_{m_t}}^{(t)})|1 \le t \le T\}$，需要最大化$\frac{1}{T}\sum_{t=1}^{T} \log P(\vec{x}|w_{p_1}^{(t)}, \cdots, \widetilde{w_u}, \cdots, w_{p_{m_t}}^{(t)})$

Content augmentation：向**X-Prompt**中注入an indicative keyword，对于keyword的处理如下图。

对每个training example$\vec{x}$提取出keyword candidates$[w_k^1, \cdots, w_k^C]$，然后每个keyword插入到一个用于rank的prompt中选出最indicative的一个keyword，也即$w_k^\star = \arg\max_{w_k^c} \log P(\vec{x}|\vec{r}(w_k^c))$，其中$\vec{r}(w_k^c) = (w_{p_1}^{(r)}, \cdots, w_{p_{m_r}}^{(r)})$称为**ranking prompt template**



(a) Keyword extraction　　(b) Keyword ranking

3 Experiments

实验主要是测试了两个task：**open-ended text generation**、**style transfer(rewriting)**，前一个任务是测试**X-Prompt**如何instruct语言模型生成user-specific语言，后一个任务是按要求转换语言的style(比如$impolite \rightarrow polite$
)

open-ended text generation：

1. 数据集： Top 20 most followed users in Twitter social platform dataset + the Sentiment dataset5 from which we extract top 800 users’ (in total 68K) tweets (800-user dataset)，同时剔除了test example中与training example具有相同indicative keyword的样本
2. 基本配置：base model为 OPT-6.7b ，选用 Adam 优化器……
3. 定量评估：选取的指标为 perplexity 和 accuracy ，实验结果如下图。**X-Prompt**在OOD上表现得更加好，而**Prompt tuning**、**X-Prompt(w/o CAL)**在ID上表现更好(因为它们focus on training examples)

Table 6: Quantitative evaluation results in 800-user and 20-user datasets. **No prompt** denotes the original OPT-6.7b baseline without any prompt and k**-shot** denotes a baseline which prepends k examples from a user's training set as a prompt for customizing this user's style.

Method	800 Users (ID)		20 Users (ID)		20 Users (OOD)	
	PPL↓	Accuracy↑	PPL ↓	Accuracy ↑	PPL ↓	Accuracy ↑
No prompt	73.2	27.1	38.9	34.8	37.7	35.2
8-shot	69.9	27.2	36.0	35.0	-	-
16-shot	68.9	27.5	35.5	35.3	-	-
32-shot	62.7	28.5	34.0	36.4	-	-
Prompt tuning	56.0	29.5	29.9	**37.8**	29.5	38.0
X-Prompt	56.2	29.3	30.8	37.2	**28.5**	**38.6**
X-Prompt (*w/o* CAL)	**55.7**	**29.9**	**29.7**	37.7	29.4	37.9

1. 定性评估：手工构建了100个在training阶段没有见过的prompt，然后让两个人对LM生成结果在三个方面做评估：Content、Style、Overall，实验结果如下图

Table 7: Human evaluation of generated texts in content, style and overall quality dimensions.

Prompt Method	Content↑	Style↑	Overall↑
NL	**0.79**	0.33	0.22
Prompt tuning	0.34	0.92	0.30
X-Prompt (*w/o* CAL)	0.38	**0.93**	0.35
X-Prompt	0.69	0.83	**0.54**

style transfer：

1. 数据集： Entertainment (EM) subset of GYAFC (informal → formal) + POLITEREWRITE (impolite → polite)
2. 评测指标： BLEU(Bilingual Evaluation Understudy)，该指标常用于机器翻译用于评测生成结果和reference的lexical similarity、 accuracy 用于评测style appropriateness、 harmonic(H-) mean 和 geometric(G-) mean 用来作为overall performance

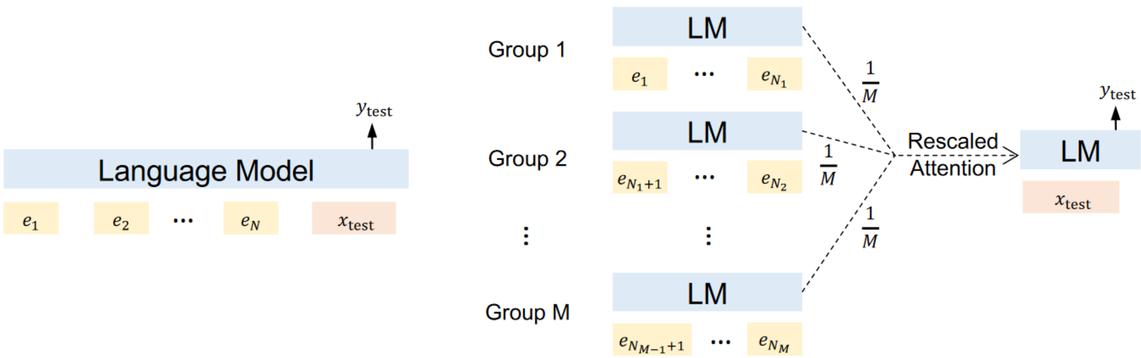
Structured Prompting: Scaling In-Context Learning to 1,000 Examples

Submitted on **13 Dec 2022**

1 Introduction

> 提出了**Structured Prompting**，打破length的限制从而使得**In-Context Learning**可以用成千的examples训练。
>
> **In-Context Learning**：对于N-shot in-context learning，给定一个N labeled examples $D_{train}=\{(x_i,y_i)\}_{i=1}^{N}$，每个数据点都可用hand-crafted template T转化成一个demonstration $d_i=T(x_i,y_i)$。所有的demonstration可以被连接成$Z=d_1\oplus\cdots\oplus d_N$，对于每个test input x_{test}，都可以构造prompt为Z和x_{test}的连接。最终的输出结果为$\arg\max_{c\in Y} P_{LM}(y^c|Z\oplus T(x_{test}))$，其中$Y$是所有可能的candidate

2 Methods



(a) Conventional Prompting **(b) Structured Prompting**

Group Context Encoding：假设有N demonstration examples，把这些examples随机分成M组 $\{Z_i\}_{i=1}^{M}$，每个group为$Z_i=d_{N_{i-1}+1}\oplus\cdots\oplus d_{N_i}$，其中$N_0=0$、$N_M=N$

> Position Embedding：所有group采用right align从而保证它们有相同的max position index，因此所有group到test input有相同的距离。为了让test input对所有的examplar adjacent并pay equal attention，有两种方式：1. 使用left padding，i.e. pad tokens or space tokens 2.设置最大长度，从左边truncate examplar

Structured Prompting：所有的examplar都喂进了Rescaled Attention，并连同test input一起喂进LM

> Rescaled Attention：每一层都将所有examplar和test input的key、value连接起来，即$\hat{K}=[K_{Z_1},\cdots,K_{Z_M},K_x]$、$\hat{V}=[V_{Z_1},\cdots,V_{Z_M},V_x]$。计算attention

的公式为：

$$\text{Attention}(Q_x, \hat{K}, \hat{V}) = A\hat{V} \tag{1}$$

$$A_{ij} \propto \begin{cases} M\exp(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}) & j \in x \\ \exp(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}) & j \in \mathcal{Z}_1, \ldots, \mathcal{Z}_M \end{cases} \tag{2}$$

where $\sum_j A_{ij} = 1$, the query vector $\boldsymbol{q}_i \in Q_x$, the key vector $\boldsymbol{k}_j \in \hat{K}^\intercal$, and d is dimension of queries and keys.

3 Experiments

1. 模型：`GPT-like(decoder-only Transformer)`，对于超大模型实验，选用 `BLOOM-176B`
2. 数据集：根据三个task：text classification、multi-choice、open-ended generation从而有对应的数据集

> 整个实验测试了Model Size为**1.3B**、**6.7B**、**13B**在三个task `text classification`、`multi-choice`、`open-ended generation`，然后又在超大模型 `BLOOM-176B` 上测试了上面三个task，最后进行消融实验验证Prompt Length、Scaling Factor、Alignment Strategy的重要性

How Does In-Context Learning Help Prompt Tuning

> Submitted on **22 Feb 2023**

1 Introduction

> 本文主要比对了 `PT(Prompt Tuning)`、`ICL(In-Context Learning)`、`IPT(Instruction Prompt Tuning)` 从而来探究ICL对PT的影响
>
> 别的地方看到的：**in-context examples**主要是帮助**model**学习**output label space**和**distribution of input text**

2 Background

`In-Context Learning`：在test input之前插入k个in-context input-output pairs，即为 $Input_{ICL} = concat([X_{icl}; Y_{icl}]_1^k; X_{test})$

`Prompt Tuning`：在test input X_{test}之前加入soft tunable prompt embeddings。一系列的tunable prompt embeddings用$E = \{e_1, \cdots, e_k\}$表示，那么$Input_{PT} = concat(E; X_{test})$。注意这里的$E$需要train，所以需要$X_{train}$、$Y_{train}$

Instruction Prompt Tuning：把soft prompts和hard in-context demonstrations连接在一起，即为$Input_{IPL} = concat(E; [X_{icl}; Y_{icl}]_1^k; X_{test})$

3 Experiments

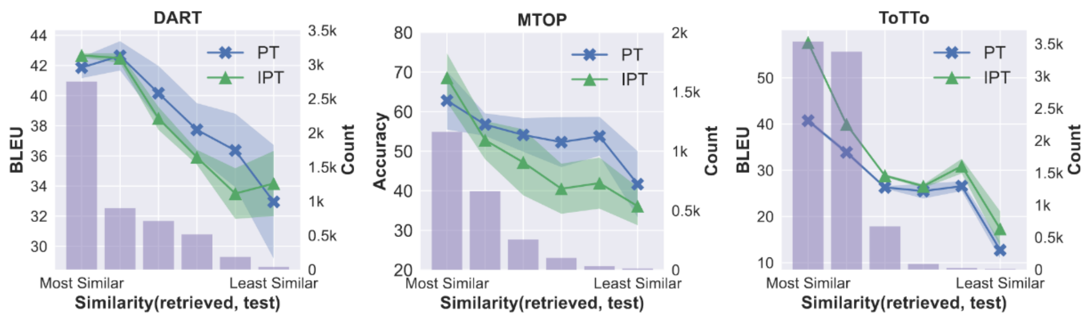
1. 数据集：选用三种language generation tasks，即为 data-to-text generation、logic-to-text generation、semantic parsing。不同任务有相应的数据集
2. 模型：BLOOM-1.1B、OPT-1.3B、GPT-2-XL-1.5B
3. **实验结论(这部分可以参考)**

	ToTTo (BLEU)	Dart (BLEU)	Spider (Exact Match)	Mtop (Exact Match)	Logic2text (BLEC)
BLOOM-1.1B					
random one-shot ICL	5.8	8.3	0.4	0.0	37.6
retrieved one-shot ICL	35.1	23.9	3.9	18.5	70.1
retrieve three-shot ICL	41.3	29.7	5.0	12.7	71.0
BLOOM-1.1B					
Prompt Tuning	$36.3_{\pm 0.3}$	$41.2_{\pm 0.9}$	$35.5_{\pm 1.6}$	$25.2_{\pm 16.4}$	$87.6_{\pm 1.5}$
Instruction Prompt Tuning	$47.1_{\pm 0.2}$	$41.4_{\pm 0.1}$	$33.2_{\pm 1.1}$	$62.6_{\pm 0.7}$	$86.4_{\pm 1.1}$
OPT-1.3B					
Prompt Tuning	$38.5_{\pm 1.0}$	$44.5_{\pm 0.2}$	$14.4_{\pm 2.3}$	$6.4_{\pm 6.5}$	$80.6_{\pm 3.7}$
Instruction Prompt Tuning	$46.3_{\pm 0.9}$	$42.9_{\pm 0.4}$	$14.2_{\pm 2.1}$	$10.4_{\pm 6.5}$	$84.6_{\pm 1.0}$
GPT-2-XL-1.5B					
Prompt Tuning	$37.3_{\pm 0.2}$	$43.5_{\pm 0.2}$	$27.0_{\pm 2.1}$	$41.4_{\pm 5.6}$	$87.2_{\pm 1.6}$
Instruction Prompt Tuning	$48.0_{\pm 0.0}$	$42.1_{\pm 0.2}$	$23.0_{\pm 0.1}$	$19.8_{\pm 14.9}$	$85.8_{\pm 1.5}$

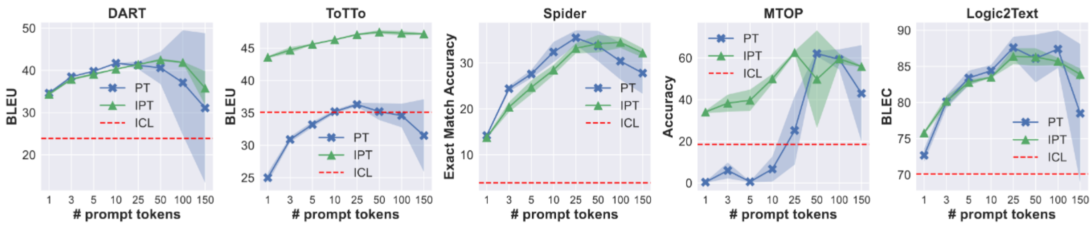
ICL表现得比PT差，这说明了对于类似OOD generation task，针对target task train一小部分参数是有价值的

PT、IPT的表现难分伯仲，取决于task类型和tunable parameter的数目等(work不work可能也还有数据集等因素)

当demonstration跟test input类似的时候，IPT可以work。这也就说明了similar demonstration对IPT的重要性



IPT在有更多的soft prompt tokens的情况下比PT表现得更稳定



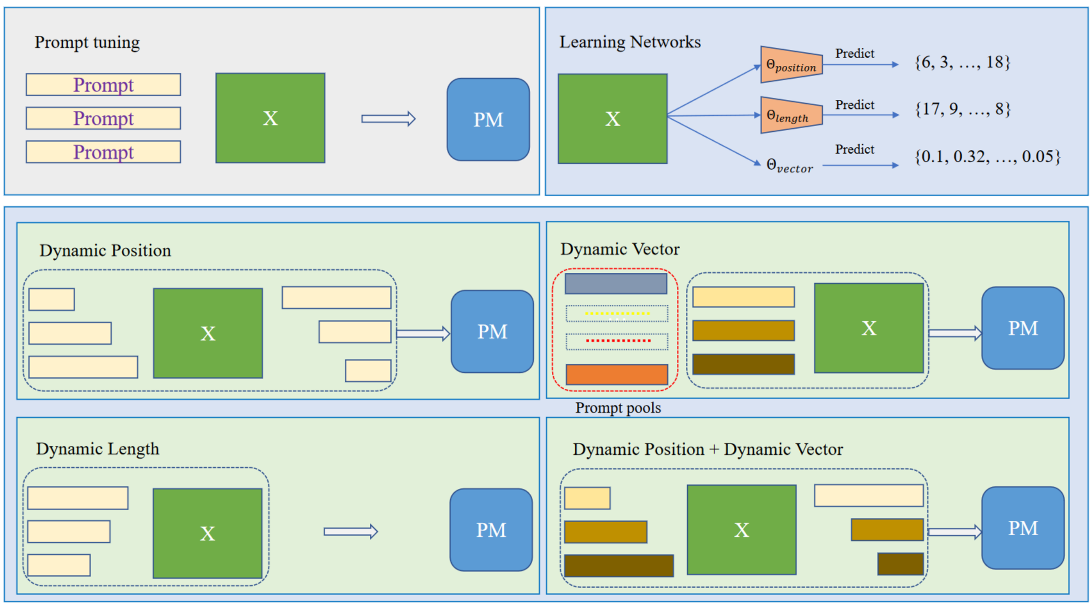
在有**in-context demonstrations**的情况下，**Prompt embeddings**对于新的**task**是**transferable**

Dynamic Prompting: A Unified Framework for Prompt Tuning

Submitted on **6 Mar 2023**, last revised **27 May 2023**

1 Introduction

提出了**DP(Dynamic Prompt)**，根据不同的instance/task来调整相对应prompt的**position**、**length**、**representation**(例如不同position相比传统的prefix/postfix可能会更好地捕捉语义信息)。**DP**的整体架构如下：



2 Methods

Unified View：把prompt分为prefix和postfix两部分，对于输入$x \in R^{m\times d}$，query matrix是$Q = xW^W \in R^{m\times d}$ key matrix是$K = xW^K \in R^{m\times d}$ value matrix是$V = xW^V \in R^{m\times d_v}$。假设prompt的长度为$l$，那么$P = [P_1; P_2]$，其中$P_1 \in R^{l_1\times d}, P_2 \in R^{l_2\times d}$。最终的输入变成$x' = [P_1; x; P_2] \in R^{(l_1+m+l_2)\times d}$，新的key matrix变为$K' = x'W^K \in R^{(l_1+m+l_2)\times d}$ value matrix变为$V' = x'W^V \in R^{(l_1+m+l_2)\times d_v}$

。通过矩阵分解：$Q' = \begin{bmatrix} Q_1 \\ Q \\ Q_2 \end{bmatrix}, K' = \begin{bmatrix} K_1 \\ K \\ K_2 \end{bmatrix}, V' = \begin{bmatrix} V_1 \\ V \\ V_2 \end{bmatrix}$，其中 $Q_1, K_1 \in R^{l_1 \times d}, Q_2, K_2 \in R^{l_2 \times d}, V_1 \in R^{l_1 \times d_v}, V_2 \in R^{l_2 \times d_v}$。因此对于输入 $x' = [P_1; x; P_2]$来说，attention head module变为 $Head = Attn([P_1; x; P_2]W^Q, [P_1; x; P_2]W^K, [P_1; x; P_2]W^V) = softmax(\frac{Q'K'^T}{\sqrt{d}})V'$省略 $\sqrt{d}$也就可以化为 $[softmax(P_1 W^Q K'^T)V'; softmax(xW^Q K'^T)V'; softmax(P_2 W^Q K'^T)V']$。最终版：

$$
\begin{aligned}
Head &= Attn(x', K', V') \\
&= \Big[\lambda_1 * \underbrace{Attn(Q_1, K_1, V_1)}_{\text{prompt tuning}} + \lambda_2 * \underbrace{Attn(Q_1, K_2, V_2)}_{\text{postfix}} + (1 - \lambda_1 - \lambda_2) * \underbrace{Attn(Q_1, K, V)}_{\text{prompt tuning}}; \\
&\beta_1 * \underbrace{Attn(Q, K_1, V_1)}_{\text{prompt tuning}} + \beta_2 * \underbrace{Attn(Q, K_2, V_2)}_{\text{postfix}} + (1 - \beta_1 - \beta_2) * \underbrace{Attn(Q, K, V)}_{\text{standard}}; \\
&\gamma_1 * \underbrace{Attn(Q_2, K_1, V_1)}_{\text{postfix}} + \gamma_2 * \underbrace{Attn(Q_2, K_2, V_2)}_{\text{postfix}} + (1 - \gamma_1 - \gamma_2) * \underbrace{Attn(Q_2, K, V)}_{\text{postfix}} \Big].
\end{aligned}
$$

Dynamic Prompting:

1. **Dynamic Position:** 用一个one-layer网络POS_θ和Gumbel-Softmax优化得到针对不同task/instance的**dpos**参数，原始的prompt可以被分为 $P = [P_{before}, P_{after}]$，其中$P_{before} = [P_1, \cdots, P_{dpos}], P_{after} = [P_{dpos+1}, \cdots, P_l]$，因此输入为$X' = [P_{before}; X; P_{after}]$。$POS_\theta$的输出为$\alpha \in R^{l+1}$(这是一个二进制的串，0到$l$一共有$l+1$个可能的位置，每个位置对应的值为0/1)，这里选用Gumbel-Softmax方式处理保证可微，$logit = Gumbel - Softmax(POS_\theta(x), \tau)$，$logit$是二进制串，只有一个位置值为1。

   adap_ins_pos：关注instance层面的position变化，需要添加$d \times (l+1)$个参数
   adap_pos：关注task层面的position变化，需要添加$l+1$个参数

2. **Dynamic Length:** 用一个one-layer网络LEN_θ和Gumbel-Softmax优化得到针对不同task/instance的l^*参数，即为：

$$P \in \mathbb{R}^{l^* \times d}, l^* = \operatorname*{argmin}_i loss(LM([\hat{P}_i; X] | \hat{P}_i \in \{\hat{P}_1, \cdots, \hat{P}_l\}, \hat{P}_i \in \mathbb{R}^{i \times d})).$$

   但实际上model的输入长度要求是固定的，因此作者作了一个替代方案

3. **Dynamic Vector:** 使用prompt pools $Pool = \{P^{(1)}, \cdots, P^{(k)}\}$生成dynamic prompts，train一个小的网络P_{O_θ}来得到每个prompt $P^{(i)}$关于给定输入x的attention score，即为$P_{new} = \sum_{i=1}^{k} \beta_i \cdot P^{(i)}, \beta = softmax(P_{O_\theta}(x))$

4. **Combination:**

   adap_ins_vec_pos：同时更新dynamic position和prompt pool
   adap_pos_ins_vec：先用dynamic position学到task层面的position，然后更新instance层面的prompt pool

3 Experiments

1. 数据集：采用五个SuperGLUE数据集来测试模型的language understanding ability
2. 实验结果：

   1. **Adaptive Position：**

Dataset	T5-LM-Small			T5-LM-Base			T5-LM-Large			T5-LM-XL		
	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position	Fixed Position	Adaptive Position	Adaptive Ins_Position
Boolq	67.31	67.55	**67.61**	62.35	**69.88**	69.17	81.20	84.60	**85.35**	89.02	88.89	**89.16**
MultiRC	68.68	68.89	**69.29**	57.42	70.19	**71.08**	58.00*	72.77	**80.20**	**84.49**	84.31	84.41
WiC	62.69	66.14	**68.34**	53.61	64.42	**64.89**	69.30	**71.20**	**71.20**	**72.57**	71.22	70.91
CB	**83.93**	**83.93**	**83.93**	78.57	**87.50**	**87.50**	87.50	89.29	**91.07**	94.64	**98.21**	96.43
RTE	65.34	**66.79**	65.70	67.51	70.75	**71.93**	82.60	**85.71**	**85.71**	88.21	**90.94**	90.58
Avg.	69.59	70.66	**70.97**	63.89	72.55	**72.91**	75.72	80.71	**82.71**	85.79	**86.72**	86.30

   可以看到总体趋势是*adap_ins_pos > adap_pos > fixed_pos*，**T5-LM-Large**模型work得最好可能说明**大模型更适合prompt tuning**

   2. **Adaptive Length：**

Dataset	T5-LM-Base		T5-LM-Large	
	Fixed Length	Adaptive Length	Fixed Length	Adaptive Ins_Length
Boolq	62.35	**67.28**	81.20	**83.46**
MultiRC	**57.41**	57.34	58.00	**66.30**
WiC	53.61	**60.50**	69.30	**71.47**
CB	78.57	**80.36**	**87.50**	84.32
RTE	67.51	**68.32**	**82.60**	79.78
Avg.	63.89	**66.76**	75.72	**77.07**

   虽然有提升，但是提升不如**Adaptive Position**

   3. **Adaptive Prompt：**

Dataset	T5-LM-Small			T5-LM-Base			T5-LM-Large		
	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine tuning	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine Tuning	Adaptive Ins_vec_pos	Adaptive Pos_ins_vec	Fine Tuning
Boolq	67.40	**68.04**	71.02	**62.51**	62.39	81.33	84.07	**84.98**	87.25
MultiRC	68.92	**69.12**	69.58	**57.96**	57.65	77.91	78.96	**82.03**	85.85
WiC	66.30	**66.61**	65.25	60.97	**64.29**	69.18	70.69	**72.57**	73.82
CB	82.14	**85.71**	92.86	**80.36**	75.00	94.62	**94.64**	**94.64**	94.64
RTE	66.42	**67.15**	68.84	61.01	**61.73**	78.62	**86.64**	**86.64**	86.59
Avg.	70.24	**71.33**	73.51	**64.56**	64.21	80.33	83.00	**84.17**	85.63

   可以看到在instance层面同时更新position和propmt pool效果并不好

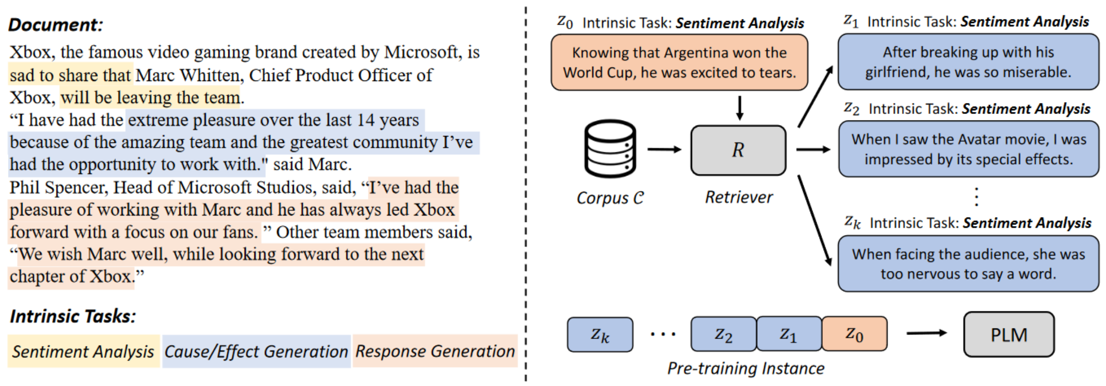
Pre-Training to Learn in Context

> Submitted on **16 May 2023**

1. Introduction

提出了**PICL(Pre-training for In-Context Learning)**，旨在提高ICL能力的同时maintain泛化能力。**PICL**主要通过在数据侧做文章来提高ICL能力：用data automatically constructed from the general plain-text corpus来预训练模型，基于很多**paragraphs**都包含**“intrinsic tasks”**这样的假设。

具体来说，把相同类型intrinsic task的paragraph连接在一起构建一个**meta-training dataset**来预训练模型。用contrastive learning的方式训一个**Encoder**使得具有相同类型intrinsic task的paragraph在向量空间中具有类似的Embedding

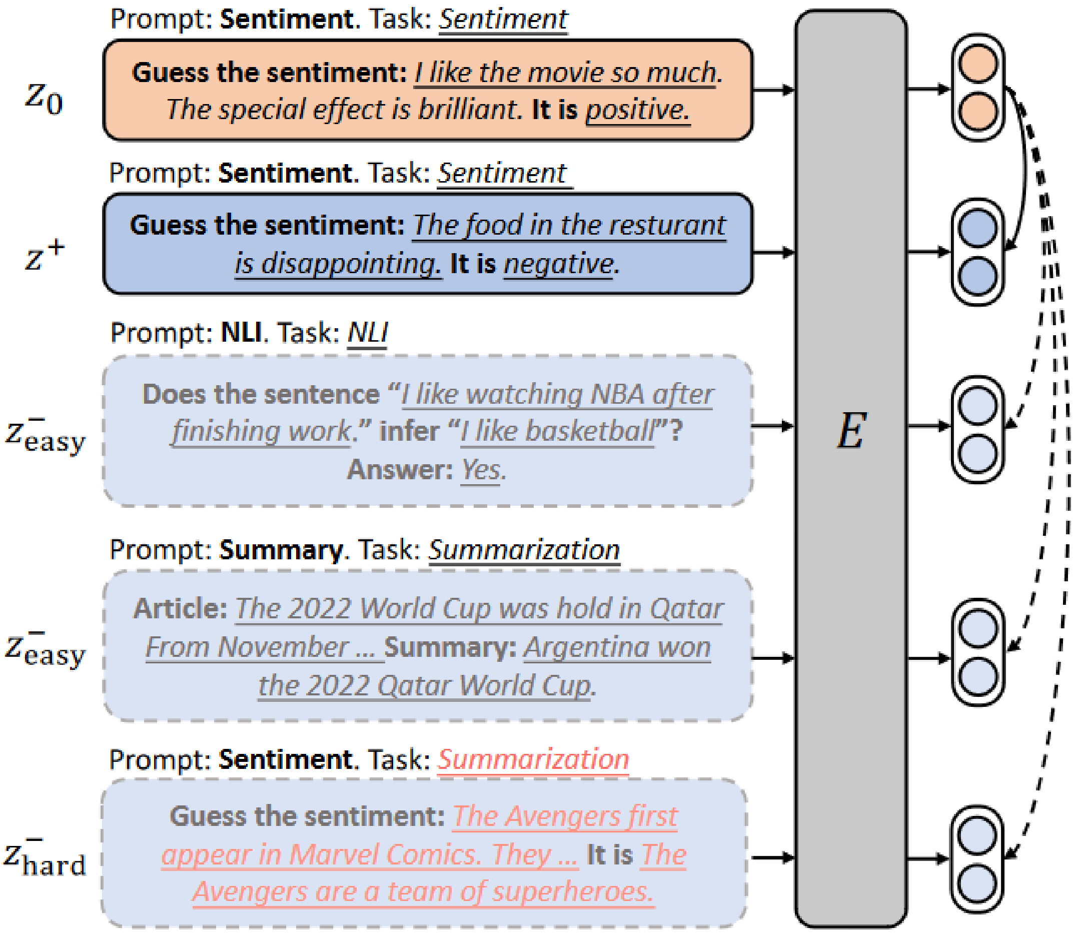


2 Methods

对于Corpus C中的每个paragraph z_0，首先用retriever R寻找k个与z_0具有相同类型intrinsic task的paragraphs $\{z_1, z_2, \cdots, z_k\}$，然后被检索出来的paragraphs会被视为demonstrations与z_0连接在一起：$z_k \oplus z_{k-1} \oplus \cdots \oplus z_1 \oplus z_0$，最后喂给模型

Retriever： 核心是**task-semantics encoder** E(**其实就是对比学习的目标**)，作者将两个paragraph z_0和z的相似性定义为点乘$E(z_0) \cdot E(z)$

1. Encoder：使用**RoBERTaBASE**作为base model，输出的vector是**输入paragraph的每个token的最后一层表示的平均**
2. Retrieval：$R(z_0) = \{z_k, z_{k-1}, \cdots, z_1\} = top - k_z(E(z_0) \cdot E(z))$，具体实现调用了**FAISS**库
3. Contrastive Learning：不同task选用下游NLP数据集，最终形成了一个dataset D。对于D中的每个z_0，正样本z^+跟z_0有相同的task类型，负样本集合为$N(z_0)$，loss可以计算为$L(z_0, z^+, N(z_0)) = -\log \frac{e^{E(z_0) \cdot E(z^+)}}{e^{E(z_0) \cdot E(z^+)} + \sum_{z^- \in N(z_0)} e^{E(z_0) \cdot E(z^-)}}$。其中$z^+$是随机sample出来的，$N(z_0)$包括两种样本：1.**Easy Negatives:** z^-_{easy} 2.**Hard Negatives:** z^+_{hard}



Data Construction：对于每个$z_0 \in C$，连接retrieved paragraphs $\{z_1, z_2, \cdots, z_k\} = R(z_0)$从而得到a pre-training instance$z_k \oplus \cdots \oplus z_0$。评价一个instance的informativeness：$s = \frac{-\sum_{i=0}^{k}\log P(z_i)+\log P(z_k\oplus z_{k-1}\oplus\cdots\oplus z_0)}{|z_k\oplus z_{k-1}\oplus\cdots\oplus z_0|}$，其中$|\cdot|$是instance的长度，$P(\cdot)$是language modeling probability

Pre-training：计算了整个sequence的loss $L_{ICL}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log P(z_k^i \oplus z_{k-1}^i \oplus \cdots \oplus z_0^i; \theta)$，再添加一个language modeling loss $L_{LM}(\theta)$。最终的优化目标为$\min_{\theta} \alpha L_{ICL}(\theta) + (1-\alpha) L_{LM}(\theta)$

3 Experiments

1. 数据集：用 OPENWEBTEXT 、 WIKICORPUS 、 BOOKCORPUS 构建pre-training data，corpus C一共包括$80M$ paragraphs，对于每个paragraph找$k = 20$ demonstrations并连接它们到$1024 tokens$

2. Baseline：

   VanillaICL ： 用连接的training examples直接prompt PLM
   ExtraLM ： 在原始的full document被分成paragraph之前进一步pre-train PLM
   Self-Sup ： 设计了四个自监督的预训练目标 **Next Sentence Generation, Masked Word Prediction, Last Phrase Prediction, and Classification**
   MetaICL ： 用人工标注的下游数据集meta-train模型

3. 评估：

Few-Shot Text Classification：

`ExtraLM` 有效，说明corpus的diversity很大； `metaICL` 有效，说明meta-training对ICL的提升有作用； `Self-Sup` 无效，说明训练时分类task受限的label space给模型输出带来了bias

Shot	Method	Param.	SST2	SUBJ	MR	RTE	AgNews	CB	SST5	Average
4-shot	VanillaICL	770M	67.5$_{9.2}$	57.7$_{7.8}$	50.3$_{0.3}$	50.8$_{1.7}$	67.5$_{2.3}$	68.1$_{2.4}$	24.4$_{5.4}$	55.2$_{0.5}$
	VanillaICL	1.5B	74.9$_{9.7}$	65.2$_{10.0}$	61.9$_{6.5}$	50.4$_{0.4}$	65.6$_{4.8}$	67.8$_{5.6}$	32.4$_{4.6}$	59.7$_{2.5}$
	VanillaICL	2.7B	75.0$_{7.5}$	65.4$_{2.9}$	71.4$_{13.3}$	49.8$_{1.8}$	65.6$_{2.8}$	60.0$_{2.1}$	32.1$_{5.4}$	59.9$_{1.1}$
	ExtraLM	770M	68.9$_{11.3}$	63.9$_{6.4}$	60.3$_{6.4}$	51.2$_{1.7}$	64.5$_{1.5}$	63.7$_{5.3}$	27.8$_{5.1}$	57.2$_{2.1}$
	Self-Sup	770M	55.0$_{7.4}$	50.3$_{0.6}$	59.7$_{3.5}$	52.2$_{2.0}$	50.3$_{7.0}$	63.4$_{7.1}$	28.8$_{3.3}$	51.4$_{2.2}$
	MetaICL	770M	69.8$_{4.0}$	63.5$_{4.6}$	65.6$_{7.5}$	**57.6**$_{2.3}$	66.3$_{2.4}$	65.2$_{3.0}$	31.7$_{2.1}$	60.0$_{1.5}$
	PICL	770M	**79.7**$_{8.6}$	**66.8**$_{7.4}$	**81.0**$_{1.3}$	54.5$_{1.8}$	**67.7**$_{3.4}$	**69.6**$_{4.3}$	**34.8**$_{4.0}$	**64.4**$_{1.6}$
8-shot	VanillaICL	770M	68.7$_{6.0}$	66.6$_{9.8}$	60.2$_{5.5}$	51.8$_{1.6}$	60.2$_{5.6}$	68.8$_{3.2}$	31.4$_{3.8}$	58.2$_{2.9}$
	VanillaICL	1.5B	72.1$_{12.6}$	63.4$_{6.5}$	63.3$_{5.4}$	52.7$_{2.8}$	54.2$_{8.4}$	**70.4**$_{5.7}$	33.5$_{3.3}$	58.6$_{2.5}$
	VanillaICL	2.7B	71.0$_{11.6}$	65.2$_{4.0}$	70.4$_{6.3}$	51.3$_{2.0}$	63.1$_{2.4}$	69.6$_{4.0}$	**34.1**$_{2.8}$	60.6$_{3.2}$
	ExtraLM	770M	69.7$_{3.4}$	65.2$_{6.5}$	63.6$_{6.0}$	52.6$_{1.6}$	58.9$_{7.0}$	69.6$_{3.8}$	32.2$_{4.7}$	58.8$_{1.6}$
	Self-Sup	770M	61.4$_{6.5}$	54.3$_{4.5}$	73.8$_{8.1}$	53.0$_{2.4}$	52.1$_{3.8}$	63.0$_{6.9}$	33.7$_{1.8}$	55.9$_{2.1}$
	MetaICL	770M	73.6$_{6.2}$	67.2$_{8.8}$	70.1$_{5.6}$	**53.6**$_{2.1}$	56.1$_{0.7}$	65.8$_{4.1}$	33.7$_{4.7}$	60.0$_{2.2}$
	PICL	770M	**78.0**$_{10.6}$	**69.3**$_{9.5}$	**77.5**$_{5.0}$	53.0$_{1.6}$	**64.7**$_{4.4}$	**70.4**$_{2.1}$	**34.1**$_{3.8}$	**63.9**$_{1.3}$

Instruction Following：测试模型的泛化性

Model	Param.	ROUGE-L
VanillaICL	770M	34.3
VanillaICL	1.5B	34.9
VanillaICL	2.7B	37.3
ExtraLM	770M	34.6
Self-Sup	770M	30.5
MetaICL	770M	35.3
PICL	770M	**37.6**

`PICL` 比 `MetaICL` 在更多的task上表现得更好说明了**与直接在下游任务上微调相比，在intrinsic task上预训练更能提升ICL能力、泛化能力更强**。 `PICL` 在text generation等任务上表现更好而 `MeatICL` 在**“Yes/No”**问题上表现更好，说明在下游数据集训练会导致对某些label过拟合

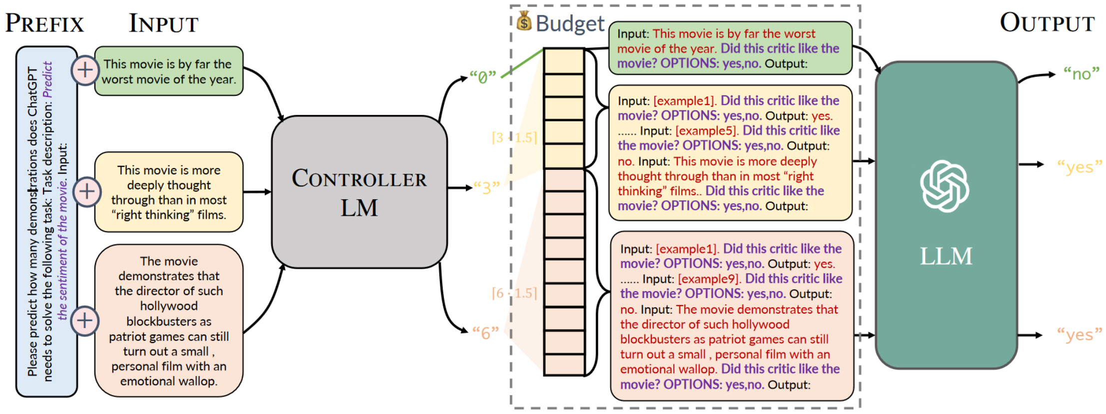
> 文章后面探究了**Retriever**、**Demonstration Number**、**Filtering**、**Full Documents**、**Data Amount**、**Data Comparison**的影响

Efficient Prompting via Dynamic In-Context Learning

> Submitted on **18 May 2023**

1 Introduction

提出了**DYNAICL(Dynamic In-Context Learning)**，为有效解决performance-efficiency trade-off问题提供了一种方案。model size和sample size是影响计算低效的两个原因，后者可以通过减少prompt长度实现，而prompt长度受到in-context learning使用demonstration数目的影响。因此该论文的核心是train一个meta controller来分配in-context demonstration的数目



2 Methodology

Meta Controller： 采用instruction-tuned model **FLAN-T5**作为base model，主要关注分类任务。训练分为**两阶段**

第一阶段目标是train一个meta controller，可以使得**generalist model, like Chatgpt**生成**"good output"**的同时利用最少的in-context examples，输出所需的example数目k，即为下图(其中t是一个threshold)。

$$\boxed{k^* = \min_{k \in \mathbb{N}} \left\{ k \,\middle|\, \mathbb{E}_{(x_{i_1}, y_{i_1}) \ldots (x_{i_k}, y_{i_k})) \sim \mathcal{D}^k} \left[\operatorname{Acc}(\mathcal{G}(P, \mathcal{T}(x_1, y_1) \oplus \cdots \oplus \mathcal{T}(x_k, y_k))) \right] > t \right\}}$$

where D^k denotes all subsets of the training data of size k.

第二阶段是利用**强化学习**来微调meta controller

Dynamic In-Context Example Allocation： 考虑到实际的computation budget，假设总共有K samples N tokens 每个example的平均长度为L，平均分配的baseline为$\frac{N}{K \times L}$。**DYNAICL**的分配策略是$E(P) = [\beta \cdot (C(P)/\widetilde{C}) \cdot N/(K \cdot L)]$，其中$C(P)$是meta controller的预测结果，$[\cdot]$代表取整操作，$\widetilde{C}$是所有examples的平均预测结果，$\beta$是token saving ration

3 Experiments

1. 数据集：选用 `a subset in the FLAN collection containing 30+ classification tasks` 来训练meta controller，大部分数据集都是分类任务，用作训练；少部分数据集不是分类任务，用作评估。同时一些分类任务对应的数据集会作为unseen task来

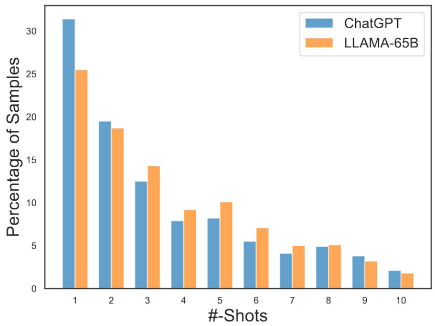
评估

2. 模型：选用 ChatGPT 作为generalist model， LLAMA-65B 作为unseen generalist model评估meta controller的泛化性

3. Baseline：

uniform baseline： 每个sample分配相同数目的in-context examples
random baseline： 遵循高斯分布随机选取一定数量的in-context examples

4. Preliminary：

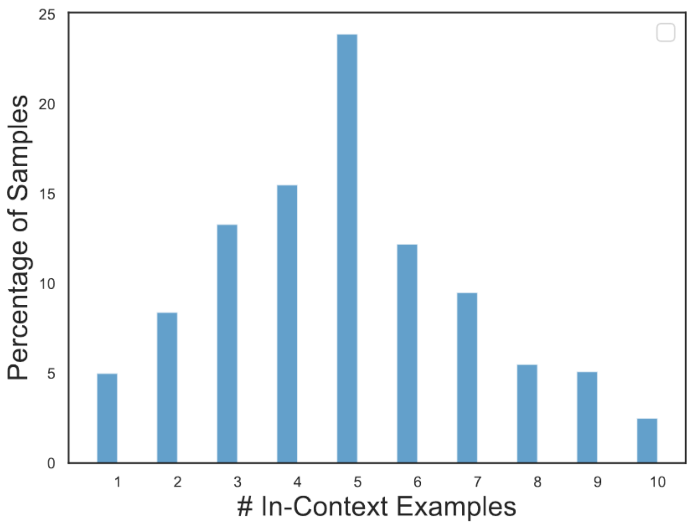


Δ Accuracy	✗→✓	✓→✗
zero-shot → 1-shot		
+ 2.5%	3.9%	1.4%
1-shot → 5-shots		
+ 1.4%	1.9%	0.5%
5-shots → 64-shots		
+ 0.3%	0.7%	0.4%

根据上图可以看出大部分可能只需要很少的shots就可以work，且越多的shots效果提升并不明显

5. 结果：

   1. 相同performance节省token，相同token在performance上表现更好
   2. 对于Unseen Generalist Model、Unseen Task都有很好的泛化性
   3. in-context examples的分布(target budget设为5)：



(b) Distribution of samples (on seen tasks) according to the number of in-context examples allocated for them. The computational budget is fixed to 5 in-context examples per sample.