

LLM as Prompt Optimizers

Paper: Large Language Models as Optimizers

这篇文章提出OPO，也就是用LLM作为优化器来生成使用task accuracy最高的hard prompt(自然语言形式)。整个工作是通过调用model api实现的，适用于闭源大模型，流程图如下图所示：

一个LLM负责优化，称为optimizer LLM；另一个LLM负责评估，称为scorer LLM。optimizer LLM的输出称为an instruction，用于连接到每个exemplar的question部分然后prompt scorer LLM，可以插入的位置是Q_begin, Q_end, A_begin,

最后文章探讨了Meta-prompt design(previous instructions的顺序, instruction score的表示方式, exemplar的数量), generated instructions的数量, starting point, 每步的diversity(LLM temperature设置)等对实际效果的影响。

Future work:

1. exemplar的选取要充分利用ICL
2. weaker starting point的收敛速度优化

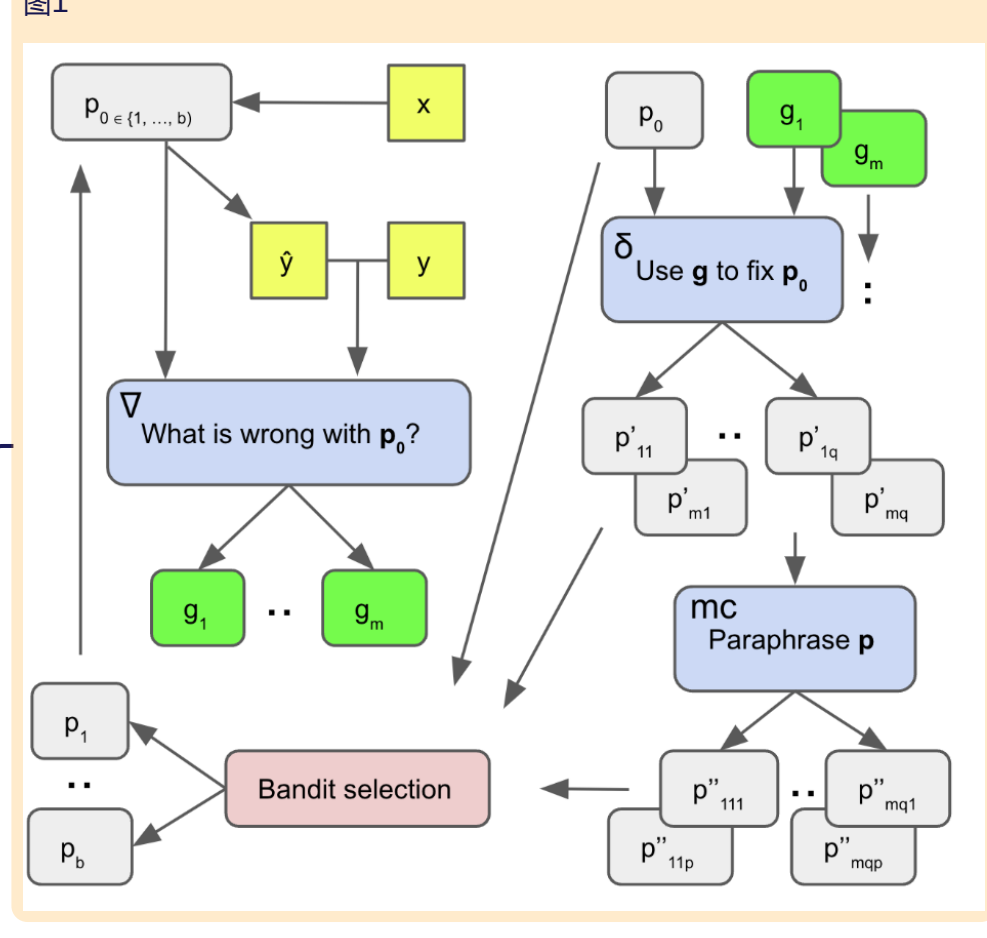
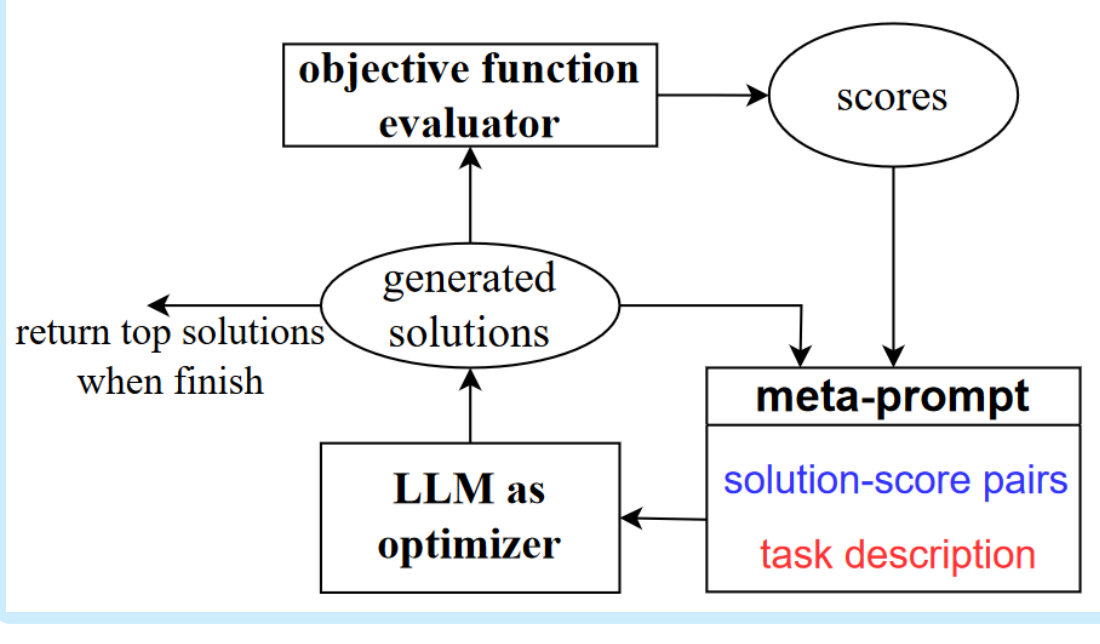


图3

Algorithm 2 *Expand()*: line 5 of Algorithm 1

Require: p : prompt candidate; D_{train} : train data

- 1: Sample minibatch $D_{\text{sample}} \subset D_{\text{train}}$
- 2: Evaluate prompt p on minibatch D_{sample} and collect errors $e = \{(x_i, y_i) : (x_i, y_i) \in D_{\text{sample}} \wedge LLM(p, x_i) \neq y_i\}$
- 3: Get gradients: $\{g_1, \dots, g_n\} = LLM_V(p, e)$
- 4: Use the gradients to edit the current prompt: $\{p'_1, \dots, p'_n\} = LLM_{\text{edit}}(p, g)$
- 5: Get more monte-carlo successors: $\{p''_1, \dots, p''_n\} = LLM_{\text{mc}}(p'_1) \cup \dots \cup \{p''_{111}, \dots, p''_{n11}\}$
- 6: **return** $\{p_1, \dots, p_n\} \cup \{p'_{11}, \dots, p'_{n1}\}$

图4

Algorithm 3 *Select()* with UCB Bandits - line 7 of Algorithm 1

Require: n : prompts p_1, \dots, p_n ; dataset D_{dev} ; T : time steps; metric function m

- 1: Initialize: $N_i(p_i) \leftarrow 0$ for all $i = 1, \dots, n$
- 2: Initialize: $Q_i(p_i) \leftarrow 0$ for all $i = 1, \dots, n$
- 3: **for** $t = 1, \dots, T$ **do**
- 4: Sample uniformly $D_{\text{sample}} \subset D_{\text{dev}}$
- 5: $p_t \leftarrow \arg \max_{p_i \in D_{\text{sample}}} \{Q_i(p_i) + c \sqrt{\frac{\ln t}{N_i(p_i)}}\}$ (UCB)
- 6: Observe reward $r_{t,i} = m(p_i, D_{\text{sample}})$
- 7: $N_t(p_i) \leftarrow N_t(p_i) + 1$
- 8: $Q_t(p_i) \leftarrow Q_t(p_i) + \frac{r_{t,i}}{N_t(p_i)}$ (UCB)
- 9: **end for**
- 10: **return** $SelectTop_n(Q_T)$

图5

Algorithm 4 *Select()* with Successive Rejects - line 7 of Algorithm 1

Require: n : prompts p_1, \dots, p_n ; dataset D_{dev} ; metric function m

- 1: Initialize: $S_0 \leftarrow \{p_1, \dots, p_n\}$
- 2: **for** $k = 1, \dots, n - 1$ **do**
- 3: Sample $D_{\text{sample}} \subset D_{\text{dev}}$; $D_{\text{sample}}(k) \leftarrow S_k$
- 4: Evaluate $p_i \in S_{k-1}$ with $m(p_i, D_{\text{sample}})$
- 5: $S_k \leftarrow S_{k-1}$, excluding the prompt with the lowest score from the previous step
- 6: **end for**
- 7: **return** Best prompt $p^* \in S_{n-1}$

Paper: Automatic Prompt Optimization with "Gradient Descent" and Beam Search

这篇paper提出了APO算法(Prompt Optimization with Textual Gradients), 主要思想就是用LM feedback代替differentiation, 用LM editing代替backpropagation, 以此来类比传统的gradient descent, 整体过程如右图1所示。

算法过程如右图2所示。每次迭代都会产生新的candidate prompts(expansion), 然后进行selection, expansion算法过程如右图3所示, selection包括UCB Bandits, Successive Rejects两种方法。分别如右图4、5所示。

图2

Algorithm 1 Prompt Optimization with Textual Gradients (ProTeGi)

Require: p_0 : initial prompt, zb : beam width, r : search depth, m : metric function

- 1: $B_0 \leftarrow \{p_0\}$
- 2: **for** $i \leftarrow 1$ **to** $r - 1$ **do**
- 3: $C \leftarrow \emptyset$
- 4: **for all** $p \in B_i$ **do**
- 5: $C \leftarrow C \cup \text{Expand}(p)$
- 6: **end for**
- 7: $B_{i+1} \leftarrow \text{Select}_b(C, m)$
- 8: **end for**
- 9: $\hat{p} \leftarrow \arg \max_{p \in B_r} m(s)$
- 10: **return** \hat{p}

Paper: Large Language Models Are Human-Level Prompt Engineers

这篇paper提出APE(Automatic Prompt Engineering)和Iterative APE, 整个工作流程如右图1所示。算法过程如右图2所示。

LLM承担的角色是proposal(生成a set of instruction candidates), scoring(为生成的instruction打分), 对于proposal中还提出了forward Mode Generation和reverse Mode Generation两种形式。对于scoring文中提出了execution accuracy, log probability两种形式。

最后这篇paper还提出了Iterative Monte Carlo Search(Iterative APE), 在当前最优的instruction candidate附近保持语义进行search(而不是在原有的proposal中search), 一个template如右图3。

实验部分主要是在不同的数据集上测试了zero-shot Learning, few-shot in-context Learning, zero-shot chain-of-thought reasoning, truthfulness。

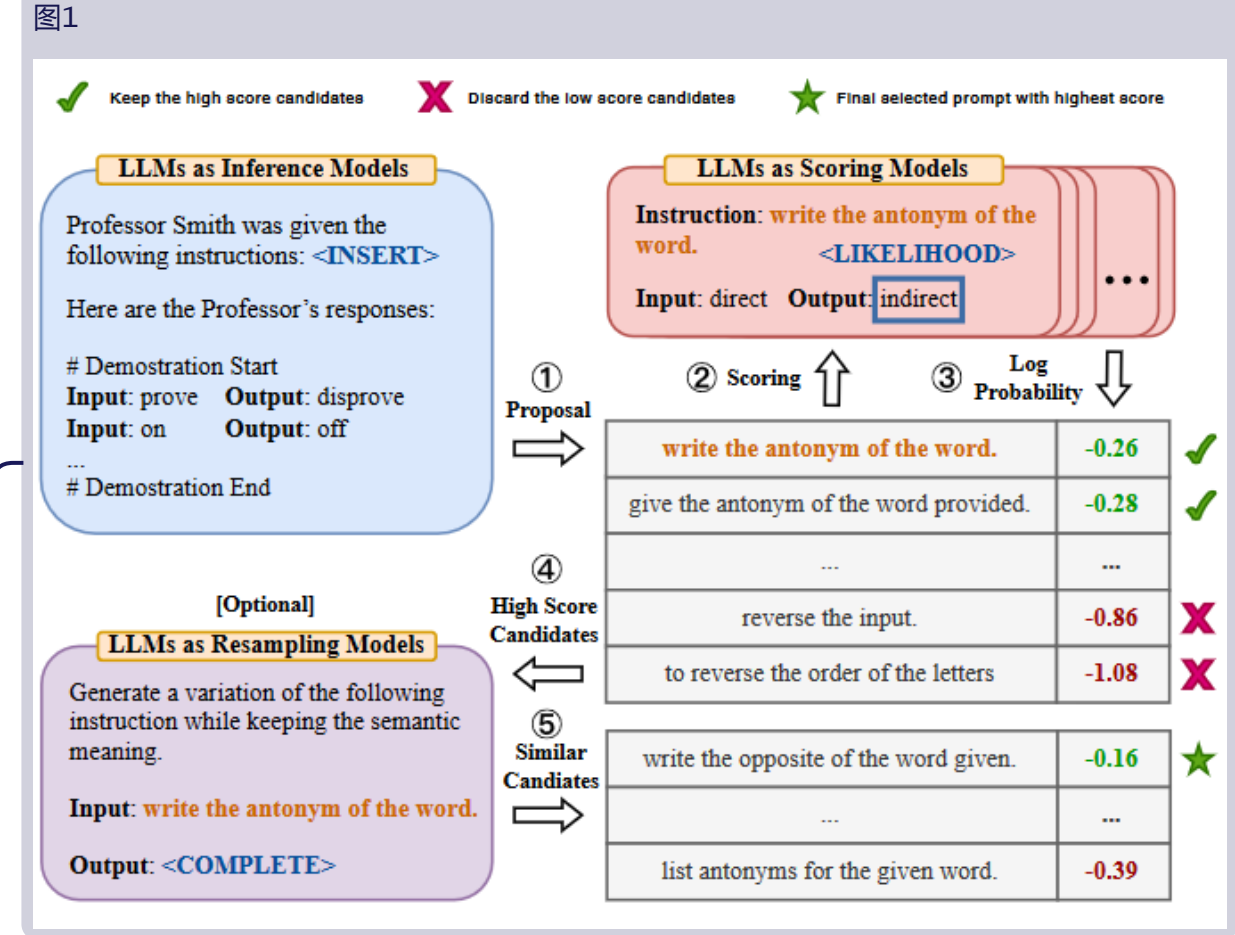


图2

Algorithm 1 Automatic Prompt Engineer (APE)

Require: $D_{\text{train}} \leftarrow \{(Q, A)\}_n$; training examples; $f: p \times D \rightarrow \mathbb{R}$: score function

- 1: Use LLM to sample instruction proposals $U \leftarrow \{p_1, \dots, p_m\}$. (See Section 3.1)
- 2: **while** not converged **do**
- 3: Choose a random training subset $\tilde{D}_{\text{train}} \subset D_{\text{train}}$.
- 4: **for all** p in U **do**
- 5: Evaluate score on the subset $\tilde{s} \leftarrow f(p, \tilde{D}_{\text{train}})$ (See Section 3.2)
- 6: **end for**
- 7: Filter the top k of instructions with highest scores $U_k \subset U$ using $\{\tilde{s}_1, \dots, \tilde{s}_k\}$
- 8: Update instructions $U \leftarrow U_k$ or use LLM to resample $U \leftarrow \text{resample}(U_k)$ (See Section 3.3)
- 9: **end while**
- Return** instruction with the highest score $p^* \leftarrow \arg \max_{p \in U_k} f(p, D_{\text{train}})$

图3

Prompt for Resampling

Generate a variation of the following instruction while keeping the semantic meaning.

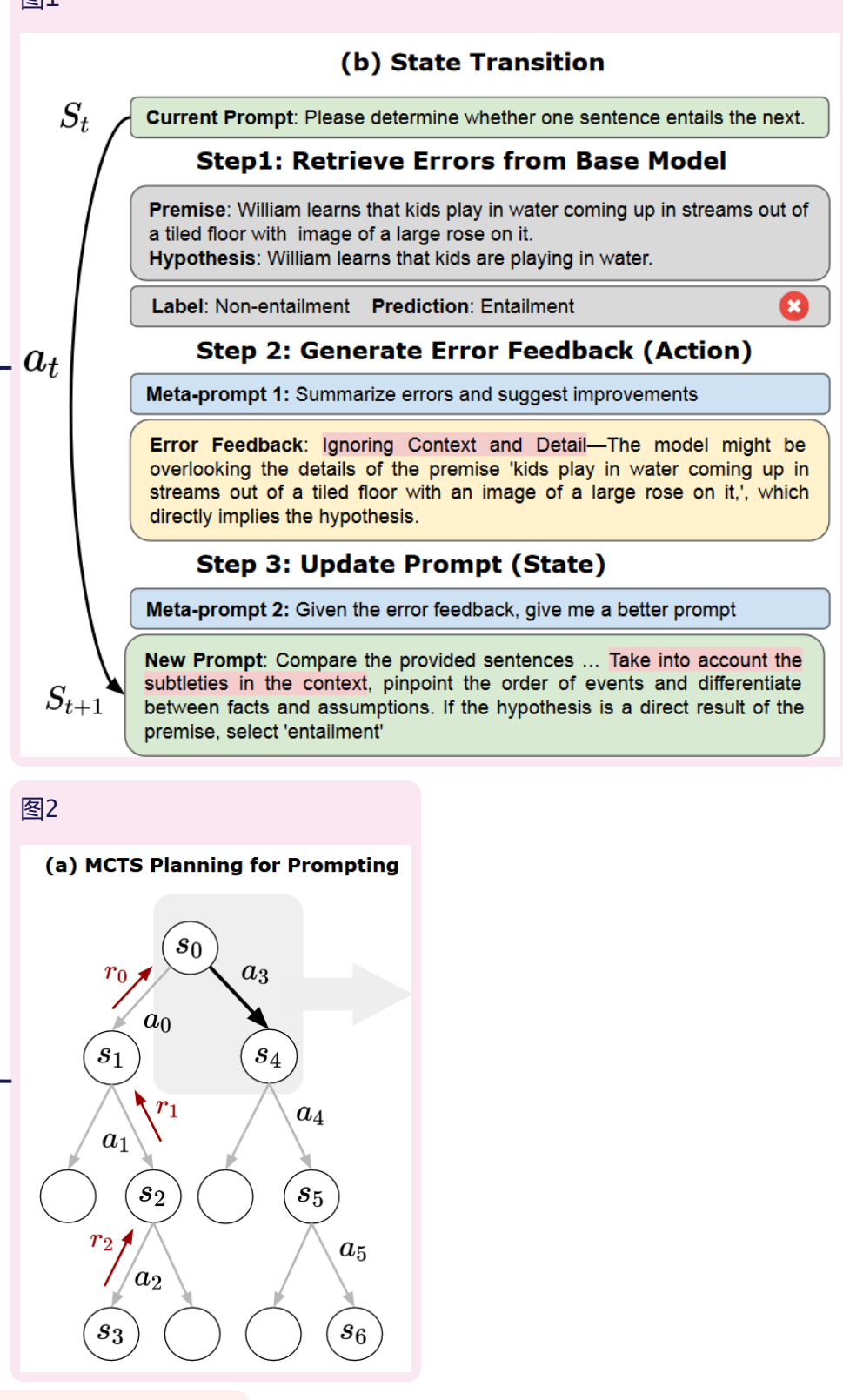
Input: [INSTRUCTION]

Output: <COMPLETE>

Paper: PromptAgent: Strategic Planning with Language Models Enables Expert-level Prompt Optimization

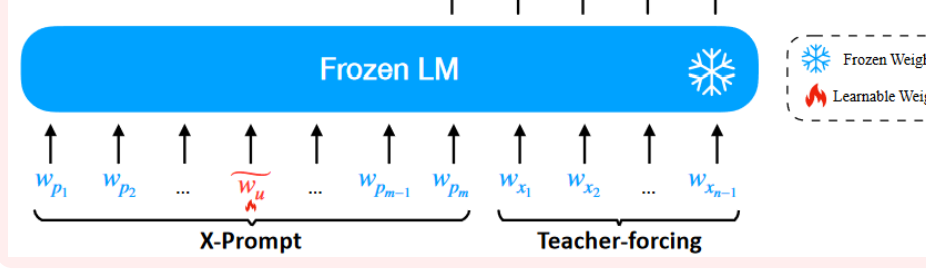
这篇paper主要提出了PromptAgent这一优化prompt的方法。主要是利用了self-reflection和strategic planning。这篇paper是把prompt optimization问题定义成MDP(马尔科夫决策过程)。其中state是每次迭代的prompt, action是对当前prompt可能的修改, 状态转移过程如右图1所示。类似RL, 这里reward的定义就是task performance

创新点主要是引入了strategic planning, 这里是MCTS(Monte Carlo Tree Search)。图1为右图2。MCTS主要通过state-action-reward保持语义进行search, MCTS迭代执行selection, expansion, simulation, back-propagation四个动作expand the tree



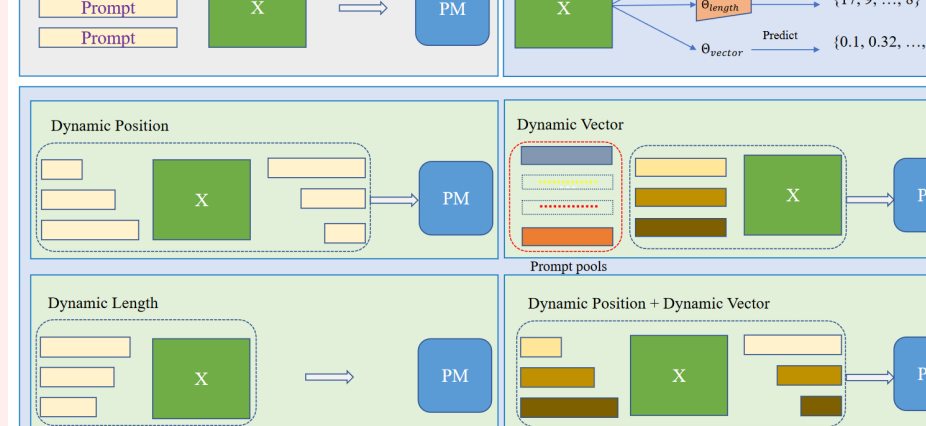
Paper: Extensible Prompts for Language Models on Zero-shot Language Style Customization

X-Prompt主要的思想是将imaginary words注入到natural language中构成prompt, 其中含有imaginary words是learnable, 文章还提出了context-Augmented Learning(CAL)的概念帮助更好地学习imaginary words(增强在OOD上的表现)。采用的task类型是open-ended text generation和style transfer



Paper: Dynamic Prompting: A Unified Framework for Prompt Tuning

Dynamic Prompting主要思想是根据不同的task/instance的情况来调整相对prompt的position, length, representation.



Paper: When do Prompting and Prefix-tuning work? A Theory of Capabilities and Limitations(粗读)

这篇paper比较理论, 公式也比较多, 所以没怎么看, 这里只记下结论。

1. content-based fine-tuning只能激发LLM原有的skill而不能学一个全新的task
2. a hierarchy: prompting < soft prompting < prefix-tuning, prefix-tuning的效果最好, prompting和in-context learning是prefix-tuning的特例
3. 对于transformer而言, 变化一个virtual token比变化一个hard token, 会产生更多的completion
4. prefix-tuning only adds a bias to the attention block output, 这个bias能激发LLM原有的skill
5. 参数相同的情况下, LoRA可以学会全新的task, prefix-tuning不可以

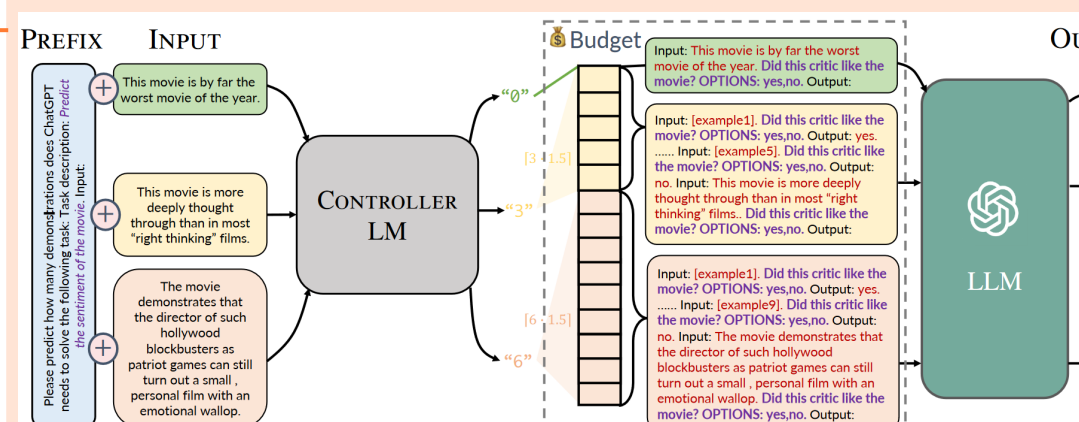
Paper: How Does In-Context Learning Help Prompt Tuning

这篇paper主要是对比了PT(Prompt Tuning), ICL(In-Context Learning), IPT(Instruction Prompt Tuning)三种方法的效果来探究ICL对prompt tuning的影响。task类型选用的是language generation task(data-to-text generation, logic-to-text generation, semantic parsing)

得到的一些结论: 1. ICL表现要比PT差 2. PT与IPT表现难分伯仲(相关task类型/tunable parameter数目等) 3. 当demonstration层test input类似时, IPT可以很好工作(别的论文有结论: in-context examples主要是帮助model学习output label space和distribution of input text) 4. IPT在有更多soft prompt tokens时表现要比PT更稳定 5. 在有in-context demonstration的情况下, prompt embeddings对于新的task是transferable

Paper: Efficient Prompting via Dynamic In-Context Learning

DYNAICL(Dynamic In-Context Learning)的提出是为了有效解决performance-efficiency trade-off问题。因为in-context learning使用的demonstration数目会影响prompt长度, 而prompt长度过大导致低效, 因此动态分配demonstration的数目可以有效解决此问题。论文的核心是一个meta controller可以动态分配in-context demonstration的数目, train分为两个阶段: 1. 使得最后用generalist model生成'good output'的同时利用最少的demonstrations 2. 利用RL来微调



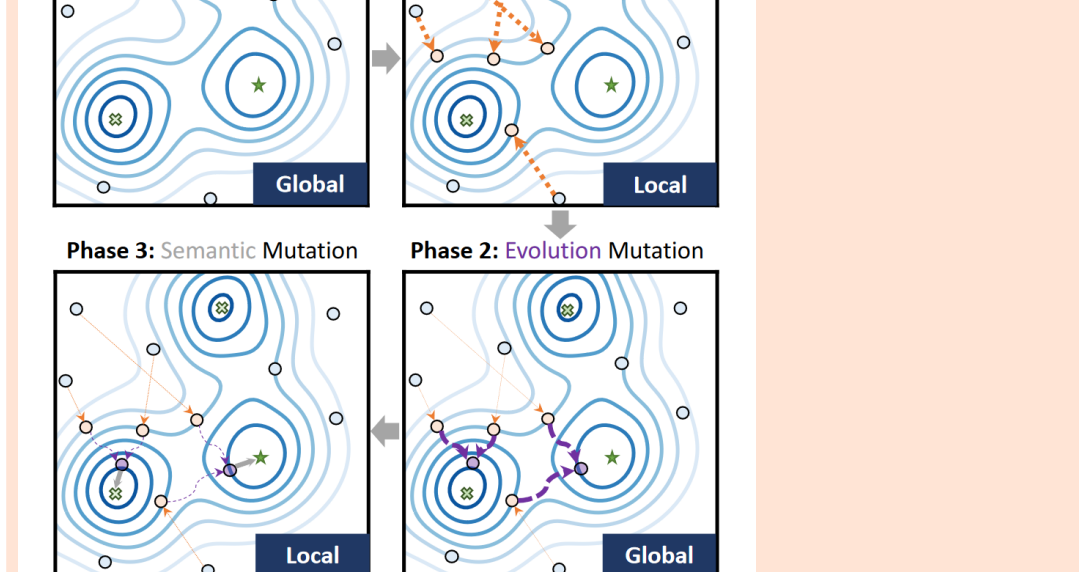
Paper: PhaseEvo: Towards unified In-Context Prompt Optimization for Large Language Models(粗读)

Submitted on 17 Feb 2024

这篇paper主要提出了一个统一的in-context prompt优化的框架PhaseEvo, 核心算法是进化算法(Evolutionary Algorithms), 采用了Exploration, Exploitation两种不同的优化策略。整个框架分为四个阶段:

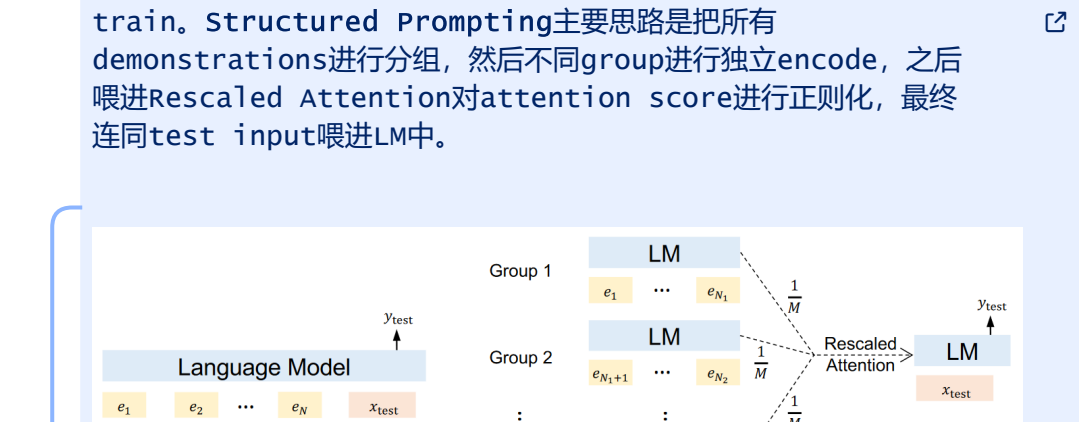
1. global initialization
2. local feedback mutation
3. global evolution mutation
4. local semantic mutation

Local通常是为了加速收敛/逼近局部最优, global是为了防止stuck in局部最优



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

这篇paper主要目的是解决传统Prompting增加demonstrations数目难度比较大的问题, 实现了使用较多数目的demonstrations进行train, Structured Prompting主要思想是把所有demonstrations进行分组, 然后不同group进行独立encode, 之后喂进rescaled Attention score进行正则化, 最终连同test input喂进LLM中。



Paper: Pre-Training to Learn in-Context Learning(PICL), 基于Corpus中很多paragraphs都包含intrinsic tasks的假设, 用Retriever将具有相同类型intrinsic tasks的paragraph检索出来, 并连接在一起构建一个meta-training dataset, 用contrastive learning的方法train一个encoder使得具有相同类型intrinsic tasks的paragraph具有类似的embedding

Paper: Active Example Selection for In-Context Learning(粗读)

这篇paper主要提出了为In-Context Learning挑选example的方法。方法背后的背景是Reorder/Calibration都不能很好解决instability/variance的问题。基于此, 作者将example selection视为sequential decision问题。整个过程类似于active learning, 作者将active example selection操作MOP, 然后采用Q-learning算法解决