idea

Task：用in-context learning来优化prompt

首先这里的prompt肯定是含有soft prompt的，soft prompt是learnable的而hard prompt是不可学习的
利用in-context learning来优化，就是利用demonstration，目前了解到demonstration的数目对效果有影响
Influence Function对data的选取?

一个可能的setting是要用LLM自己迭代优化prompt???
这里就是优化hard prompt了，可以让LLM作为Optimizers进行优化

关于LLM as Optimizers的思考

现有的方法

1. `Resampling-based methods`(选择语义类似的prompt)

   Iterative-APE/APE

2. `Reflection-based methods`(可能发生语义改变的prompt)

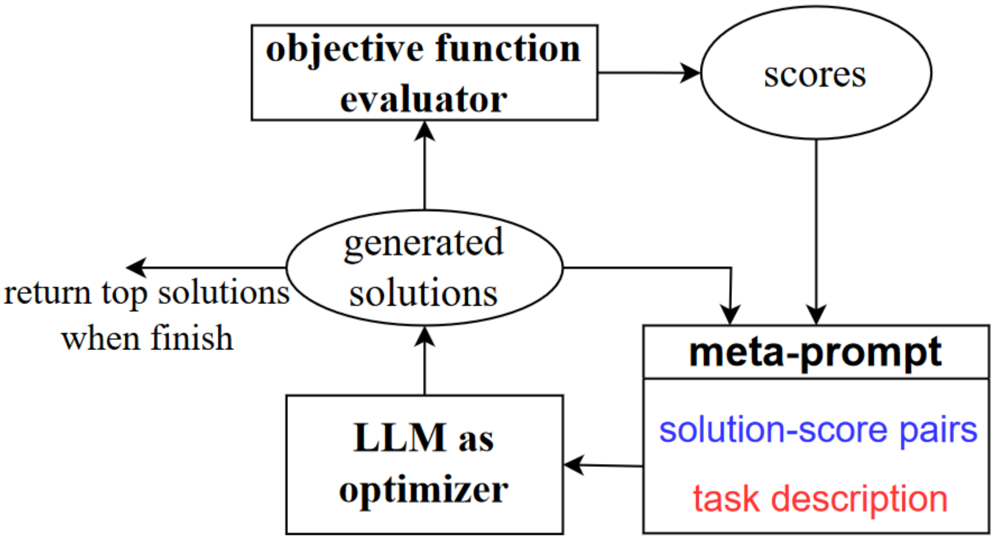
   APO
   PromptAgent
   OPRO(implicit reflection-based method): 在有些task上表现相对较差，一个是因为优化方法没有方向导致LLM不知道什么是"better prompts"，另一个可能是因为LLM并不一定在最优的prompt附近sample

> 实现上的对比:
> APE / Iterative-APE:
>
> 1. 首先根据给定的一些demonstrations(**这些demonstrations是随机的?**)，LLM生成prompts"池"
> 2. 然后根据特定的score function给prompts打分，并根据分数进行filter/refine
> 3. 更新prompts"池"(APE: 直接根据分数更新 Iterative-APE: 让LLM生成语义类似的prompts进行更新)
>
> APO: 第一个prompt ∇ 根据初始的prompt p_0 产生一系列的梯度 g; 第二个prompt δ 利用梯度 g 来edit当前的prompt p_0
>
> 1. 初始的prompt为 p_0
> 2. 通过一个Expand算法，根据当前的prompt，先生成梯度 g，然后根据梯度 g 来修改当前的prompt，最后运用一个**paraphrasing LLM**($called \quad LLM_{mc}$)在

新的prompt candidates周围explore local monte carlo search space(**要求LLM生成跟输入语义相似但是用词不同的新candidates**)

3 通过算法**UCB Bandits**或者算法**Successive Rejects**来挑选prompts

OPRO:



该方法是按照optimization trajectory生成新的prompts，同时对prompt插入的位置也做了探索

- **现有的问题**

1. LLM在self-correction上表现得并不是太好，体现在LLM产生的reflection重复/类似(无论error distribution是什么样)，因此可能导致LLM产生的prompt并不适用
2. LLM optimizer产生的新prompt并不一定会被target model follow，因为target model的instruction-following的能力是不可控制的
3. LLM对prompt format比较sensitive，语义相似的prompts可能表现得截然不同
4. 如何利用更少的数据(demonstration)/tokens让LLM把task学好，self-reflection也是同样的道理：更多的demonstrations/reflections在一定程度上会让prompt效果变好，但是也意味着api开销比较大

- **可能的方案**

1. `Automatic Behavior Optimization` (ABO):

1. At each step, the LLM optimizer is instructed to generate step-by-step prompts.

2. Next, we utilize the LLM optimizer to write an "Instruction-following Demonstration" for each prompt, i.e., an example illustrating how to strictly follow every detail of the given prompt. This practice aims to enhance the controllability of the prompt optimization process by ensuring any improvement made will be strictly followed by the target models.

3. During the reflection and prompt refinement process, given error examples, the LLM optimizer is required to identify the failure step of the target model and refine the prompt by further breaking down the solution at the problematic step. This aims to avoid invalid feedback by utilizing the LLM optimizer to perform more objective tasks during reflection.

4. For each refined prompt, the instruction-following demonstration is also updated to illustrate how to strictly follow the refined steps.

2. 要用ICL进行优化，可以考虑demonstrations的选取/表示形式。比如判断对错的task，提供错误的demonstration会不会比提供正确的demonstration效果更好？context提供的内容脑洞是不是也可以开大点？

Prompt Tuning vs Prompt Engineering

来自一个 Youtube视频

Prompt Engineering是Hard prompt，通常是人工构建的

> 关于Prompt Engineering的一个blog: Prompt Engineering
> Some insights:
>
> 1. Many studies looked into how to construct in-context examples to maximize the performance and observed that choice of **prompt format, training examples, and the order of the examples** can lead to dramatically different performance, from near random guess to near SoTA
> 2. Tips for Example Selection: Choose examples that are semantically similar to the test example using k-NN clustering in the embedding space、Use a directed graph to select a diverse and representative set of examples、Use Contrastive Learning to train embeddings、Use Q-learning to do sample selection、Motivated by uncertainty-based active learning, Diao suggested to identify examples with high disagreement or entropy among multiple sampling trials
> 3. Tips for Example Ordering: **A general suggestion is to keep the selection of examples diverse, relevant to the test sample and in random order to avoid majority label bias and recency bias**
> 4. Instruction Prompting: **When interacting with instruction models, we should describe the task requirement in details, trying to be**

specific and precise and avoiding say "not do something" but rather specify what to do、Explaining the desired audience is another smart way to give instructions、In-Context Instruction Learning

5. The benefit of **CoT** is more pronounced for complicated reasoning tasks, while using large models (e.g. with more than 50B parameters). Simple tasks only benefit slightly from CoT prompting
6. Augmented Language Models: Augmented Language Models: a Survey has great coverage over multiple categories of language models augmented with reasoning skills and the ability of using external tools. We can use **external/internal retrieval** to get knowledge about a topic before answering the question
7. External APIs: Toolformer: Language Models Can Teach Themselves to Use Tools, toolformer-pytorch

Prompt Tuning是Soft prompt，是AI生成的embedding，可解释性差

Hard prompt vs Soft prompt

Hard prompt是Discrete Prompt，prompt是一个实际的文本字符串

> Hard prompts are manually handcrafted text prompts with discrete input tokens. ~ HuggingFace

Soft Prompt是Continuous prompt，直接在底层语言模型的embedding中进行描述

> Soft prompts are learnable tensors concatenated with the input embeddings that can be optimized to a dataset; the downside is that they aren't human readable because you aren't matching these "virtual tokens" to the embeddings of a real word. ~ HuggingFace

> "soft" prompts designed by an AI that outperformed human-engineered "hard" prompts. ~ Source

Prefix Tuning vs Prompt Tuning vs P-tuning

1. Prefix Tuning与Prompt Tuning的区别：The prefix parameters are inserted in **all** of the model layers, whereas prompt tuning only adds the prompt parameters to the model input embeddings. The prefix parameters are also optimized by a separate feed-forward network (FFN) instead of training directly on the soft prompts because it causes instability and hurts performance

2. P-tuning(与Prefix Tuning的区别)：The prompt tokens can be inserted anywhere in the input sequence, and it isn't restricted to only the beginning. The prompt tokens are only added to the input instead of adding them to every layer of the model. Introducing *anchor* tokens can improve performance because they indicate characteristics of a component in the input sequence

Completion/Token Concept

The inputs are called *prompts* and outputs are referred to as *completions*.

LLMs take the input *prompts* and chunk them into smaller units called *tokens* to process and generate language. Tokens may include trailing spaces and even sub-words. This process is language dependent.

Prompt/Instruction Concept

prompt概念的运用要早于instruct。对于生成式模型来讲prompt一直都指模型的启动输入，也就是整体叫prompt，instruct是prompt概念在instructGPT出现后的的一种新的组织形式。

我这里给出一个具体的例子，下面是给模型的一个prompt，也就是说下面整体是一个prompt。

我们目前

{

"instruction": "请解释道路交通事故中的民事责任是什么意思，并阐述其适用条件。",

"input": "",

"output": "民事责任是指在道路交通事故中，依据《侵权责任法》和其他相关法律规定，当事人因过错或者其他违法行为给他人造成损失，应当承担对方的损失赔偿等民事责任。民事责任的适用条件包括：1.有过错：造成损害的行为必须是具有过错的行为，不能是无过错的行为。2.损失原因：造成损失的行为必须是损失的原因之一，不能是附带行为或者不相关行为。3.损失连带性：承担民事责任的人必须直接责任连带于造成损失的行为，如果没有直接关联，那么也不需要承担民事责任。总之，当出现道路交通事故中的民事责任问题时，需要根据具体情况来判断，如果当事人的行为具有过错，并且由此给他人造成了损失，那么该当事人就应当承担民事责任，进行赔偿等相应处理。"

}

LLM self-reflection

来自一个blog: [Can LLMs Critique and Iterate on Their Own Outputs?]
Some insights:

1. Interestingly, this capability seems to be emergent in GPT-4 but not GPT-3.5 or Claude
2. **Nonetheless, I'm fairly convinced now that LLMs can effectively critique outputs better than they can generate them, which suggests that we can combine them with search algorithms to further improve LLMs**
3. Like most algorithmic ideas in probabilistic inference and optimal control, having an agent critique its decisions to make them better is an old idea that has been re-implemented over and over again

4 **However, if autoregressive generation makes a mistake, CoT prompting cannot go back and fix the error. The benefit of self-reflection is that the model can identify mistakes (potentially using CoT prompting itself), and correct them by starting over entirely. As neural net context length in LLMs increase, I expect that self-reflection will become the more effective CoT prompting technique. If you really squint and stretch your imagination, you can think of reflection as similar to a denoising operator for LLM outputs, similar to diffusion modeling but operating in semantic and logical space**

来自一篇paper讲解: Self-Refine: Iterative Refinement with Self-Feedback, 这篇paper主要是一个LLM进行self-refine prompt优化

Contrastive Learning

Contrastive Learning的一个中文综述知乎文章

Contrastive Learning的一个概述