核心部分

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

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

可以做的方向(详细版):

- 设计一个新的metric: 在demonstration selection时, 运用Influence Function/perplexity/mutual information/semantic distance/entropy等, 或者可以综合在一起
- 在demonstration order的问题上, 运用**curriculum learning**的思想, 把examplar由易到难排列起来。可以考虑用LLM自动化该过程, 让LLM承担**Difficulty Measurer** + **Training Scheduler**的"角色"。例如就可以用在APO的Expand上, 挑选从易到难的数据集。可以认为**curriculum learning**在training阶段可能会产生好的效果, 虽然调用api本质上来说是inference阶段, 但是APO在自动优化prompt时解释相当于training得到一个好的prompt

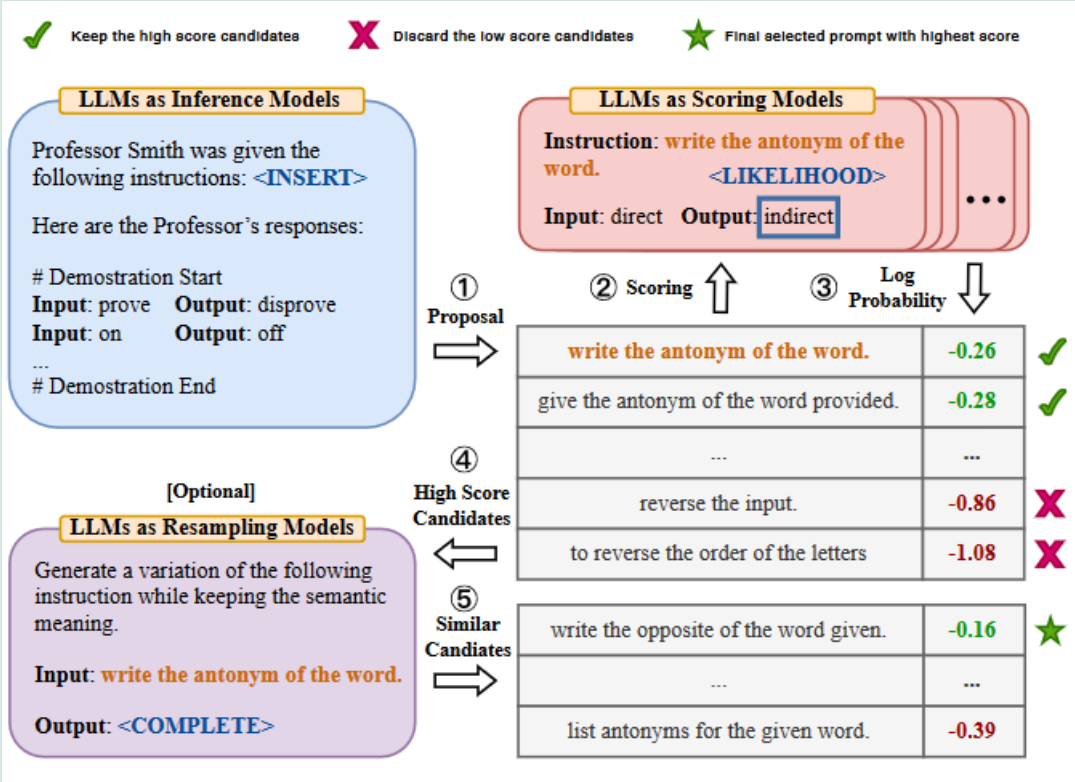
---

关于LLM as Optimizers的思考

现有的方法

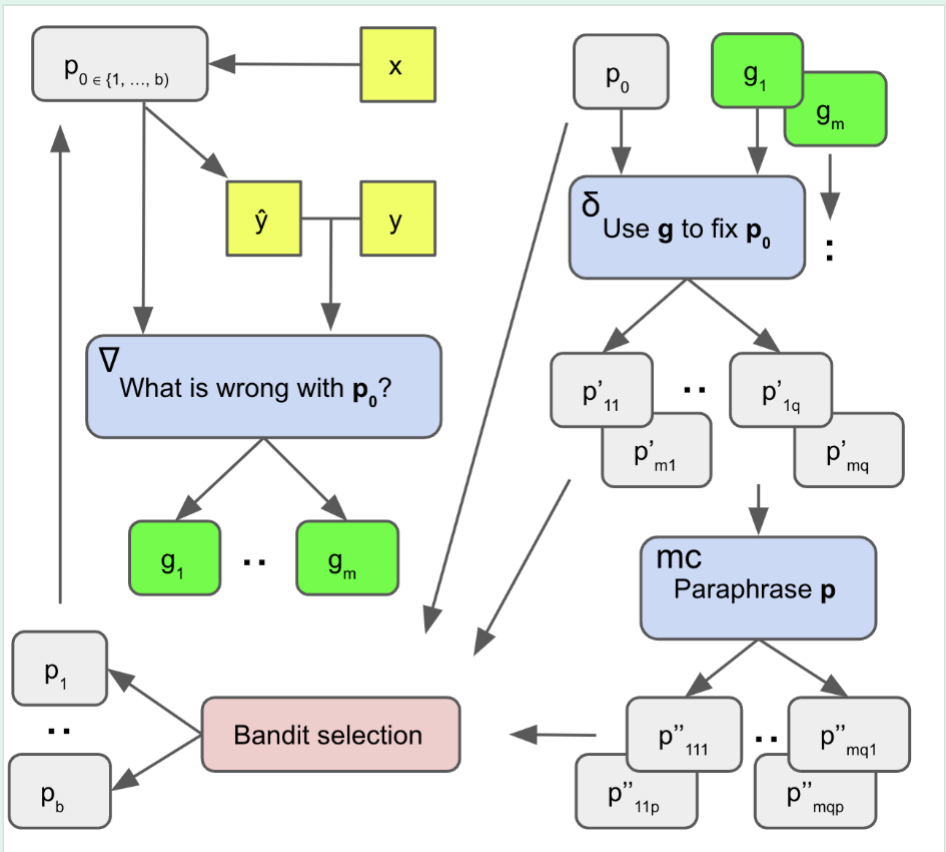
1. `Resampling-based methods`(选择语义类似的prompt)
    - Iterative-APE/APE
2. `Reflection-based methods`(可能发生语义改变的prompt)
    - APO
    - PromptAgent
    - OPRO(implicit reflection-based method): 在有些task上表现相对较差, 一个是因为优化方法没有方向导致LLM不知道什么是"better prompts", 另一个可能是因为LLM并不一定在最优的prompt附近sample

> 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)

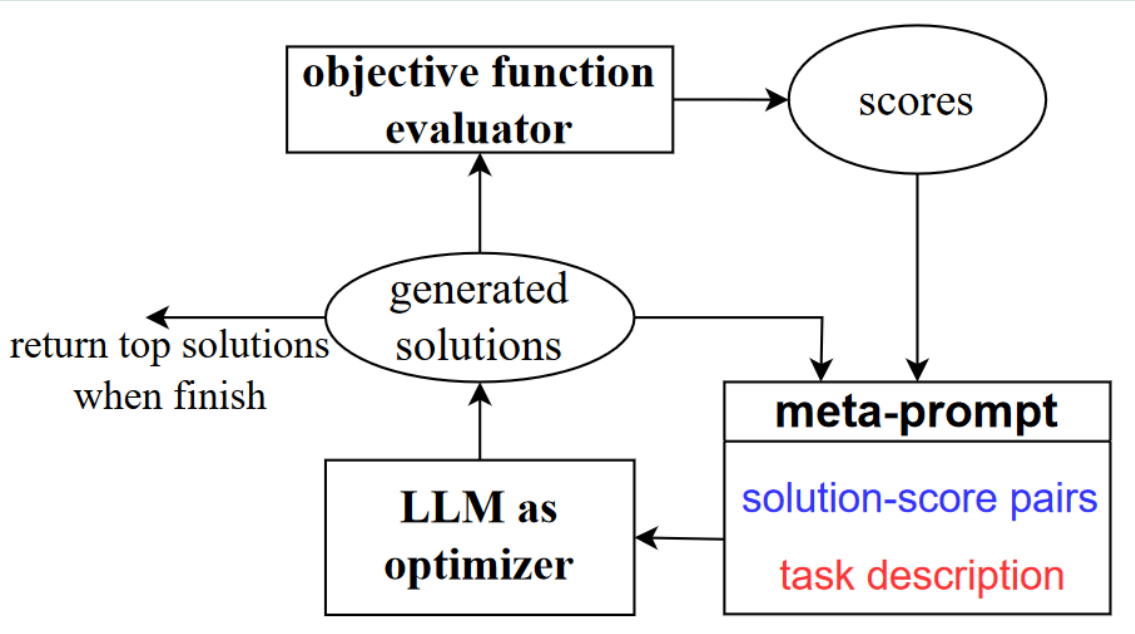
APO: 第一个prompt ∇ 根据初始的prompt p_0 产生一系列的梯度 g; 第二个prompt δ 利用梯度 g 来edit当前的prompt p_0



1. 初始的prompt为 p_0

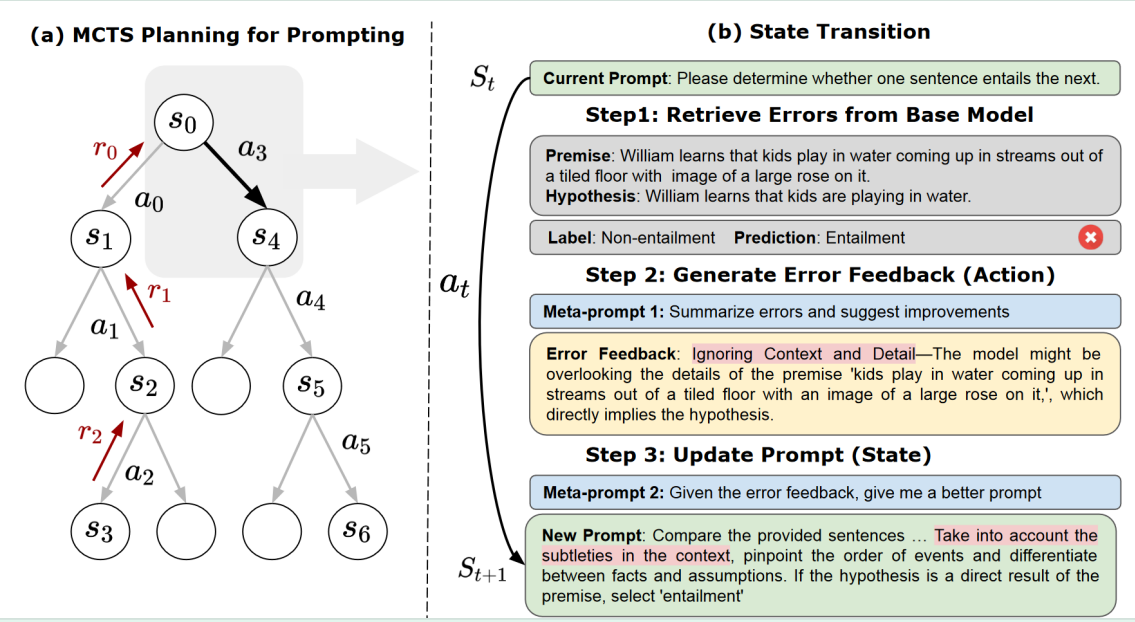
2. 通过一个Expand算法, 根据当前的prompt, 先生成梯度 g , 然后根据梯度 g 来修改当前的prompt, 最后运用一个**paraphrasing LLM**($called \quad LLM_{mc}$)在新的prompt candidates周围explore local monte carlo search space(**要求LLM生成跟输入语义相似但是用词不同的新candidates**)
3. 通过算法**UCB Bandits**或者算法**Successive Rejects**来挑选prompts

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

PromptAgent: 将prompt optimization的问题建模成**MDP(Markov Decision Process)**, 其中State为每次迭代的prompt, Action为对当前prompt可能的改动



1. 产生action发生state变化需要两步: 1.从training samples中收集errors 2.对这些errors产生一些feedbacks
2. 剩下的步骤就是扩展如左图中的tree, 使用**MCTS(Monte Carlo Tree Search)**来更新一个function $Q(s_t, a_t) : S \times A \rightarrow R$, 这个function用于look ahead and estimate the potential rewards。该过程有四个步骤: **1.Selection 2.Expansion 3.Simulation 4.Back-propagation**

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的幻觉问题
4. LLM对prompt format比较sensitive, 语义相似的prompts可能表现得截然不同
5. 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
6. model的**shortcut learning**问题: 知乎文章、Paper: Shortcut Learning in Deep Neural Networks、判断model真正work

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.

   , 探究修改后的prompt对instruction following能力的影响
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? 可不可以用LLM生成demonstration? 可以考虑用**Influence Function**
3. 评价指标的设计不同对自动优化prompt的影响
4. tell LLM **long prompts / too many tokens are expensive**

笔记部分

Concept/Concept Comparison

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。"
}
```


关于LLM的一些小tricks/(面试)知识

- 训练节省显存: 知乎文章, 主要介绍了如**Gradient Accumulation(梯度累积)、Gradient Checkpointing(梯度检查点)、Mixed Precision(混合精度训练)、Low-Precision Training(低精度训练, 直接使用BF16)、Efficient Optimizer、PEFT**等
- 1B模型对应多少显存: 知乎文章, B和G都是10亿(1000M/1024M), 1B模型参数对应多少G和**参数精度**有关。全精度训练(fp32)一个参数对应32bits, 也即4bytes；半精度训练(fp16、bf16)一个参数对应16bits, 也即2bytes。

  > 训练时的显存开销包括:
  >
  > 1. 模型参数本身
  > 2. 梯度: 一个参数对应一个梯度值
  > 3. 优化器状态: Adam的一阶动量和二阶动量使得优化器状态所占显存是参数的2倍
  > 4. 输入数据、对应的中间的hidden states、输出数据

- LoRA显存占用省在哪: 知乎文章, 节省在**优化器状态部分**。假设参数显存占用为x, 使用Adam优化器全量训练显存占用$4x$, LoRA冻住主干参数, 增加$m\%$的可训练LoRA权重, 显存占用为$(1+4m\%)x$

  > **参数部分:** $(1+m\%)x$
  >
  > **梯度部分:** $(0+m\%)x$ (lora进去了之后直接冻结参数?应该就不用算原来的梯度了)
  >
  > **优化器状态部分:** $2m\%x$

- 为什么现在的LLM都是Decoder only的架构?
- 大模型调参tricks: 知乎文章, 大部分超参取固定最优值即可。当计算量增加时, 调大batch size调小learning rate; 当计算量减小时, 调小batch size调大learning rate
- LLM的一些参数:

  > Context Window(上下文窗口): 单词预测时, model可以参考的周围词的数目
  >
  > Hidden Layer Size(隐藏层大小): 指的是隐藏层神经元数目, 通过非线性变换将输入层和输出层连接起来

论文写作NLP

一篇Medium上的文章: Tips for Writing NLP Papers

内容

- **不要忘记为什么:** 论文需要明确阐明 1)试图回答哪些研究问题 以及 2)为什么它们很重要
- **由抽象到具体:** 将更多技术细节留给方法、数据和实验设置部分
- **最后写引言、摘要和结论:** 需要涵盖以下方面 1️⃣工作的动机 2️⃣与之前工作的差距 3️⃣提出的工作 4️⃣发现 5️⃣结束语(通常涉及未来工作、局限性或发现的影响)

语言

- **保持简单、词汇不要过于复杂**
- **要简洁、清晰具体**
- **不要写过长的句子**

与之前的工作比较

- **了解文献**
- **相关工作不是购物清单:** 不要仅仅列出与你的论文相关的论文。尝试按主题对其进行分组, 得出结论, 并用它们来强调你的工作旨在解决文献中的空白
- **要有批判精神**

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

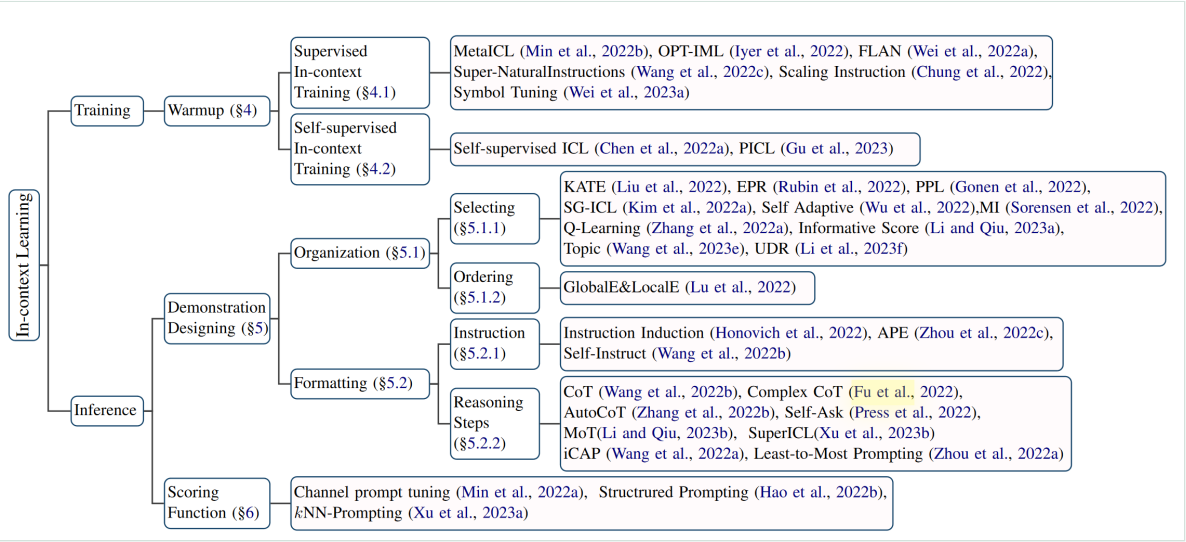
4. **However, if autoregressive generation makes a mistake, CoT prompting cannot go back and fix the error. The benefit of self-reflection is that the model can identify mistakes (potentially using CoT prompting itself), and correct them by starting over entirely. As neural net context length in LLMs 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

In-Context Learning

有一篇综述文章: A Survey on In-context Learning

Definition: In-context learning is a paradigm that allows language models to learn tasks given only a few examples in the form of demonstration.



Demonstration Organization

Demonstration Selection

Unsupervised Method

- KATE: 主要思路是借助一个**sentence encoder**, 对当前问题x进行编码。通过KNN算法找到train set中k个语义最接近(评价指标为distance)的samples, 按照跟问题x的相似度进行排序, 越相似的示例放在越后面。最终输入为**demonstrations + x**
- An Information-theoretic Approach to Prompt Engineering Without Ground Truth Labels: 评价指标换为**Mutual Information(互信息)**, 选择**input和相应model output**互信息最大的template
- Demystifying Prompts in Language Models via Perplexity Estimation: 评价指标换为**perplexity**, 选择**perplexity**最低的prompt
- Diverse Demonstrations Improve In-context Compositional Generalization: 选择diverse demonstrations(这个感觉跟做的task有关系, 有些task可能也选不出diverse demonstrations)
- Self-Generated In-Context Learning: Leveraging Auto-regressive Language Models as a Demonstration Generator: 用LLM自己生成demonstrations, 减少对外部input-label pairs的依赖
- In-context Example Selection with Influences: 采用Influence Function对demonstration进行选择

- Finding Support Examples for In-Context Learning: 提出了一个新的metric为**InfoScore**来描述demonstrations的信息含量多少, 首先过滤掉uninformative的examples, 然后迭代修改评估demonstrations(diversity-guided)
- 用LLM迭代优化选出来的demonstrations: Misconfidence-based Demonstration Selection for LLM In-Context Learning

Supervised Method

- Learning To Retrieve Prompts for In-Context Learning: two-stage retrieval方法, 对于给定的一个input, 首先build一个unsupervised retriever recall similar examples as candidates, 然后build一个supervised retriever选择candidates。scoring LM评估每个candidate和input的连接, 得分高的candidates标记为positive, 否则标记为negative
- Large Language Models Are Latent Variable Models: Explaining and Finding Good Demonstrations for In-Context Learning: 基于prompt tuning的方法, 应该是调整soft prompt?
- Active Example Selection for In-Context Learning: 基于RL的方法, 建模成MDP问题, 通过Q-learning算法解决

Demonstration Ordering

- What Makes Good In-Context Examples for GPT-3?: 根据与input的semantic distance排序选择demonstration
- Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity: 用熵(entropy)来衡量从而将选择的demonstration排序

Demonstration Formatting

Instruction Formatting

- Instruction Induction: From Few Examples to Natural Language Task Descriptions: 给定一些demonstrations, 让LLM自己生成task instruction
- Large Language Models Are Human-Level Prompt Engineers: APE算法
- Self-Instruct: Aligning Language Models with Self-Generated Instructions: LLM自动生成Instruction

Reasoning Steps Formatting

- Chain-of-Thought Prompting Elicits Reasoning in Large Language Models: CoT(Chain-of-Thought) prompt
- Automatic Chain of Thought Prompting in Large Language Models: **Auto-CoT**, 自动构建包含问题和推理链的说明信息。整个算法分为两步: 1.question clustering(用K-means方法将数据集中的question划分到k个簇中) 2.demonstration sampling(从每个簇中选择一个代表性question, 采用具有启发性的Zero-Shot-CoT方法生成推理链)
- iCAP: 提出一个context-aware的prompter可以在每步reasoning时动态调整contexts

Scoring Function

Scoring Function	Target	Efficiency	Task Coverage	Stability
Direct	$\mathcal{M}(y_j \mid C, x)$	+++	+	+
PPL	$\mathrm{PPL}(S_j)$	+	+++	+
Channel	$\mathcal{M}(x \mid C, y_j)$	+	+	++

Factors/Why ICL works

Inference	Label space exposure (Min et al., 2022c) Demonstration input distribution (Min et al., 2022c) Format of input-label pairing (Min et al., 2022c; An et al., 2023) Demonstration input-label mapping (Min et al., 2022c; Kim et al., 2022b) (Wei et al., 2023b) Demonstration sample ordering (Lu et al., 2022) Demonstration-query similarity (Liu et al., 2022) Demonstration diversity (An et al., 2023) Demonstration complexity (An et al., 2023)

> Counter-intuitively, **the input-label mapping matters little to ICL**. When a model is large enough, it will show an emergent ability to learn input-label mappings, even if the labels are flipped or semantically-unrelated

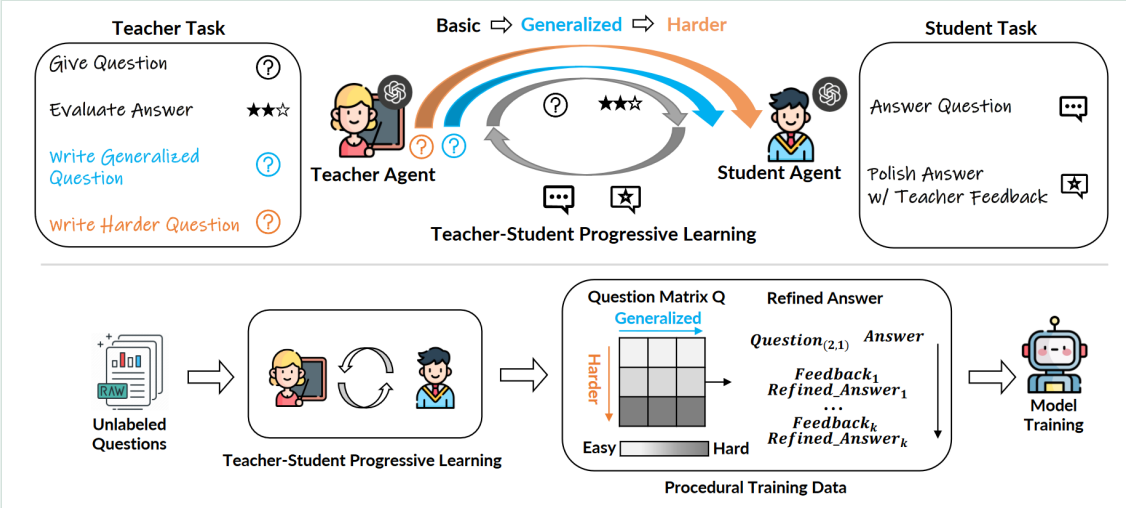
一个关于ICL work的数学解释: An Explanation of In-context Learning as Implicit Bayesian Inference。the ICL ability emerges when the pretraining distribution follows **a mixture of hidden Markov models**

Popular Learning Methods

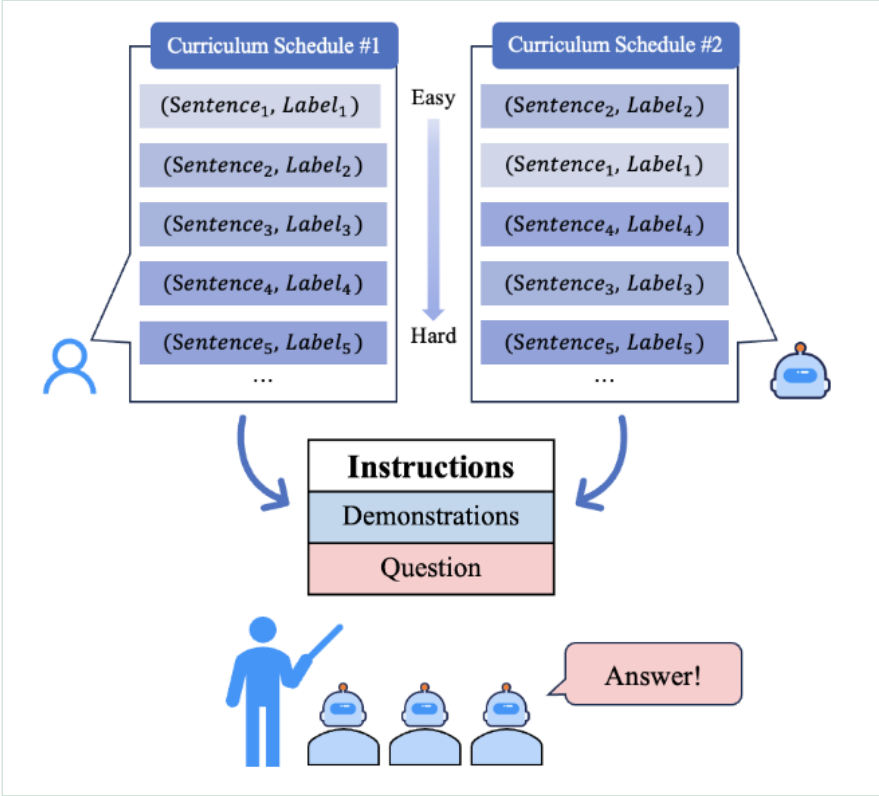
Curriculum Learning

> 从易到难学习, 感觉可以类比到In Context Learning上, 对每个demonstration进行难易评价, 然后每个prompt从易到难插入demonstration
>
> 感觉也不太对, CL更像是training阶段用的trick
>
> 两个survey: A Survey on Curriculum Learning、Curriculum Learning: A Survey
>
> CL中两个重要的东西: **Difficulty Measurer** + **Training Scheduler**, 前者决定学习的优先程度后者决定学习数据的多少。可以用LLM担任这两个"角色"

- YODA : Teacher-Student Progressive Learning for Language Models: 这篇paper主要是提出了Teacher-Student agents的架构, 用原始的unlabelled questions, 通过teacher和student agent的不断迭代, 产生feedback、answer等。最终将数据用于训练开源LLM (e.g. LLaMA2)



- [Let's Learn Step by Step: Enhancing In-Context Learning Ability with Curriculum Learning]: 这篇paper的主要思想是将curriculum learning和ICL结合在一起, 将demonstration的order设置成从simple到difficult。实验证明ICCL效果主要在tuned model上会好一点, 在base model上的结果甚至不如ICL



Contrastive Learning

[Contrastive Learning的一个中文综述知乎文章]

[Contrastive Learning的一个概述]

Meta Learning

一篇blog: [Meta-Learning: Learning to Learn Fast]、[对应的中文版]