



東南大學
SOUTHEAST UNIVERSITY



Optimizing generative AI by backpropagating language model feedback

Nature 2025

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan
Lu, Zhi Huang, Carlos Guestrin & **James Zou**

Stanford University

汇报人：张兴才

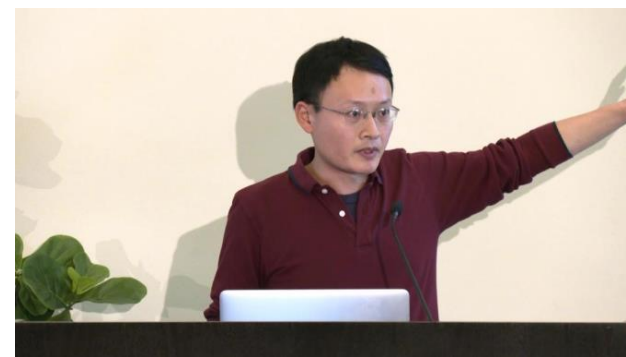
2025年 6月 20日

Research Interests

- self-improving
- controllable AI



Mert Yuksekgonul
Stanford University



James Zou
Stanford University

- [1] Attention Satisfies: A Constraint-Satisfaction Lens on Factual Errors of Language Models. ICLR' 2024
- [2] When and why vision-language models behave like bags-of-words, and what to do about it? ICLR' 2023
- [3] A visual-language foundation model for pathology image analysis using medical Twitter. Nature Medicine' 2023
- [4] Post-hoc Concept Bottleneck Models. ICLR' 2023

1

背景

2

方法

3

实验

4

总结

1

背景

2

方法

3

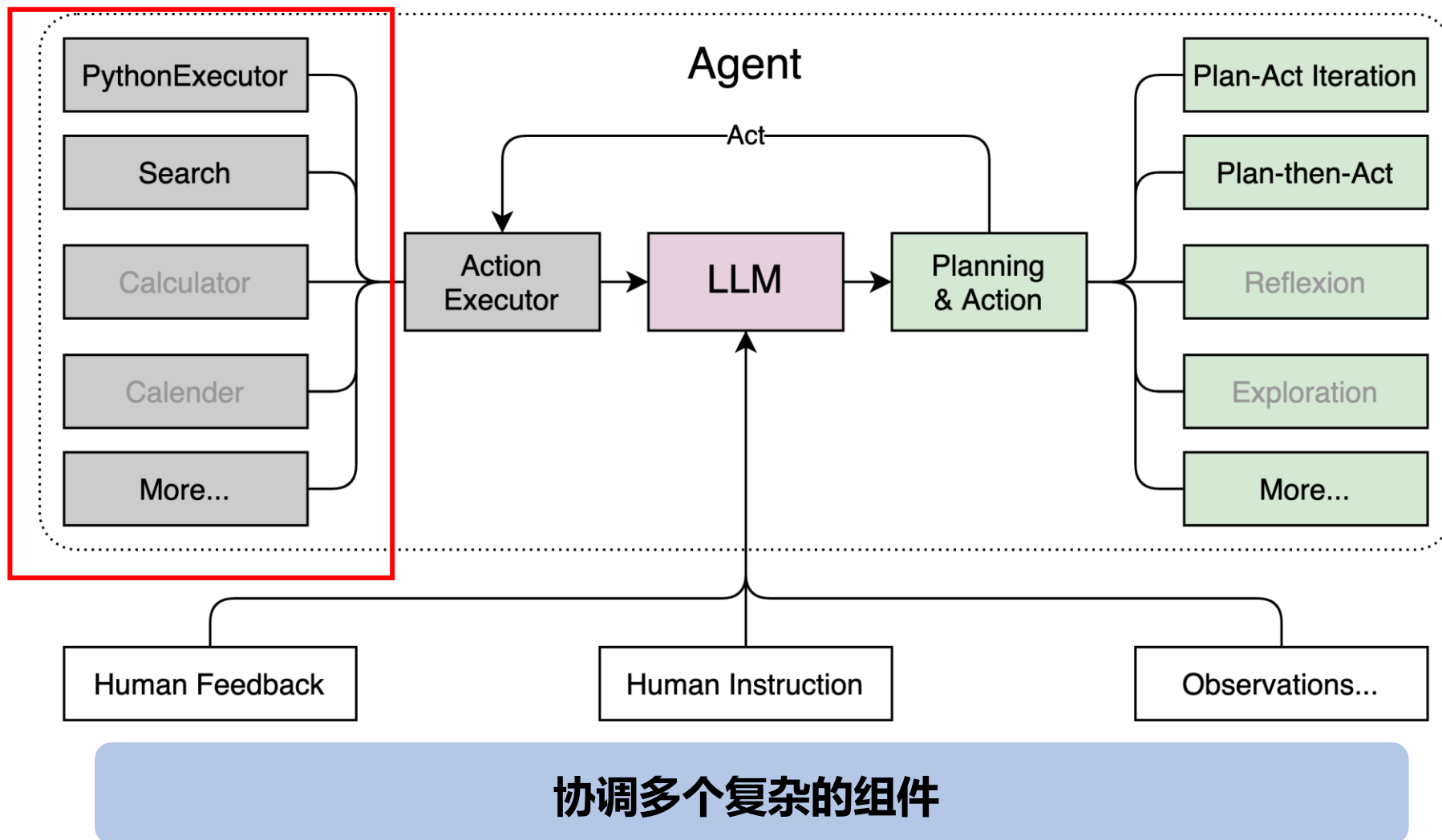
实验

4

总结

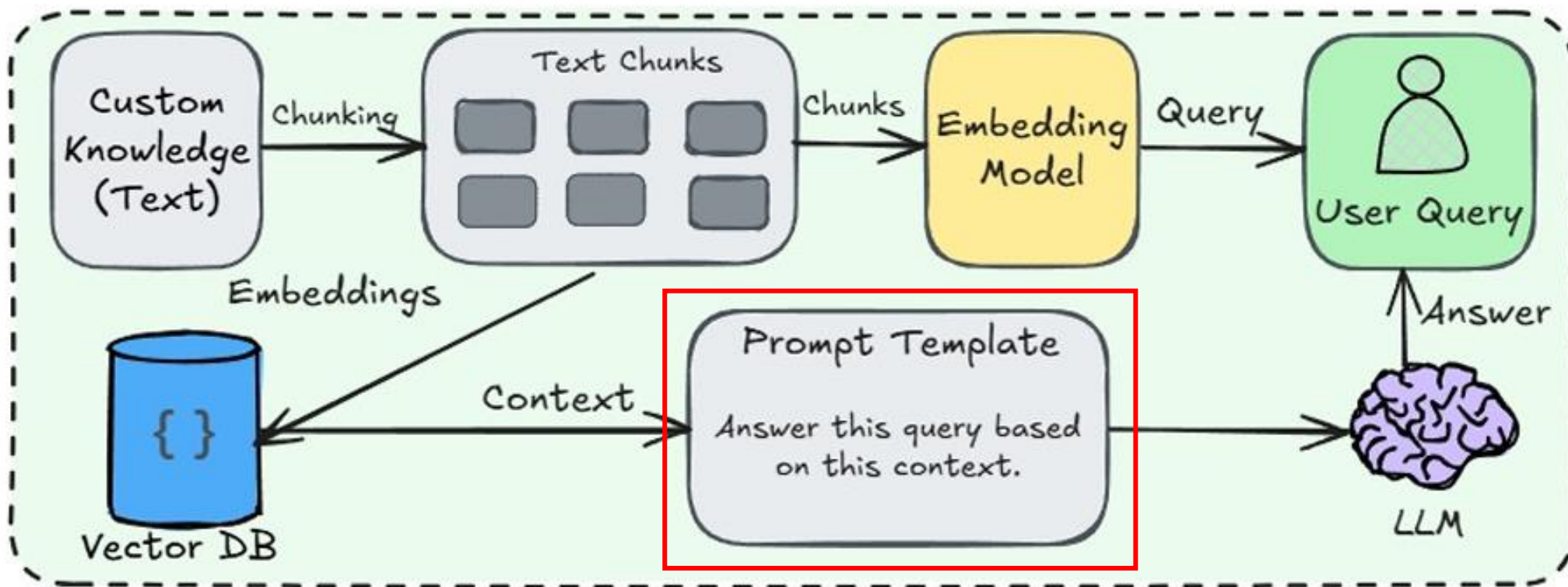
背景：复合式AI系统 (Compound AI System)

新一代的AI应用



背景：复合式AI系统 (Compound AI System)

手工构建prompt的AI系统



依赖领域专家手工构建和调整，造成高昂的开发和维护成本

背景：复合式AI系统 (Compound AI System)

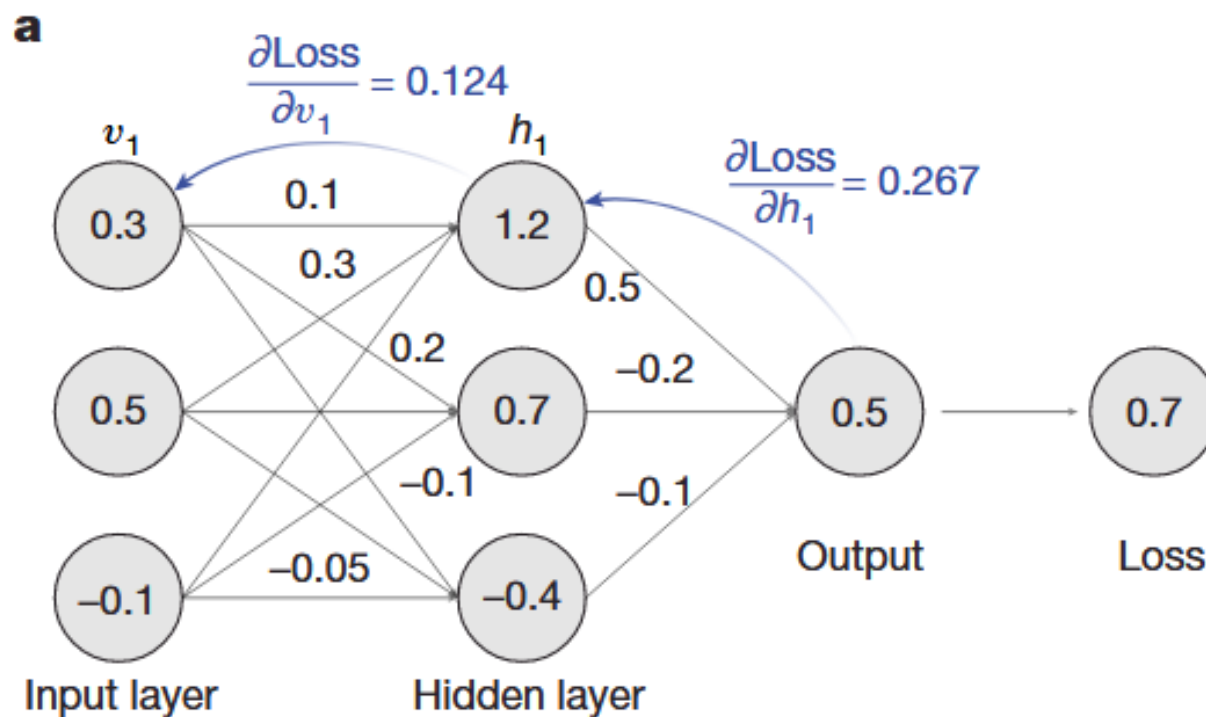
手工构建prompt的AI系统



依赖领域专家手工构建和调整，造成高昂的开发和维护成本

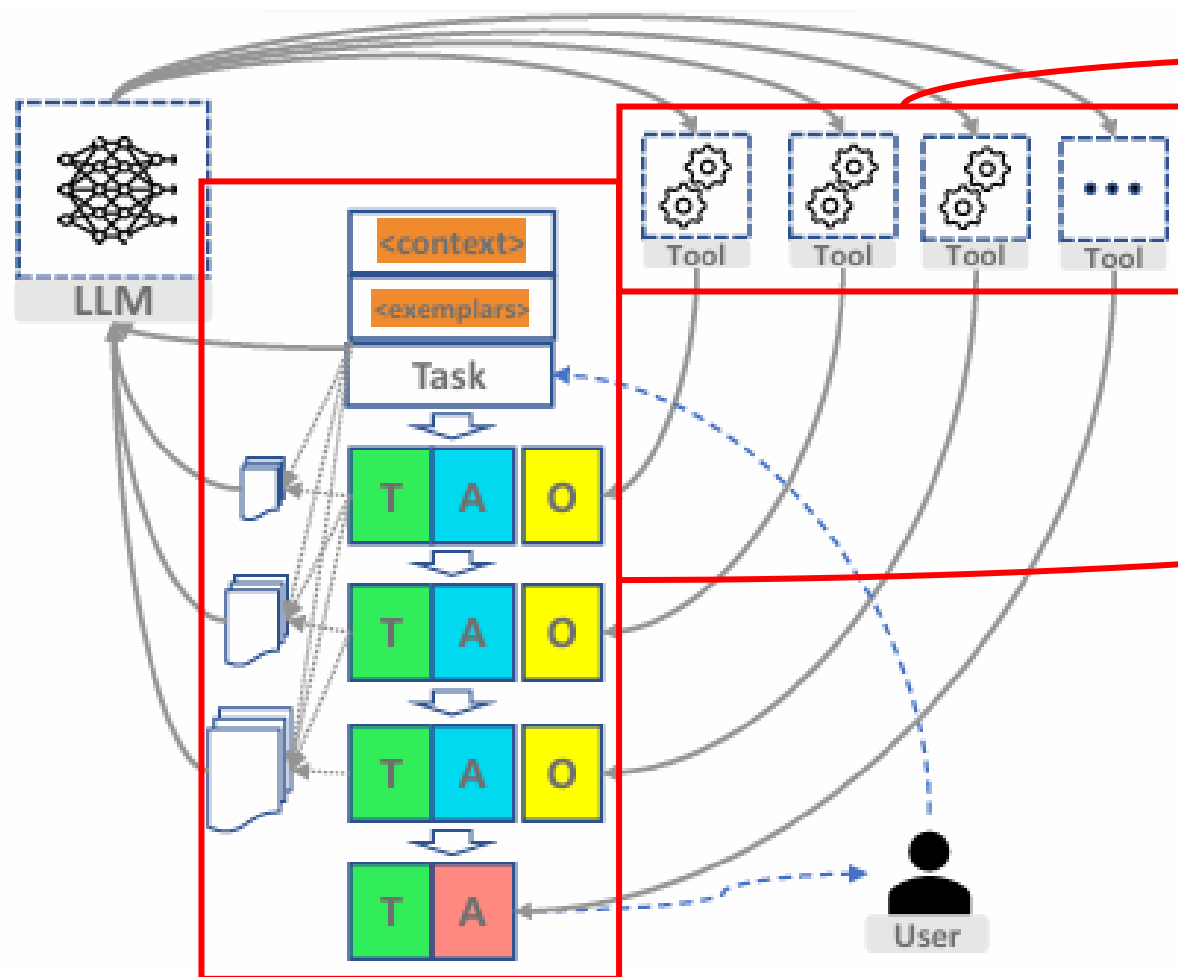
机遇：神经网络的数值梯度优化

➤ 神经网络的数值反向传播



通过反向传播传递数值梯度从而修改模型权重提高表现

挑战：传统梯度方法的失效



➤ 复合式AI系统特性

- 由LLM API、搜索引擎等工具组件构成
- 自然语言交互

传统数值梯度的反向传播变得困难

提纲

1

背景

2

方法

3

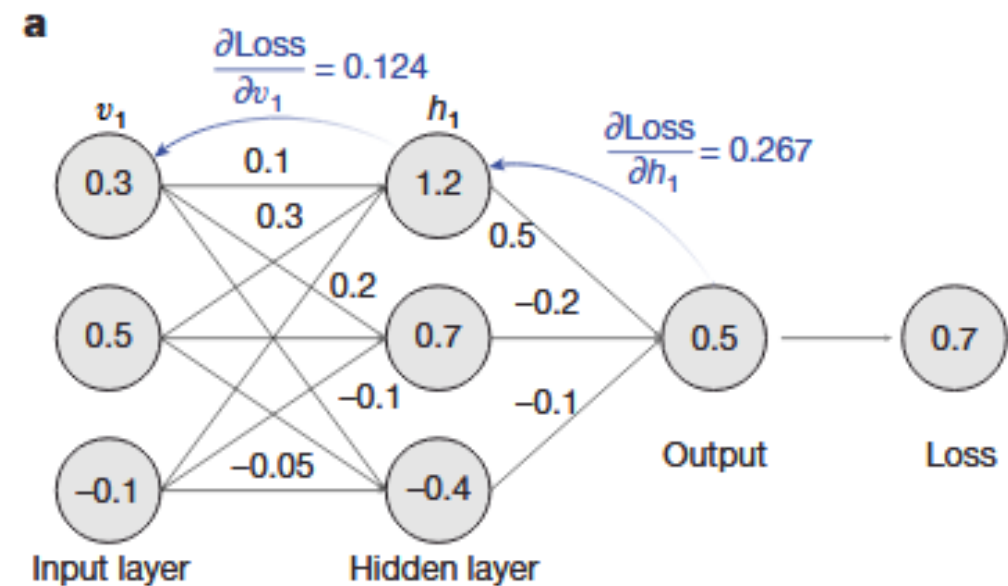
实验

4

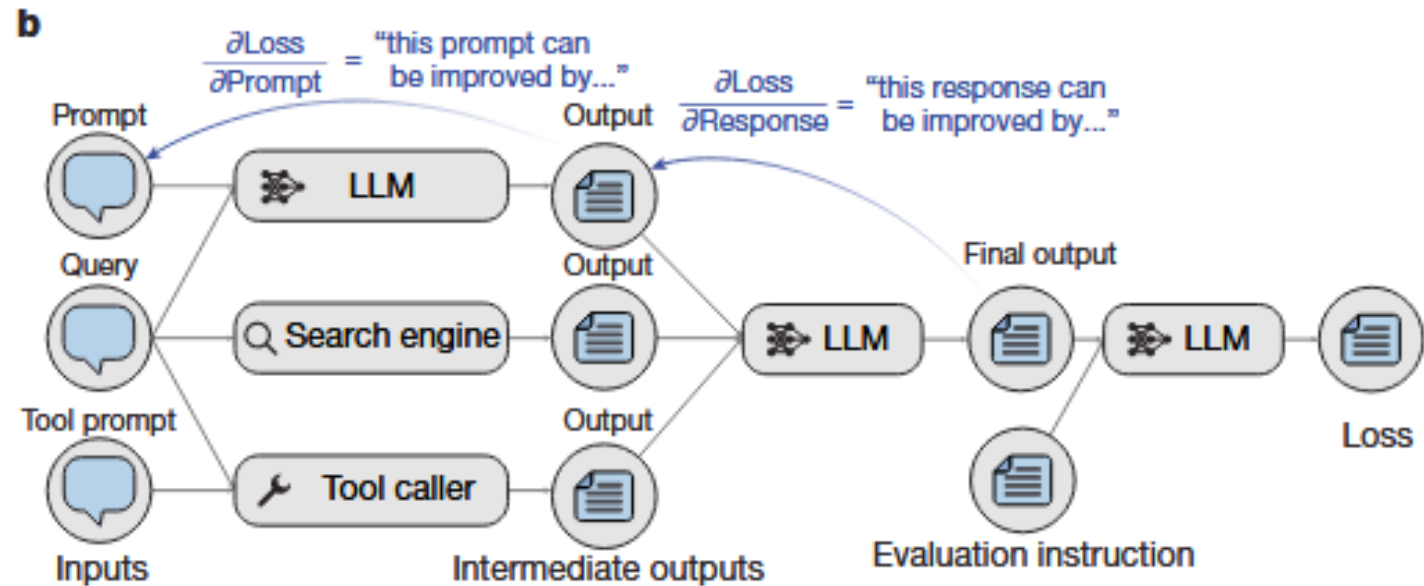
总结

方法：TextGrad

整体思路：



数值梯度的反向传播

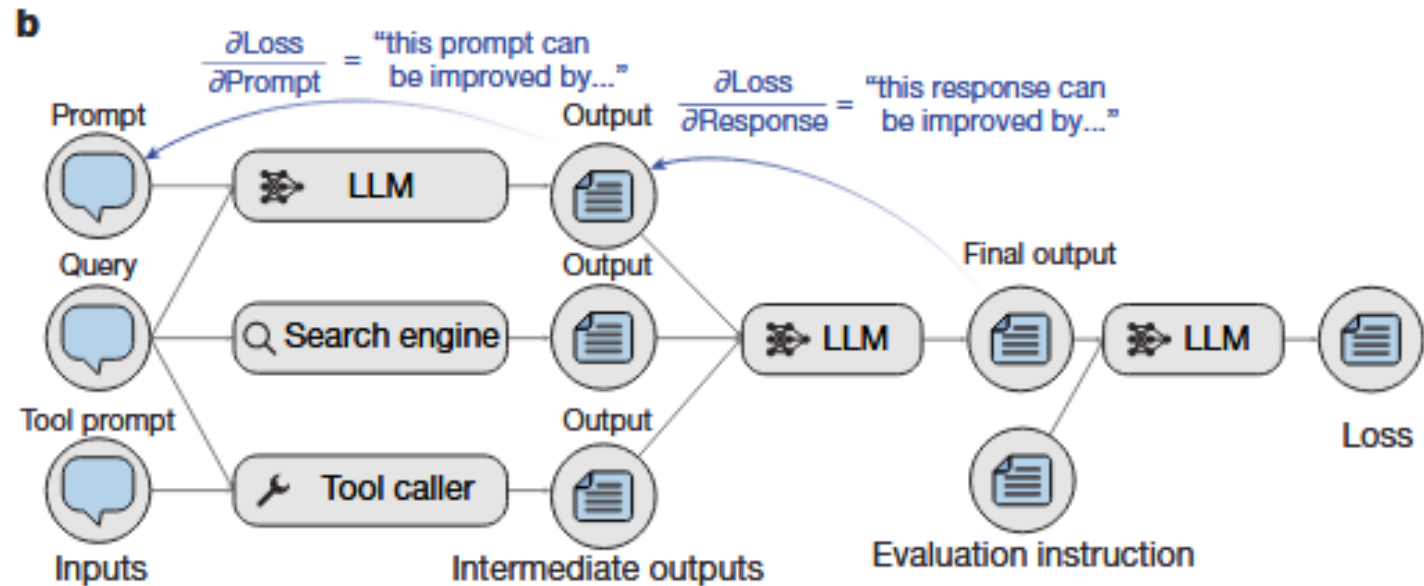


自然语言梯度的反向传播

方法：TextGrad

计算图 (computation graph) 由变量 (Variables) 构成：

- **值 (Value)**：非结构化数据（一般为文本数据）
- **角色描述 (Role description)**：描述变量在计算图中的作用
- **是否需要优化 (Requires grad)**：反向传播过程中是否可以更新变量的值




```
1 import textgrad as tg
2
3 system_prompt = tg.Variable("You will be given a question and think step
  -by-step.", requires_grad=True, role_description="system prompt to the
  language model that will be reused across queries")
4
5 model = tg.BlackBoxLLM(system_prompt=system_prompt)
```

方法: TextGrad

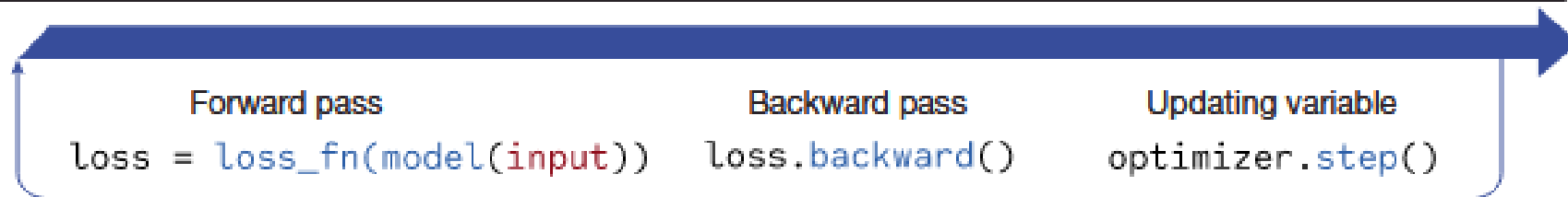
TextGrad与PyTorch使用方法相似:

1. Analogy in abstractions

	Maths	PyTorch	 TextGrad
Input	x	<code>Tensor(image)</code>	<code>tg.Variable(article)</code>
Model	$\hat{y} = f_{\theta}(x)$	<code>ResNet50()</code>	<code>tg.BlackboxLLM("You are a summarizer.")</code>
Loss	$L(y, \hat{y}) = \sum y_i \log(\hat{y}_i)$	<code>CrossEntropyLoss()</code>	<code>tg.TextLoss("Rate the summary.")</code>
Optimizer	$GD(\theta, \frac{\partial L}{\partial \theta}) = \theta - \frac{\partial L}{\partial \theta}$	<code>SGD(list(model.parameters()))</code>	<code>tg.TGD(list(model.parameters()))</code>

2. Automatic differentiation

PyTorch and TextGrad share the same syntax for backpropagation and optimization



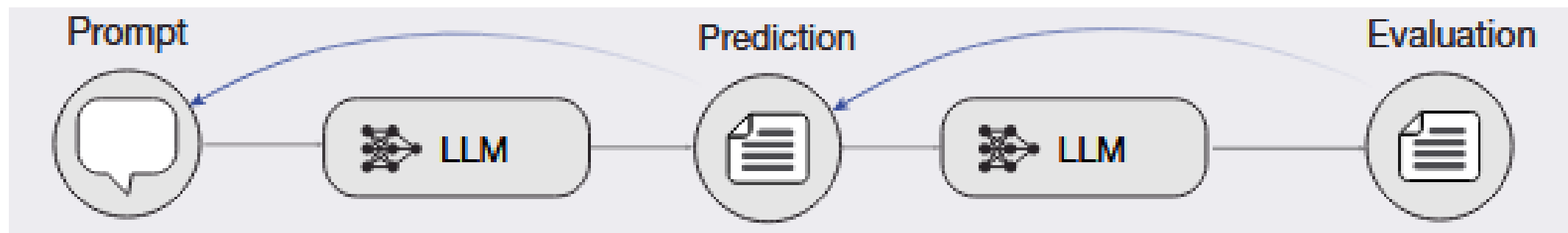
- **任务:** 调用LLM总结文章
- **输入:** 文章
- **输出:** 总结内容

- **Loss:** 对于总结的评估
- **优化目标:** 总结内容、
LLM prompt

方法: TextGrad

前向传播:

- 包含两个LLM调用的AI系统 (预测LLM+评估LLM)
- 计算图:



$$\text{Prediction} = \text{LLM}(\text{Prompt} + \text{Question}), \quad (1)$$

$$\text{Evaluation} = \text{LLM}(\text{Evaluation Instruction} + \text{Prediction}), \quad (2)$$

$$\text{Prompt} + \text{Question} \xrightarrow{\text{LLM}} \text{Evaluation Instruction} + \text{Prediction} \xrightarrow{\text{LLM}} \text{Evaluation}, \quad (3)$$

方法: TextGrad

反向传播:

Example implementation for the gradient operator

优化prompt

$$\frac{\partial \mathcal{L}}{\partial x} = \nabla_{\text{LLM}}(x, y, \frac{\partial \mathcal{L}}{\partial y}) \triangleq \text{"Here is a conversation with an LLM: \{x|y\}."} \quad (6)$$

+

prompt

预测结果

LLM(Here is a conversation with an LLM: \{x|y\}.

Below are the criticisms on \{y\}:

$\left\{ \frac{\partial \mathcal{L}}{\partial y} \right\}$ 对预测结果评估

Explain how to improve \{x\}.),

➤ **X**: prompt **Y**: 预测结果

➤ **L**: 调用LLM

➤ $\partial \mathcal{L} / \partial y$: 对预测结果评估

➤ $\partial \mathcal{L} / \partial x$: 优化prompt

方法：TextGrad

反向传播过程中提供给评估LLM的信息：

Glossary for Backward Mode of the LLMCall function

Glossary of tags that will be sent to you:

- # - <LM_SYSTEM_PROMPT>: The system prompt for the language model.
- # - <LM_INPUT>: The input to the language model.
- # - <LM_OUTPUT>: The output of the language model.
- # - <OBJECTIVE_FUNCTION>: The objective of the optimization task.
- # - <VARIABLE>: Specifies the span of the variable.
- # - <ROLE>: The role description of the variable.

- 预测LLM prompt
- 预测LLM输入
- 预测LLM输出
- 优化目标
- 计算图中的变量
- 变量的角色描述

方法: TextGrad

评估LLM的system prompt:

System prompt for the backward mode of the LLMcall function

You are part of an optimization system that improves a given text (i.e. the variable). You are the gradient (feedback) engine. Your only responsibility is to give intelligent and creative feedback and constructive criticism to variables, given an objective specified in `<OBJECTIVE_FUNCTION </OBJECTIVE_FUNCTION>` tags. The variables may be solutions to problems, prompts to language models, code, or any other text-based variable. Pay attention to the role description of the variable, and the context in which it is used. You should assume that the variable will be used in a similar context in the future. Only provide strategies, explanations, and methods to change in the variable. DO NOT propose a new version of the variable, that will be the job of the optimizer. Your only job is to send feedback and criticism (compute 'gradients'). For instance, feedback can be in the form of 'Since language models have the X failure mode...', 'Adding X can fix this error because...', 'Removing X can improve the objective function because...', 'Changing X to Y would fix the mistake ...', that gets at the downstream objective.

If a variable is already working well (e.g. the objective function is perfect, an evaluation shows the response is accurate), you should not give feedback.

{GLOSSARY}



实验：实验设置

任务类型: 编程、科学难题、推理prompt、化学结构、医学治疗方案.

- 编程任务:

dataset: LeetCode Hard数据集

baseline: Reflexion、gpt-4o

- 科学难题:

dataset: CoT、Best reported

baseline: Google-proof QA、MMLU-Machine Learning、MMLU-College Physics

- 推理prompt:

dataset: Object Counting、Word Sorting、GSM8k

baseline: CoT、DSPy

所有的实验中，推理模型使用**gpt-3.5-turbo**，评估模型使用**gpt-4**

评估：代码优化

$$\text{Code-Refinement Objective} = \text{LLM}(\text{Problem} + \text{Code} + \text{Test-time Instruction} + \text{Local Test Results}) \quad (14)$$

- 优化目标：代码
- 输入：问题、先前的代码、评估指令以及本地的测试结果

Code Refinement Objective

LLM("You are an intelligent assistant used as an evaluator, and part of an optimization system. You will analyze a code implementation for a coding problem and unit test results. The code will be tested with harder tests, so do not just check if the code passes the provided tests. Think about the correctness of the code and its performance in harder test cases. Give very concise feedback. Investigate the code problem and the provided implementation. For each failed unit test case, start analyzing it by saying "The code did not pass this test because...". Explain why the current implementation is not getting the expected output. Do not provide a revised implementation. Carefully suggest why there are issues with the code and provide feedback.

{Test-time Instruction}

The coding problem:

{Problem}

Code generated that must be evaluated for correctness and runtime performance

{Code}

The test results:

{Local test Results}

e TextGrad for code optimization

```
for i in range(n):
    if nums[i] < k:
        balance -= 1
    elif nums[i] > k:
        balance += 1
    if nums[i] == k:
        result += count.get(balance, 0) +
            count.get(balance - 1, 0)
    else:
        result += count.get(balance, 0)
        count[balance] = count.get(balance, 0) + 1
```

Code at iteration t

```
for i in range(n):
    if nums[i] < k:
        balance -= 1
    elif nums[i] > k:
        balance += 1
    else:
        found_k = True
        if nums[i] == k:
            result += count.get(balance, 0) +
                count.get(balance - 1, 0)
        else:
            count[balance] = count.get(balance, 0) + 1
```

Code at iteration t+1

Gradients

Handling `nums[i] == k`: The current logic does not correctly handle the case when `nums[i] == k`. The balance should be reset or adjusted differently when `k` is encountered. ...

Task	Method	Completion Rate
LeetCode Hard [26]	Zero-shot [26]	0.26
	Reflexion (1 demonstration, 5 iterations) [26]	0.31 ± 0.012
	TEXTGRAD (0 demonstrations, 5 iterations)	0.36 ± 0.018

实验：科学难题

$$\text{Solution Refinement Objective} = \text{LLM}(\text{Question} + \text{Solution} + \text{Test-time Instruction}) \quad (15)$$

- 优化目标：解决方案
- 输入：问题、先前的解决方案、评估指令

Solution Refinement Objective

LLM("Below is a multi-choice question and a prediction. You are a critical and creative scientist. Your job is to investigate the prediction. Critically go through reasoning steps, and see if there is a reason why the prediction could be incorrect.

Use the Janusian Process, think about whether alternative answers could be true.

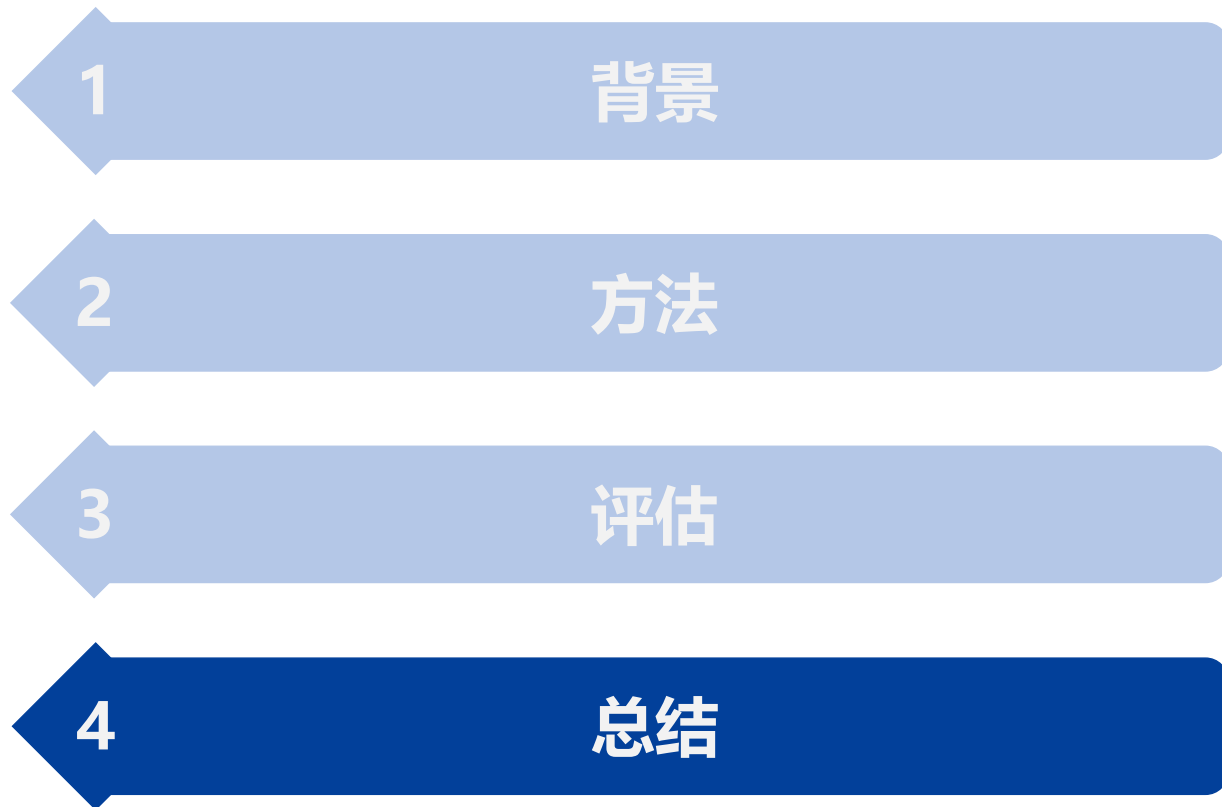
Question: {Question}

Answer by the language model: {Solution}")

```
1 # Assume we have the test_dataset
2 question, answer = test_dataset[i]
3 # Initialize the test time loss function
4 # This has the system prompt provided above as 'Solution Refinement Objective'
5 test_time_loss_fn = MultipleChoiceTestTime()
6 # Get a zero-shot solution from an LLM
7 solution: Variable = zero_shot_llm(question)
8 # Optimize the solution itself
9 optimizer = tg.TextualGradientDescent(parameters=[solution])
10
11 for iteration in range(max_iterations):
12     optimizer.zero_grad()
13     # Note how the loss is self-supervised (does not depend on any ground truth.)
14     loss = test_time_loss_fn(question, solution)
15     # Populate the gradients, and update the solution
16     loss.backward()
17     optimizer.step()
```

Dataset	Method	Accuracy (%)
Google-proof QA [27]	CoT [46, 47]	51.0
	Best reported [48]	53.6
	TEXTGRAD	55.0
MMLU-Machine Learning [45]	CoT [46, 47]	85.7
	TEXTGRAD	88.4
MMLU-College Physics [45]	CoT [46, 47]	91.2
	TEXTGRAD	95.1

提纲



Conclusion

- 提出了一种全新的优化框架：基于文本的方向传播
 - 将传统的数值梯度优化思想引入自然语言任务中，用自然语言反馈替代数值梯度
 - 将一个复合AI系统表示成一个计算图
- 通用性较强，可以用于代码任务、科学问题、医学、化学等多个领域的优化。
- 由将TextGrad语法设计成了与PyTorch类似的实现方式，便于学习的框架

Thinking

➤ 能否提高

- 从文章的所有例子和实验中看上去，目前的梯度优化仅支持对话的形式的AI系统，处在优化LLM本身推理能力上，还无法对包含工具调用的AI系统实现优化（与前文提到的机遇不一致）
- 文中的文本梯度是依赖于注入提示词的LLM进行评估，但是LLM的评估并不是一直正确的，比如对于开放性的问答任务，答案的评估标准是难以界定的
- 这种通过多轮反馈来得到正确结果的方式，必然会花费更多的token、时间

➤ 泛化：

- 文章已经涉及到了多种任务的优化，这一块没有想到

➤ 用到我们的idea中：

- 我们的工作中，反馈使用来评估agent的回答是否正确，从而决定是否存储记忆，能否将记忆进行进一步的反馈，从而提高记忆质量。
- 我们系统中的反馈目前可能并不是很完善，可以参照TextGrad的框架来优化一下我们的问答反馈机制



東南大學
SOUTHEAST UNIVERSITY

恳请各位老师与同学批评指正！