

# RAG 技术详解与实战应用

第10讲：探索Deepseek：

打造思维能力更强的RAG系统



# 目录



## 1. 上节回顾



## 2. Deepseek简介



## 3. Deepseek的部署和蒸馏



## 4. Deepseek赋能RAG



## 微调LLM大模型

- 大模型在RAG的位置：内容生成
  - 模型能力瓶颈：垂直领域知识、结构化输出、部署资源
- 微调基本概念
  - 微调能提升模型下游任务能力
  - 常用微调技术：全参、冻结、LoRA
  - LoRA微调原理与数学解析：引入低秩矩阵来模拟微调过程。
  - LoRA参数的建议
- 基于LazyLLM对大模型进行微调
  - 数据准备：Alpaca格式
  - 微调参数如何配置

## 大模型评测

- 设计针对阅读理解信息抽取任务的评价指标：
  - 精确匹配率
  - 语义相似度
  - 原文包含度
- 基于通用评价指标和针对阅读理解设计的评价指标进行评测：
  - 通用评测指标的失效；
  - 微调效果提升得到验证；
  - 微调小模型超过在线大模型；

## 微调Embedding模型

- Embedding模型在RAG系统中的作用
- 训练数据：问题、正负样本、Prompt
- 评测数据：问题及答案所在上下文
- 如何基于LazyLLM进行微调
- 如何在LazyLLM中比如RAG下使用微调好的模型：
  - 微调好的LLM
  - 微调好的Embedding



# 目录



## 1. 上节回顾



## 2. Deepseek简介



## 3. Deepseek的部署和蒸馏



## 4. Deepseek赋能RAG



## ➤ 模型简介

- DeepSeek是由中国人工智能公司DeepSeek开发的大语言模型，基于Transformer架构，采用混合专家模型（MoE）和多头潜在注意力机制（MLA），在推理效率和性能上表现出色。
- 其核心产品DeepSeek-V3和DeepSeek-R1拥有6710亿参数，每次激活参数量为370亿，大幅降低了计算成本。

## ➤ 技术特点

- **混合专家模型（MoE）**：将模型分成多个专家模块，每个任务仅激活少量专家，减少参数量，提升效率。
- **多头潜在注意力机制（MLA）**：通过低秩压缩技术减少Key-Value缓存，显著提升推理效率。
- **训练方式**：采用大规模强化学习与高质量合成数据结合，无需依赖大量标注数据。

## ➤ 应用场景

- 自然语言处理：文本生成、翻译、情感分析等。
- 智能对话：智能客服、聊天机器人。
- 代码生成：支持上百种编程语言，生成、解释和修复代码。



# DeepSeek简介



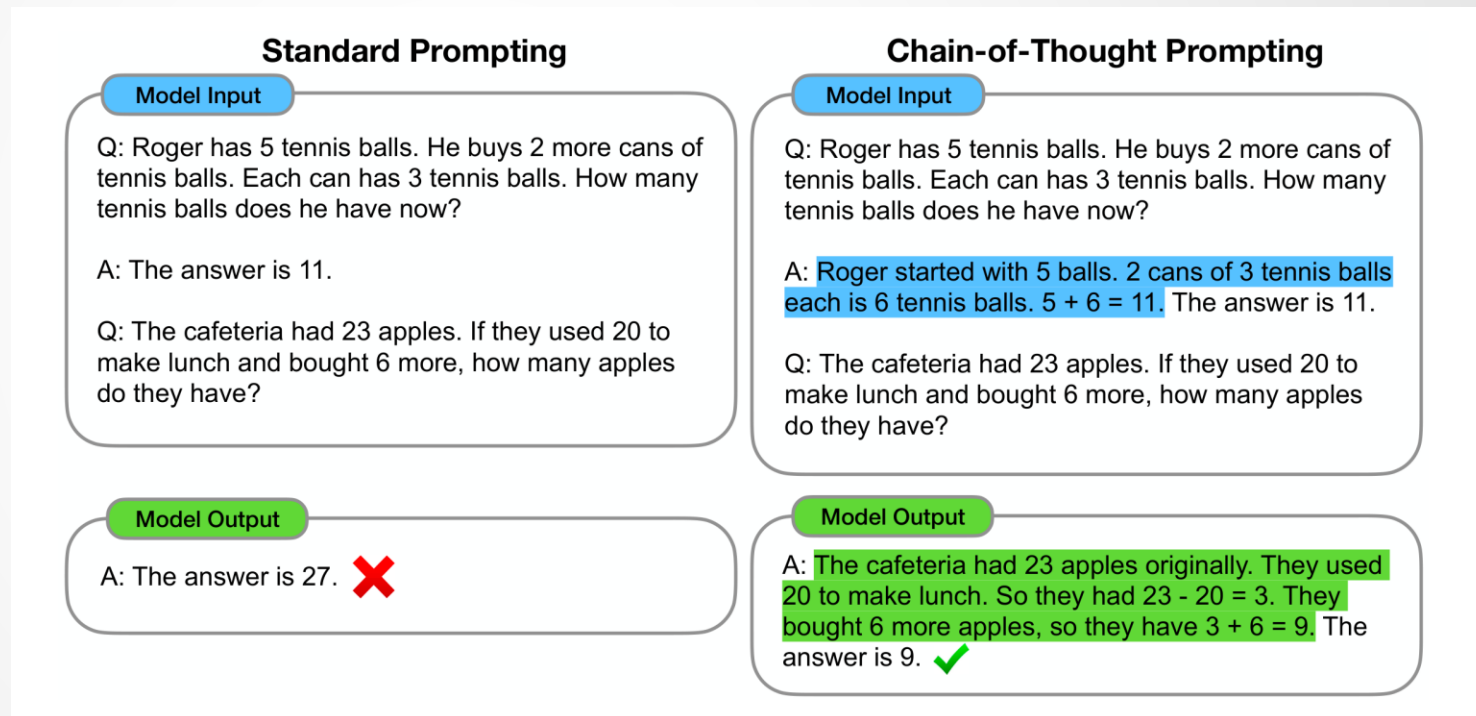
**思维链 (Chain of Thought, CoT)** 2022年Google的论文《Chain-of-Thought Prompting Elicits Reasoning in Large Language Models》中首次提出，以提升大模型复杂问题推理能力

## 核心定义

通过将复杂问题拆解为逐步推理的中间步骤（即思维链），显著提升大模型性能。

## 与传统Prompt对比

- **传统方式**: Input → Output (直接映射)
- **CoT方式**: Input → Reasoning Chain → Output (中间链式推理)



# DeepSeek原理 - 思维链

完整的包含 CoT 的 Prompt 由三部分组成：

- **指令** (Instruction)：明确任务目标与输出格式规范（如JSON/Markdown结构化要求）；
- **逻辑** (Rationale)：包含多跳推理路径、领域知识调用及中间验证步骤；
- **示例** (Exemplars)：提供少样本 (Few-Shot) 的解题范式，每个示例包含完整的问题-推理-答案三元组。

**Direct Reasoning**  
*Input*  
A small sunflower has 3 dozen seeds and a large sunflower has 50% more seeds than a small sunflower. How many sunflower seeds are there altogether?  
*Output*  
54

✗

**Chain-of-Thoughts (Zero-Shot)**  
*Input*  
A small sunflower has 3 dozen seeds and a large sunflower has 50% more seeds than a small sunflower. How many sunflower seeds are there altogether? Let's think step by step.  
*Output*  
A small sunflower has  $3 \times 12 = 36$  seeds. A large sunflower has 50% more seeds than a small sunflower, which is  $36 \times 0.5 = 18$  more seeds. So a large sunflower has  $36 + 18 = 54$  seeds. Altogether, there are  $36 + 54 = 90$  sunflower seeds.  
The answer is 90.

✓

**Chain-of-Thoughts (Few-Shot)**  
*Input*  
Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?  
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been  $21 - 15 = 6$ . The answer is 6.  
Q: A small sunflower has 3 dozen seeds and a large sunflower has 50% more seeds than a small sunflower. How many sunflower seeds are there altogether?  
A:  
*Output*  
A: A small sunflower has  $3 \times 12 = 36$  seeds. A large sunflower has 50% more seeds than a small sunflower, which is  $36 \times 0.5 = 18$  more seeds. So a large sunflower has  $36 + 18 = 54$  seeds. Altogether, there are  $36 + 54 = 90$  sunflower seeds.  
The answer is 90.

✓

Exemplars

Question

Instruction

Rationale

Answer





## CoT训练四阶段

### 阶段一：强化学习验证

- 模型：DeepSeek-V3（MoE架构，数理能力强化）
- 目标：通过纯强化学习产出初始模型 **DeepSeek-R1-Zero**

### 阶段二：数据合成模型

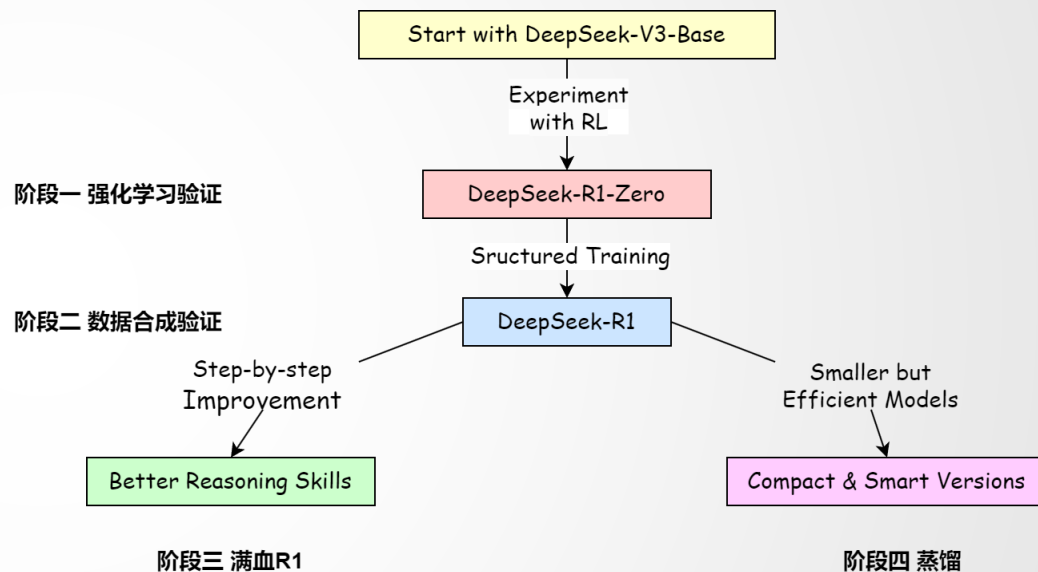
- 输入：DeepSeek-R1-Zero生成数据
- 输出：训练得到 **DeepSeek-V3-checkpoint**
- 数据成果：生成 **600k推理数据集**

### 阶段三：DeepSeek-R1训练

- 数据混合：600k推理数据 + 200k非推理数据
- 方法：全参数微调DeepSeek-V3，然后进行强化学习
- 输出：最终模型 **DeepSeek-R1**

### 阶段四：蒸馏实验

- 对象：开源模型（Qwen、Llama等）
- 方法：相同混合数据集全参数微调
- 输出：**DeepSeek-R1-Distill-（Qwen/Llama）-(\*B)**



# 使用在线的Deepseek模型服务



## 调用DeepSeek的DeepSeek-R1模型

```
1. chat = lazyllm.OnlineChatModule('deepseek-reasoner', source='deepseek')
```

## 调用商汤的DeepSeek-R1模型

```
1. chat = lazyllm.OnlineChatModule('DeepSeek-R1', source='sensenova')
```

## 调用阿里的DeepSeek-R1模型

```
1. chat = lazyllm.OnlineChatModule('deepseek-r1', source='qwen')
```

注意：在使用线上模型前，别忘了设置对应供应商的API-KEY



# 目录

-  1. 上节回顾
-  2. Deepseek简介
-  3. Deepseek的部署和蒸馏
-  4. Deepseek赋能RAG



## 本地部署

```
1. chat = lazyllm.TrainableModule('DeepSeek-R1').\
2.     deploy_method((lazyllm.deploy.Vllm,{
3.         'tensor-parallel-size': 8,
4.         'pipeline-parallel-size': 2,
5.         'max-num-batched-tokens': 131072,
6.         'launcher': launchers.remote(nnode=2, ngpus=8, sync=False)
7.     })))
```

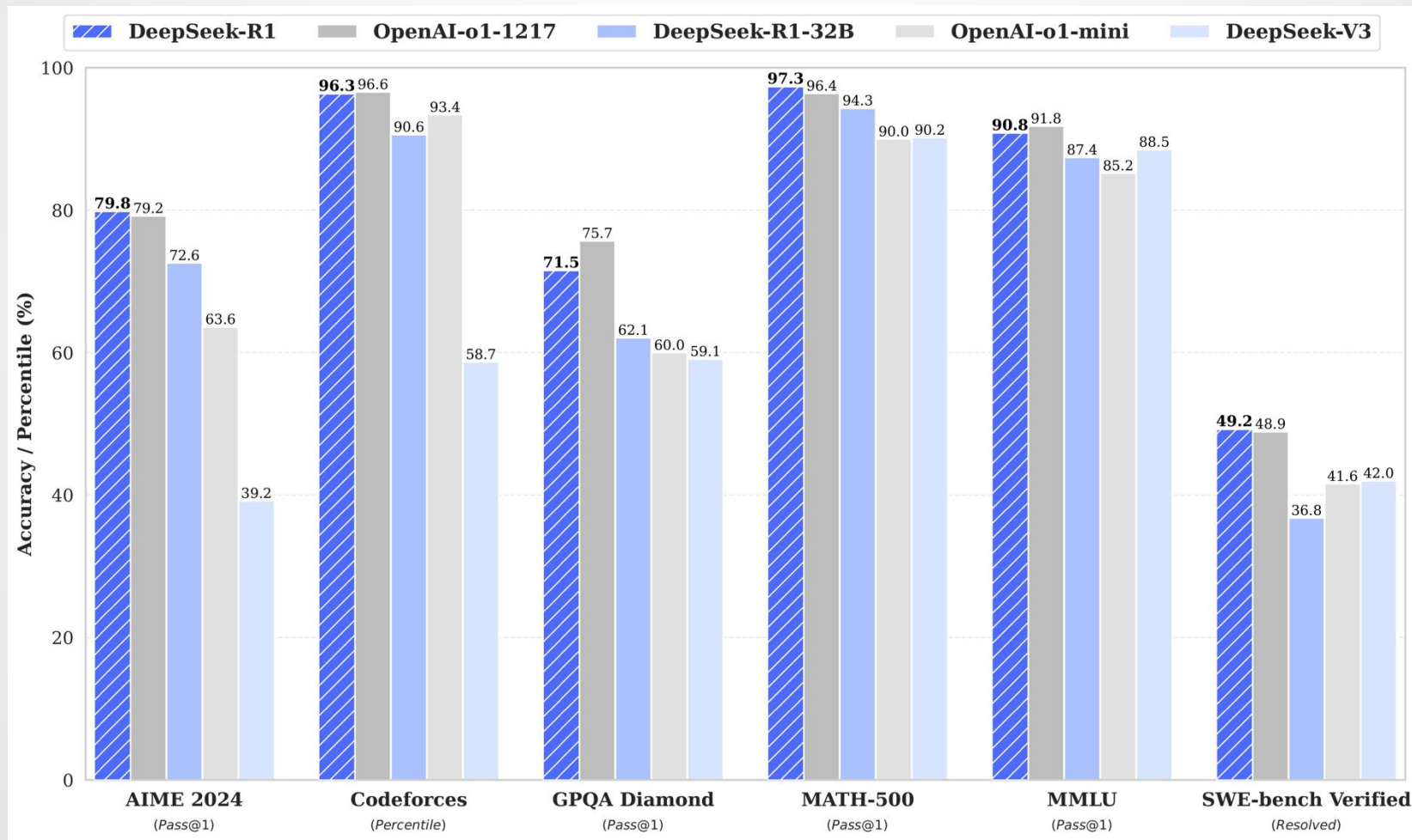
- **tensor-parallel-size**: 设为8表示将模型沿着数据流向方向切成8个小模型，用于分布式推理。
- **pipeline-parallel-size**: 设为2表示将模型垂直于数据流向方向切为2段，结合张量并行共得到16个小模型（每个在一张卡上）
- **launcher (启动任务的平台及算力配置)** :
  - **launchers.remote** :支持的remote算力平台有：SenseCore和Slurm等。
  - **nnode (节点数量)** : 2, 表示需要2个节点。
  - **ngpus (每个节点的计算卡数量)** : 8, 表示每个节点有8张计算卡。
  - **sync (同步设置)** : False, 表示部署命令后不等待任务结束，可继续执行其他任务，如部署WebModule交互界面。
- **max-num-batched-tokens (一个批次的最大token量)** : 数值越大，性能越好，但占用计算卡显存资源越多。



# DeepSeek R1的效果展示 – 优势



DeepSeek-R1模型一个明显的优势之一是数学推理能力



# Deepseek R1的效果展示 – 劣势

## 模型评测

上期阅读理解任务

| 模型                   | 精确匹配率 | 语义相似度 | 原文包含度 |
|----------------------|-------|-------|-------|
| Internlm2-Chat-7B    | 2.10  | 74.51 | 5.19  |
| DeepSeek-V3          | 5.29  | 74.85 | 15.17 |
| DeepSeek-R1          | 2.3   | 69.62 | 7.78  |
| Internlm2-Chat-7B训练后 | 39.72 | 86.19 | 94.91 |

## 例子展示

**context**：...等模式进行游戏。若游戏中离，则4分钟内不得进行配对(每次中离+4分钟)。开放...

**question**：若游戏中离，则多少分钟内不得进行配对？

**Prompt模板**：f"请用下面的文段的原文来回答问题\n\n\n### 已知文段：{context}\n\n\n### 问题：{question}\n"

|    | Internlm2-Chat-7B训练后 | DeepSeek-R1   |
|----|----------------------|---|
| 回答 | 4分钟                  | 根据提供的文段原文: "**"若游戏中离，则4分钟内不得进行配对(每次中离+4分钟)。"** 因此，若游戏中离，**4分钟**内不得进行配对。 |



# Deepseek R1的蒸馏 – 简介

## 背景痛点

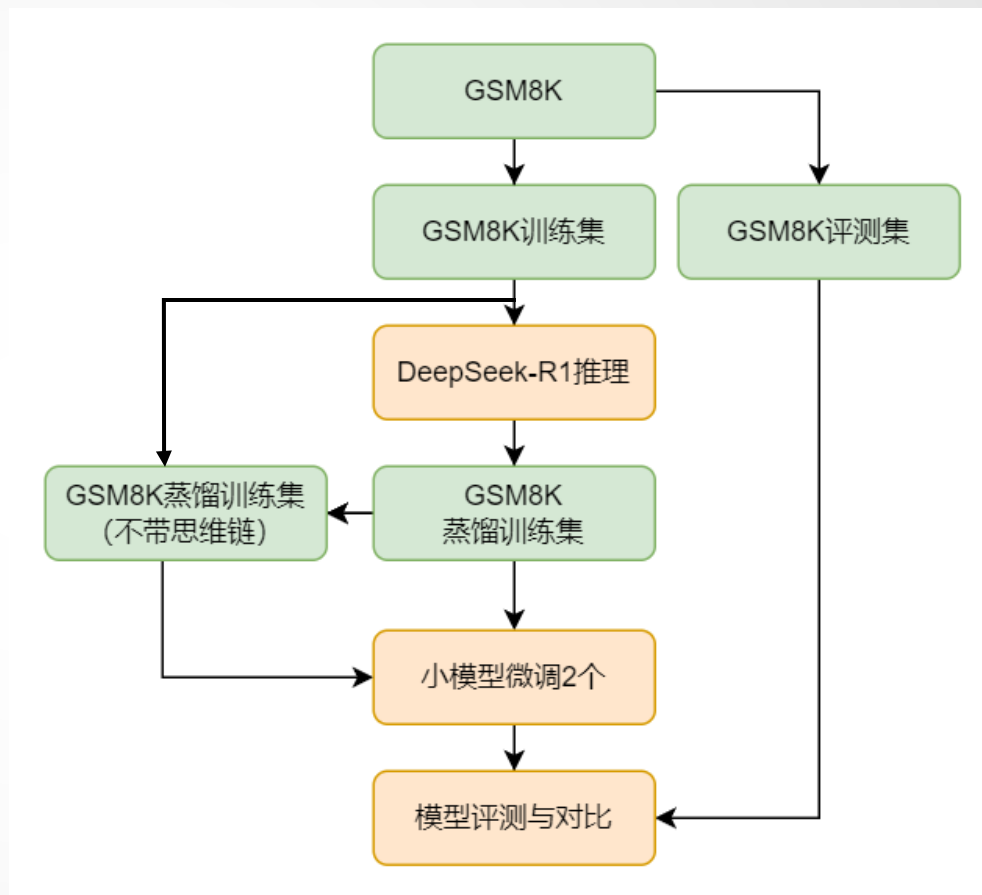
- 小模型在阅读理解内容提取上表现不错，但思维能力存在不足
- 目标：将R1的数学推理强项，通过蒸馏迁移至轻量模型

## 蒸馏原理

用大模型（老师）生成的带**思维链（CoT）**的问答对，来监督微调小模型（学生）

## 方案设计

- **生成CoT数据**：用GSM8K题目+R1推理→{Q: 问题, A:带思维链的答案}
- **带CoT数据组**：完整R1思维链的微调
- **无CoT数据组**：剔除思维链仅保留答案的微调
- **评测对象**：原始小模型 / 微调后双版本 / DeepSeek-R1





# Deepseek R1的蒸馏 – 数据集准备



**GSM8K** (Grade School Math 8K) , 该数据集由 OpenAI 团队构建的数据集

## 数据集概览

- **用途**: 数学问题求解基准
- **规模**: 8,000+小学数学题 (训练集7,473题 / 测试集1,319题)
- **结构**: {question, answer}对, 答案含详细步骤与最终数值

## 核心特点

- **问题类型**: 四则运算+应用题
- **答案格式**: 自然语言推导 + `\n#### ans` 结尾 (明确的数值答案)
- **评估优势**: 通过####后固定数值快速判断模型推理正确性

```
{  
  "question": "James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How  
many total meters does he run a week?",  
  "answer": "He sprints  $3*3=9$  times\nSo he runs  $9*60=540$  meters\n####  
540"  
}
```





# Deepseek R1的蒸馏 – 数据集准备



```
import os
import json
from modelscope.msdatasets import MsDataset

def build_data_path(file_name): # 生成存储目录
    data_root = os.path.join(os.getcwd(), 'dataset')
    if not os.path.exists(data_root):
        os.makedirs(data_root)
    save_path = os.path.join(data_root, file_name)
    return save_path

def get_dataset():
    # 构建存储路径
    train_path = build_data_path('train_set.json')
    eval_path = build_data_path('eval_set.json')
    # 下载数据集
    ds = MsDataset.load('modelscope/gsm8k', subset_name='main')
    # 重命名字段, 用于微调: question → instruction, answer → output
    ds = ds.rename_column('question', 'instruction').rename_column('answer', 'output')
    # 保存数据
    with open(train_path, 'w') as file:
        json.dump(ds['train'].to_list(), file, ensure_ascii=False, indent=4)
    with open(eval_path, 'w') as file:
        json.dump(ds['test'].to_list(), file, ensure_ascii=False, indent=4)
    return train_path, eval_path
```



# Deepseek R1的蒸馏 – 数据集构造



## 构造目标

- **目标：**构建带有思维链、正确格式和正确答案的问答对，用于蒸馏小模型。

## 构建过程

1. **提取训练集：**提取GSM8K的训练集(7473个题)，仅保留提问；
2. **问答对重建：**将7473个题输入DeepSeek-R1，保存输出作为新回答

### ■ 要点：

#### 1. 答案保存标准

- **格式要求：**答案必须包含标准答案，格式为`\\boxed{true_answer}`。
- **思维链要求：**答案中需包含思维链，筛选标志为`</think>`。

#### 2. 保存策略

- 若不满足保存条件，记录问题，完成一轮推理后重新推理，最多重复15次。

```
{  
  "instruction": "Mel is three years younger than Katherine. When Katherine is two dozen years old, how old will Mel be in years?",  
  "output": "<think>\nOkay, let 's see. The problem says Mel is three years younger than Katherine. So, if I figure out Katherine' s age first, then I can subtract three to get Mel 's age. \n\nThe question is asking, when ...Let me do the math here: 24 minus 3 equals 21. So that should be Mel's age when Katherine is 24. Let me double-check. If Mel is always three years younger, then no matter when, the difference stays the same. So when Katherine is 24, subtracting 3 gives 21. Yeah, that makes sense. I think that's the answer.\n</think>\n\nWhen Katherine is two dozen (24) years old, Mel, being three years younger, will be:\n\n\\boxed{21}",  
  "input": ""  
}
```



# Deepseek R1的蒸馏 – 数据集构造



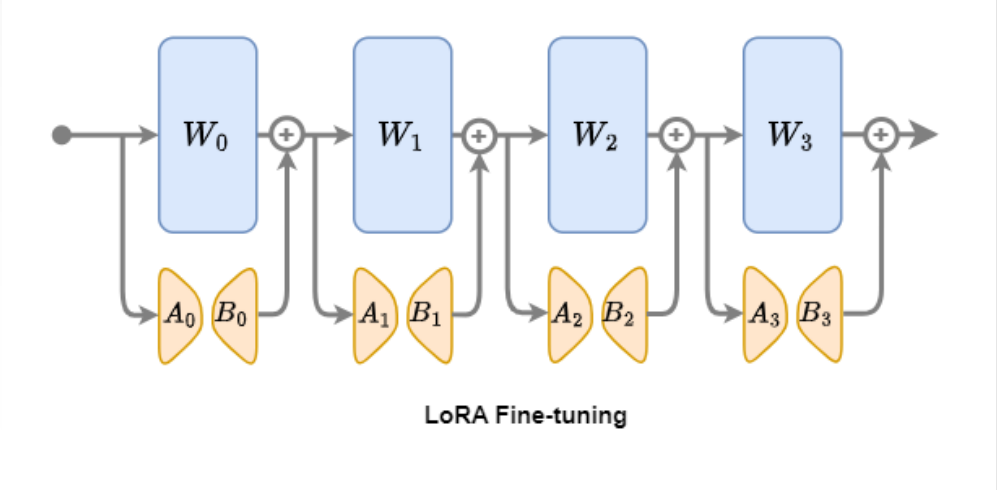
```
1. def distill_dataset(data_path, model=None):
2.     inputs = load_data(data_path)                                # 加载预处理后的训练集
3.     with warp(_concurrent=1) as wp:                             # 并发调用 DeepSeek-R1 模型
4.         wp.func = model
5.         res_list = []
6.         try_n = 0
7.         while inputs:
8.             print(">>>" * 12, f"{try_n+1} times left: ", len(inputs))
9.             queries = [item['instruction'] for item in inputs]    # 提取问题
10.            results = wp(queries)                                # 触发并发推理
11.            valid_data, inputs = filter(inputs, results)          # 筛选符合标准的答案
12.            res_list.extend(valid_data)
13.            try_n += 1
14.            if try_n == 15:                                       # 重试上限: 15 次
15.                break
16.            save_res(res_list, build_data_path('distilled_train_data.json'))
17.
18. def filter(inputs, results):
19.     valid, retry = [], []
20.     for i, item in enumerate(inputs):
21.         true_v = item['output'].split('\n####')[-1].strip()    # 答案正确+格式正确+包含思维链
22.         if f'\boxed{{{true_v}}}' in results[i] and '</think>' in results[i]:
23.             valid.append({'instruction': item['instruction'], 'output': results[i], 'input': ''})
24.         else:
25.             retry.append(item)
26.     return valid, retry
```



## 微调回顾

和上期教程一样采用 **LoRA (Low-Rank Adaptation)** 实现轻量微调：

$$h = Wx + \frac{\alpha}{r}BAx$$



## 参数配置

| 参数 | Model             | Epochs | Batch-size | GPUs     | Gradient Accumulation step | learning_rate | lora_rank |
|----|-------------------|--------|------------|----------|----------------------------|---------------|-----------|
| 说明 | InternLM2-7B-Chat | 2      | 16         | 8 (A800) | 1                          | 1.00E-04      | 8         |



# Deepseek R1的蒸馏 – 微调模型



```
1. # 1. 获取数据
2. train_set_path, eval_set_path = get_dataset()
3. eval_set = load_data(eval_set_path)
4. # 2. 蒸馏数据
5. teacher_model = lazyllm.OnlineChatModule('DeepSeek-R1')
6. sft_data_path = distill_dataset(train_set_path, teacher_model)
7. # 3. 微调模型
8. student_model = lazyllm.TrainableModule('internlm2-chat-7b')\      # 指定使用模型 internlm2-chat-7b
9.     .mode('finetune')\      # 设置模式为微调
10.    .trainset(sft_data_path)\      # 设置上一步蒸馏出的训练集
11.    .finetune_method((finetune.LlamaFactory, {      # 使用的微调引擎: LLaMA-Factory
12.        'learning_rate': 1e-4,      # 学习率
13.        'cutoff_len': 5120,      # 数据最大截断长度
14.        'max_samples': 20000,      # 训练集最大样本数
15.        'val_size': 0.01,      # 检验集大小比例设置
16.        'per_device_train_batch_size': 2,      # 每张卡上的批次大小
17.        'num_train_epochs': 2.0,      # 训练重复的总轮数
18.        'launcher': launchers.remote(nnode=1, ngpus=8)      # 单机8卡远端设备训练
19.    })\
20.    .prompt(dict(system= 'You are a helpful assistant.' , drop_builtin_system=True))\ # 设置系统 Prompt
21.    .deploy_method(deploy.VLLM)      # 指定部署用的引擎为 vLLM
22. student_model.prompt._soa = '<|im_start|>assistant\n\n<think>'      # 为特殊soa标记增加<think>
23. student_model.evalset([item['instruction'] for item in eval_set])      # 设置评测用的推理数据集
24. student_model.update()      # 一键启动: 微调、部署、推理
25. ## 4. 评测模型
26. score = caculate_score(eval_set, student_model.eval_result)      # 评测模型
27. print("All Done. Score is: ", score)
```



# Deepseek R1的蒸馏



File Edit Selection View Go Run Terminal Help

Explorer: DEMO [SSH: SCO-LAZYPLATFORM] > .lazyllm > LazyLLMTutorial > rag > courseware > courseware\_codes > chapter10 > distill\_deepseek\_r1.py

Terminal:

```
LazyLLMTutorial > rag > courseware_codes > chapter10 > distill_deepseek_r1.py
You, 18 hours ago | 1 author (You)
1 import os
2 import re
3 import json
4 import argparse
5
6 import lazyllm
7 from lazyllm import finetune, deploy, launchers, warp
8 You, 2 days ago * chapter10
9 from modelscope.msdatasets import MsDataset
10
11
12 def load_data(data_path):
13     with open(data_path, 'r') as file:
14         dataset = json.load(file)
15     return dataset
16
```

Terminal Output:

```
(lazyllm) 69e3d7be-6e7f-11ef-a667-661a8191c638% ls
chinese_math17290.json dataset infer_true_cp.json online_deepseek.py sft_llm.py
data distill_deepseek_r1.py math_chinese_rag.py save_ckpt
(lazyllm) 69e3d7be-6e7f-11ef-a667-661a8191c638%
```

Terminal Title: zsh - chapter10

Status Bar: SSH: SCO-lazyplatform | You, 2 days ago | Ln 8, Col 1 | Spaces: 4 | UTF-8 | LF | Python



# Deepseek R1的蒸馏 – 模型评测



## 评测标准

- **数值正确性**: 生成答案与标准答案完全匹配;
- **格式合规性**: 答案需以 `\\boxed{...}` 包裹。

## 对比实验设计

- 蒸馏前模型: 未经优化的原始小模型;
- 蒸馏后模型1: 基于完整蒸馏数据 (带思维链) 微调的模型;
- 蒸馏后模型2: 基于不带思维链的蒸馏数据微调的模型;
- 基准模型: DeepSeek-R1 作为性能上限参考。



# Deepseek R1的蒸馏 – 模型评测



```
1. def extract_boxed_content(text):
2.     pattern = r'boxed{((?:[^\}]*|{.*?})*)}'
3.     contents = re.findall(pattern, text)           # 从指定格式中提取答案
4.     return contents
5.
6. def caculate_score(eval_set, infer_set):
7.     assert len(eval_set) == len(infer_set)
8.     score = 0
9.     for index, eval_item in enumerate(eval_set):
10.        output = infer_set[index]                  # 提取一条推理结果（与评测集对应）
11.        if 'boxed{' in output:                      # 判断基本格式是否正确
12.            res = extract_boxed_content(output)      # 抽取推理的结果
13.            res = list(set(res))                     # 去除重复的答案
14.            res = res[0] if len(res) == 1 else res   # 单答案就不提取出，多答案保持list
15.            if type(res) is list                    # 多答案下不计分直接跳过
16.                continue
17.            true_v = eval_item['output'].split('\n#### ')[-1].strip() # 提取准确答案
18.            if true_v == res.strip():                # 判断答案是否完全一致
19.                score += 1                           # 答案完全一致累计一分
20.    return f'{score}/{len(eval_set)}, {round(score/len(eval_set),4)*100}%'
```





# Deepseek R1的蒸馏效果呈现

| 模型               | InternLM2-7B-Chat<br>【原始】 | InternLM2-7B-Chat<br>【微调后-不带思维链】 | InternLM2-7B-Chat<br>【蒸馏后 – 带思维链】 | DeepSeek-R1<br>【教师模型】 |
|------------------|---------------------------|----------------------------------|-----------------------------------|-----------------------|
| 答对题数<br>(共1319题) | 331                       | 839                              | 951                               | 1201                  |
| 准确率              | 25.09%                    | 63.61%                           | 72.10%                            | 91.05%                |

- 1.基础蒸馏增益：无思维链的蒸馏使准确率从25.09%跃升至63.61%，绝对提升达38.5个百分点，证明基础蒸馏是有效的；
- 2.思维链附加值：引入CoT机制后准确率再提升8.5个百分点，验证思维链对知识迁移的强化作用；
- 3.师生差距：学生模型（72.1%）与教师模型（91.05%）存在18.95个百分点的性能差，揭示模型容量对推理能力的关键影响；
- 4.规模效率比：7B蒸馏模型达到671 B教师模型79.2%的准确率水平，以近1/100参数量实现4/5的性能表现！



# 目录

-  1. 上节回顾
-  2. Deepseek简介
-  3. Deepseek的部署和蒸馏
-  4. Deepseek赋能RAG



# 引入CMRC2018训练集增强阅读能力

## 主要步骤

- 混合数据集：上期阅读理解CMRC2018训练集 + 本期蒸馏出的GSM8K训练集
- 微调混合后的数据集
- 评测对比

## 评测结果

| 任务<br>(数据集)                            | 阅读理解信息抽取能力<br>(CMRC2018) |        |        | 数学推理能力<br>(GSM8K) |
|--|--------------------------|--------|--------|-------------------|
| 模型                                     | 精确匹配率                    | 语义相似度  | 原文包含度  | 准确度               |
| Internlm2-Chat-7B                      | 2.10%                    | 74.51% | 5.19%  | 25.09%            |
| DeepSeek-R1                            | 2.3%                     | 69.62% | 7.78%  | 91.05%            |
| Internlm2-Chat-7B<br>训练后 (仅CMRC2018数据) | 39.72%                   | 86.19% | 94.91% | 24.69%            |
| Internlm2-Chat-7B<br>训练后 (仅GSM8K蒸馏数据)  | 2.3%                     | 69.31% | 8.62%  | 72.10%            |
| Internlm2-Chat-7B<br>训练后(混合数据)         | 39.22%                   | 86.22% | 93.71% | 73.24%            |

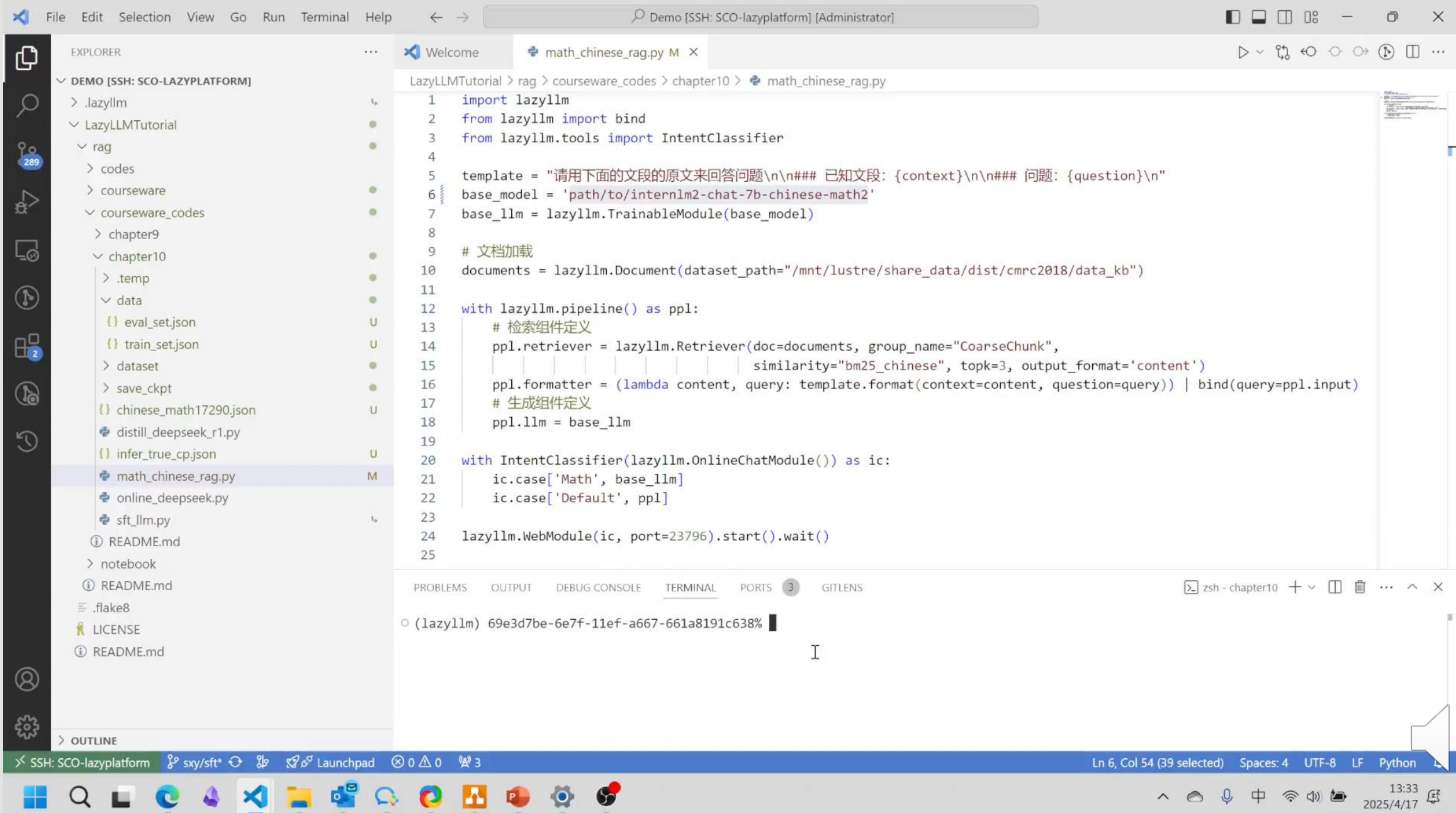


# Deepseek赋能你的RAG



```
1.  template = "请用下面的文段的原文来回答问题\n\n### 已知文段: {context}\n\n### 问题: {question}\n"
2.  base_model = 'path/to/internlm2-chat-7b-chinese-math2'
3.  base_llm = lazyllm.TrainableModule(base_model)
4.
5.  # 文档加载
6.  documents = lazyllm.Document(dataset_path="path/to/cmrc2018/data_kb")
7.
8.  with lazyllm.pipeline() as ppl:
9.      # 检索组件定义
10.     ppl.retriever = lazyllm.Retriever(doc=documents,
11.         group_name="CoarseChunk", similarity="bm25_chinese", topk=3)
12.     ppl.formatter = (lambda nodes, query: template.format(
13.         context="".join([node.get_content() for node in nodes]), question=query)) | bind(query=ppl.input)
14.     # 生成组件定义
15.     ppl.llm = base_llm
16.
17.  with IntentClassifier(lazyllm.OnlineChatModule()) as ic:
18.     ic.case['Math', base_llm]
19.     ic.case['Default', ppl]
20.
21.  lazyllm.WebModule(ic, port=23496).start().wait()
```





File Edit Selection View Go Run Terminal Help

Demo [SSH: SCO-lazyplatform] [Administrator]

EXPLORER

- DEMO [SSH: SCO-LAZYPLATFORM]
  - .lazyllm
  - LazyLLMTutorial
    - rag
      - codes
      - courseware
        - chapter9
        - chapter10
          - .temp
          - data
            - eval\_set.json
            - train\_set.json
          - dataset
          - saveckpt
          - chinese\_math17290.json
          - distill\_deepseek\_r1.py
          - infer\_true\_cp.json
          - math\_chinese\_rag.py
          - online\_deepseek.py
          - sft\_llm.py
        - README.md
        - notebook
        - README.md
        - .flake8
        - LICENSE
        - README.md

math\_chinese\_rag.py

```
1 import lazyllm
2 from lazyllm import bind
3 from lazyllm.tools import IntentClassifier
4
5 template = "请用下面的文段的原文来回答问题\n\n### 已知文段: {context}\n\n### 问题: {question}\n"
6 base_model = 'path/to/internlm2-chat-7b-chinese-math2'
7 base_llm = lazyllm.TrainableModule(base_model)
8
9 # 文档加载
10 documents = lazyllm.Document(dataset_path="/mnt/lustre/share_data/dist/cmrc2018/data_kb")
11
12 with lazyllm.pipeline() as ppl:
13     # 检索组件定义
14     ppl.retriever = lazyllm.Retriever(doc=documents, group_name="CoarseChunk",
15                                     similarity="bm25_chinese", topk=3, output_format='content')
16     ppl.formatter = (lambda content, query: template.format(context=content, question=query)) | bind(query=ppl.input)
17     # 生成组件定义
18     ppl.llm = base_llm
19
20 with IntentClassifier(lazyllm.OnlineChatModule()) as ic:
21     ic.case['Math', base_llm]
22     ic.case['Default', ppl]
23
24 lazyllm.WebModule(ic, port=23796).start().wait()
25
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

zsh - chapter10

(lazyllm) 69e3d7be-6e7f-11ef-a667-661a8191c638%

Ln 6, Col 54 (39 selected) Spaces: 4 UTF-8 LF Python

13:33 2025/4/17

# Q&A

1. 蒸馏就是让大模型生成数据，然后给小模型做监督微调，那我用gpt生成数据，然后训我的模型，是不是可以说我的模型是gpt蒸馏出来的？
2. 之后如果我想自己做一个带思维链的模型，还需要再复现一遍Deepseek R1的过程么？





**感谢聆听**  
**Thanks for Listening**

