LMCC 第二轮认证上机环境配置与真题选讲

西安铁一中学 徐保毅

QQ 465039728

一、考生上机考试指导手册（2025年版）

考试流程总览

1. 开始考试 获取考试文件（通过指定命令）

```
ma-cli obs-copy obs://obs-lmcc-2025/LMCC-Common/ ~/work/
```

（以上机考试指导为准）

2. 作答（在VSCode中编辑文件）

注意：修改 submission.py

3. 提交答案（通过指定命令）

```
ma-cli obs-copy ./LMCC-Common/题目的文件夹名字 obs://obs-lmcc-2025/考生账号/
```

（以上机考试指导为准）

考生考试指导手册.docx

注意：考试机器环境已经配置好，能调试运行 evaluate.py 就行

考试必读.md

- 1. 试题的 evaluate.py 都是不可修改的，评测时会使用原版文件
- 2. 环境要求：(考场已经提供)

  2025必须使用 `PyTorch-2.6.0` 环境，否则可能缺少必要的依赖包或版本不匹配。
- **3. Vscode 中修改 submission.py**
- **4. Vscode 中运行 evlatuate.py，测试 submission.py 的得分**

  4.1 在 Vscode 中打开终端

  4.2 切换到 T1/T2 工作目录

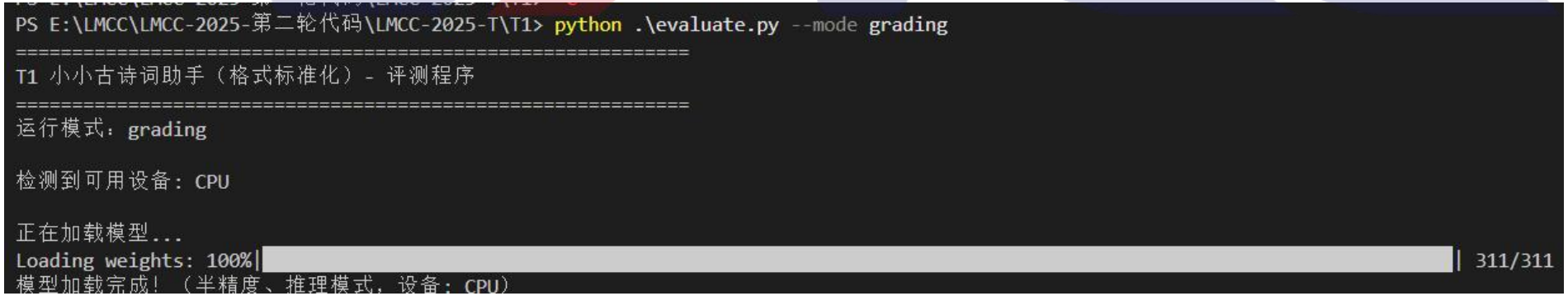
  4.3 运行 evaluate.py

```
python .\evaluate.py
python .\evaluate.py --mode demo
python .\evaluate.py --mode grading
```

在 VScode 中运行 evaluate.py

python .\evaluate.py --mode demo

python .\evaluate.py --mode grading



二、LMCC第二轮本地环境配置

考前练习：需要自己搭建本机环境进行模拟测试

- **1. 基本环境配置**

- 1.1 下载并安装python（版本3.9以上）
- 1.2 pytorch 安装
- 1.3 transformers 安装
- 1.4 Vscode 安装 (或者 Pycharm)

- **2. 下载模型权重**

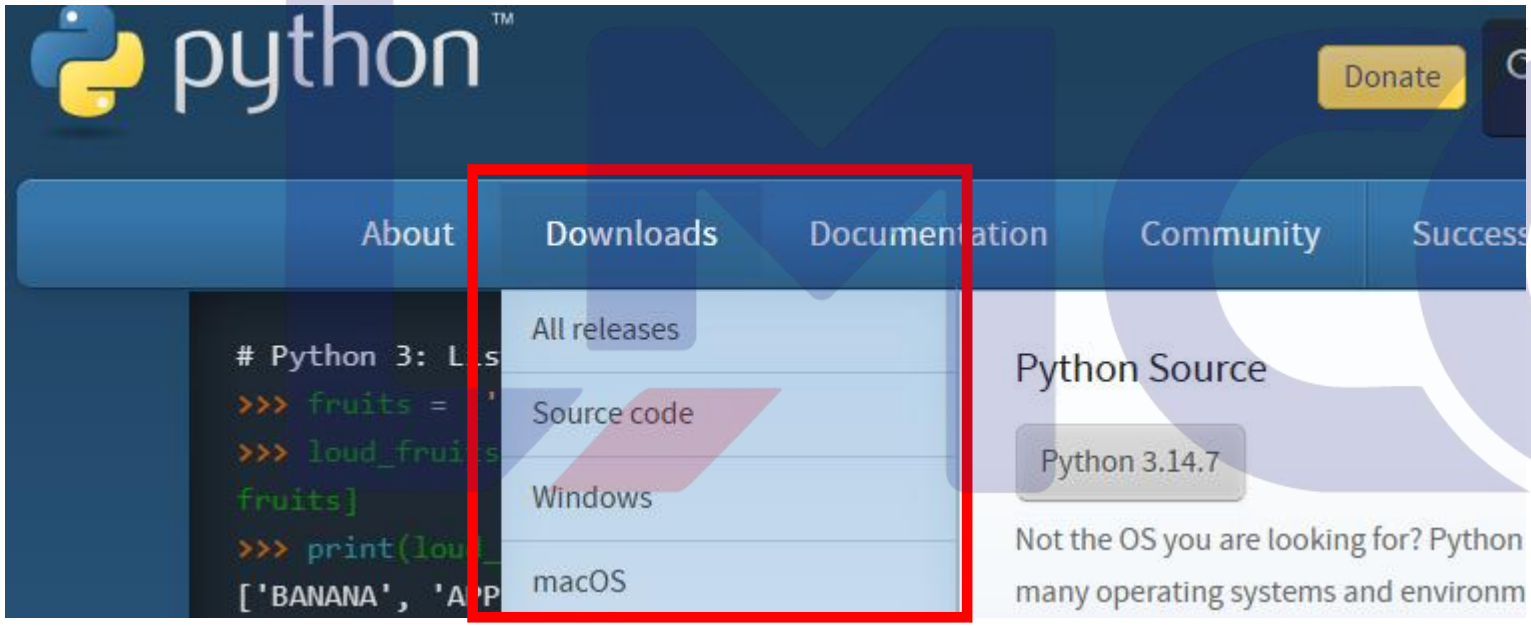
- 2.1 下载安装 Git
- 2.2 使用 Git clone .../Qwen3-0.6B、.../Qwen3-Embedding-0.6B

- **3. 解题 并 测试**

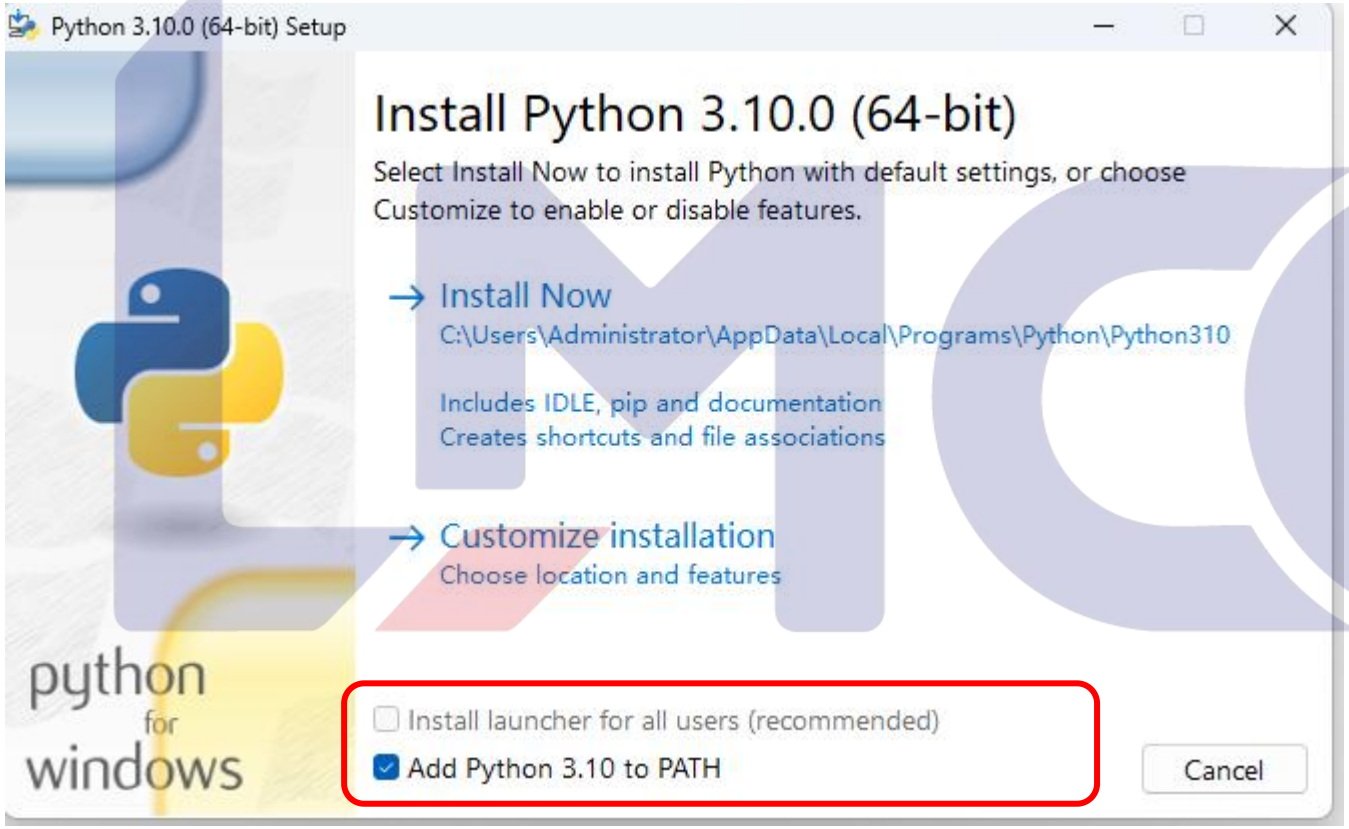
- 3.1 修改 submission.py （解题）
- 3.2 修改 并 运行 evaluate.py （测试）（考试时不能修改 evaluate.py ）

1.1 下载并安装python（版本3.9以上）

- 官方网址：https://www.python.org/



安装时须要选择添加环境变量



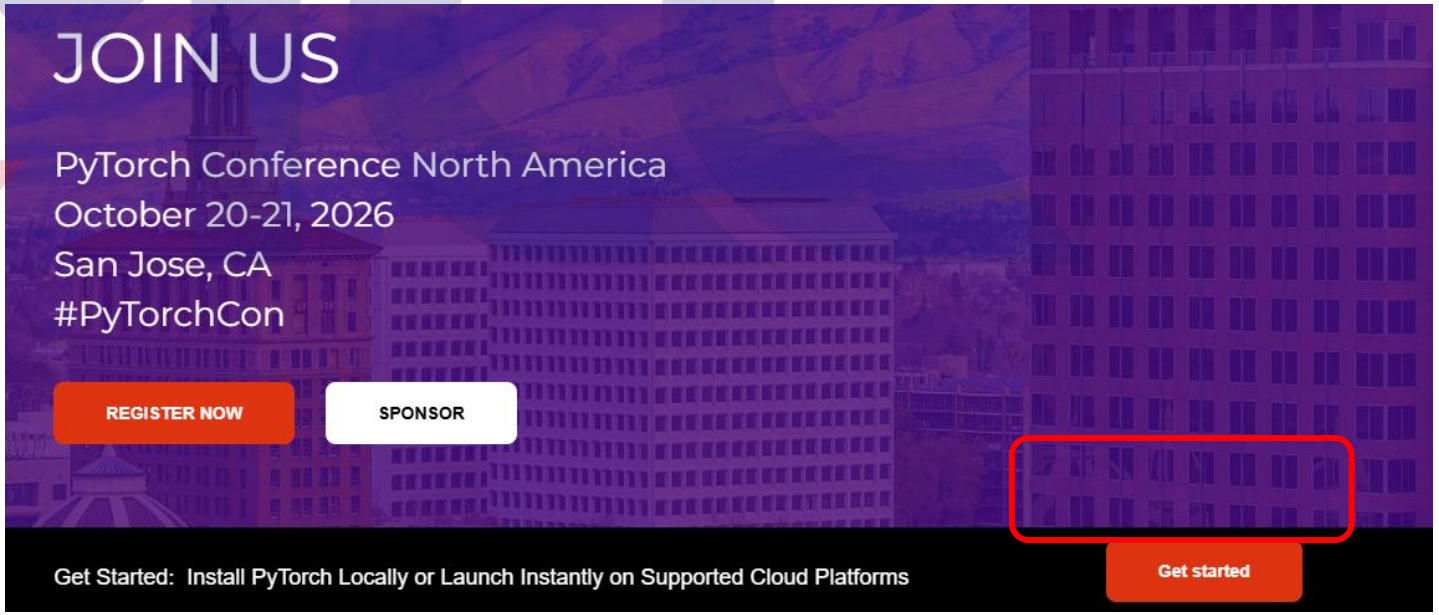
1.2 pytorch 安装

4. **环境要求**：运行评测程序前，请确保已激活正确的 conda 环境：

```
conda activate PyTorch-2.6.0
```

必须使用 `PyTorch-2.6.0` 环境，否则可能缺少必要的依赖包或版本不匹配。

官方网站：
https://pytorch.org/



2025考题必须使用 `PyTorch-2.6.0` 环境

Start Locally

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

NOTE: Latest Stable PyTorch requires Python 3.10 or later.

PyTorch Build	Stable (2.13.0)			Preview (Nightly)	
Your OS	Linux		Mac	Windows	
Package	Pip		LibTorch	Source	
Language	Python			C++ / Java	
Compute Platform	CUDA 12.6	CUDA 13.0	CUDA 13.2	~~ROCm 7.2~~	CPU
Run this Command:	pip3 install torch torchvision --index-url https://download.pytorch.org/whl/cu126				

v2.6.0

Wheel

OSX

```
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0
```

LINUX AND WINDOWS

```
# ROCM 6.1 (Linux only)
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
# ROCM 6.2.4 (Linux only)
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
# CUDA 11.8
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
# CUDA 12.4
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
# CUDA 12.6
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
# CPU only
pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download
```

GPU加速

没有独立显卡

GPU 加速与无独立显卡

- # CUDA 12.6 有独立显卡
- pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download.pytorch.org/whl/cu126

- # CPU only 无独立显卡
- pip install torch==2.6.0 torchvision==0.21.0 torchaudio==2.6.0 --index-url https://download.pytorch.org/whl/cpu

➢ 在 cmd/powershell 中运行 pip 安装命令

1.3 transformers 安装

在已配置好 PyTorch 环境的前提下，直接运行：

➢直接安装

- pip install transformers

➢或国内镜像源加速

- pip install transformers -i https://pypi.tuna.tsinghua.edu.cn/simple

1.4 VScode 下载安装 (或者 Pycharm)

- VScode 是一款通用编辑器

- 访问 VS Code 官方网站：https://code.visualstudio.com/
- 双击运行下载好的 .exe 安装程序。
- 添加到 PATH（推荐）

VScode 在终端中运行 python 程序

- 在代码编辑器任意空白处点击右键，选择 在终端中运行 Python 文件 (Run Python File in Terminal)。

➢python .\evaluate.py

2. 下载模型权重

- 2025 需要下载模型权重 Qwen3-0.6B/Qwen3-Embedding-0.6B
- 往外网址：https://huggingface.co/
- 国内网址：https://www.modelscope.cn/home

推荐使用 Git 克隆下载

2.1 下载安装 Git

- 官网 https://git-scm.com/
- 下载后，直接安装
- Git --version 检查是否安装成功

2.2 使用 Git clone ...

- 使用国内魔塔社区
- https://www.modelscope.cn/models/Qwen/Qwen3-0.6B
- https://www.modelscope.cn/models/Qwen/Qwen3-Embedding-0.6B

➢注意下载位置

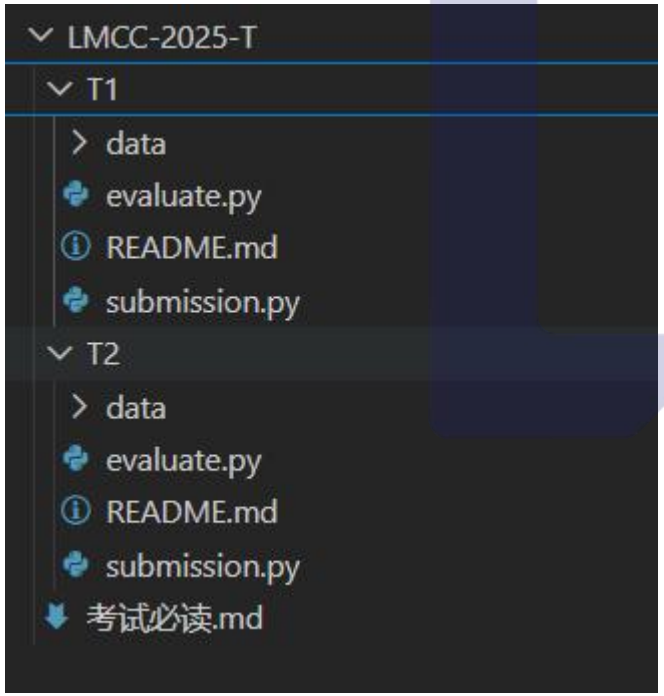
➢`git clone https://www.modelscope.cn/Qwen/Qwen3-0.6B.git`

➢`git clone https://www.modelscope.cn/Qwen/Qwen3-Embedding-0.6B.git`

➢下载好的 Qwen3-0.6B \Qwen3-Embedding-0.6B 放在测试题目目录下

3. 解题 并 测试

- 3.1 修改 submission.py （解题）




```
# ============================================================
# Prompt 定义（考生实现）
# ============================================================
def build_system_prompt() -> str:
    """
    考生实现：定义 system prompt

    要求：
    1. 引导模型理解题目格式标准化任务，将各种格式的题目转换成统一的JSON格式
    2. 说明输入是各种格式的古诗词选择题（格式可能千奇百怪），输出是标准JSON
    3. 引导模型从各种格式中正确识别出：
```

- 3.2 修改 并 运行 evaluate.py （测试）

- （考试时不能修改 evaluate.py ）
- 部分题目可能根据需要修改 import
- 修改模型路径，使用相对路径或绝对路径
- 运行测试

```
model_path = os.environ.get("MODEL_NAME", "/Users/yuning/LMCC/RealRound2/LMCC-2025-T/Qwen/Qwen3-0.6B")
```

```
python evaluate.py
python evaluate.py --mode demo
python evaluate.py --mode grading
```

运行方式：演示模式/评分模式

注意：--mode 参数 可能是 可选项 或 必选项

演示模式（详细输出）：

```
python evaluate.py --mode demo
```

评分模式（简洁输出）：

```
python evaluate.py --mode grading
```

三、第二轮题目选讲

T1 小小古诗词助手（格式标准化）

故事背景

你来到小学帮助老师整理古诗词题库。老师收集了很多古诗词填空题，但是这些题目的格式五花八门：

题目格式的混乱现状：

- 有的用标准格式：`选项：A. xxx B. xxx C. xxx D. xxx`
- 有的用括号：`(A)xxx (B)xxx (C)xxx (D)xxx`
- 有的用方括号：`[A]xxx [B]xxx [C]xxx [D]xxx`
- 有的用数字：`①xxx ②xxx ③xxx ④xxx`，然后告诉你"①对应A，②对应B..."
- 有的会在题目里加上各种提示、注意事项、友情提示等干扰信息
- 有的会把答案描述得很啰嗦："选项A是xxx，选项B是xxx..."
- **最重要的是**：题目中已经标注了正确答案（但标注方式也五花八门）

老师希望你能编写一个**题目格式标准化工具**，将这些五花八门的题目统一转换成标准的JSON格式，方便后续的题库管理和系统使用。这个工具基于 **PyTorch** 和 **Transformers** 库，可以自动识别各种格式的题目并输出标准格式。

任务目标

你需要在 `submission.py` 中完成**两个核心函数**，让格式标准化工具具备以下能力：

1. 定义转换规则：`build_system_prompt() -> str`

编写一个 system prompt，引导模型理解题目格式标准化任务，将各种格式的题目转换成统一的JSON格式。

T1 README.pdf

要点1：考题规则与评测机制拆解

- **修改权限与环境约束**：
  - 只能修改 `submission.py` 中的 `build_system_prompt()` 与 `build_generation_parameters()` 两个函数。
- **严苛的判分机制**：
  - 单题 5 分，共 10 题，满分 50 分。
  - **判定标准是完全匹配**：模型输出的 JSON 字典与期望 JSON 字典必须 `predicted_json == expected_json`（包含 `question`、`options`、`answer` 三个字段）。格式错一位即得 0 分。
- **硬性指标限制**：
  - `System Prompt` 最大 Token 数不得超过 **2048 tokens**。
  - 10 道题总推理时间不得超过 **10 分钟**。
- **隐藏数据与防过拟合（不要打表）**：
  - 强调最终评测使用的是无考生可见的 `.final.jsonl` 测试集。
  - **禁忌**：切勿将自测数据（`test_data.jsonl`）中的题目文本或特定答案硬编码（Hardcode）进 Prompt 中，否则最终评测必失败。

要点2：学会使用命令行测试程序

- 由于 Windows 和 Linux 命令行有细微差异， 以 Vscode 下powershell 为例

在运行 `.py` 文件前，必须确保终端的当前工作路径“定位”到了代码所在的文件夹。

命令	用途	含义与说明
`pwd`	**显示当前路径**	*Print Working Directory*，查看当前终端处于哪个文件夹下。
`ls` 或 `dir`	**列出文件夹内容**	查看当前目录下的所有文件和文件夹，确认你的 `.py` 文件是否在这里。
`cd <文件夹名>`	**进入指定文件夹**	*Change Directory*，例如 `cd src` 进入 `src` 文件夹。
`cd ..`	**返回上一级目录**	向上退回一层文件夹。
`cd ~`	**回到用户主目录**	快速跳转到当前系统用户的家目录（如 `C:\Users\username`）。

Powershell下常用命令-Python.pdf

- `cls` 或 `clear`：清理终端界面（刷屏太多时快速清屏）。
- `Ctrl + C`：强制中断正在运行的 Python 程序（比如死循环或长时间未响应的任务）。
- `↑` / `↓` **方向键**：快速翻阅之前输入过的历史命令，无需重复输入。
- `Tab` ：文件名/文件夹名补全。

要点3： Prompt 提示词书写要点

- 1. 清晰角色赋予，能够让模型回复更专业和有针对性。
- 2. 任务描述要清晰：明确目标、列举限制条件、指定输出格式结构要求。
- 3. 选择合适的示例（让模型根据示例学会执行新任务）
- 4. 使用 demo 模式，观察模型思考过程，改进提示词。
- 5. “提示末端”效应（越靠后记忆清晰），提示词并不是越多越好，示例也是。

T2

T2 小小古诗词助手（检索作答）

故事背景

你来到小学帮助小朋友学习古诗词。孩子们每天都会来问你各种古诗词选择题，但他们经常记不全诗句。

好在，学校图书馆提供了一个包含**100首经典唐诗**的知识库。你决定开发一个**基于检索增强生成（RAG）**的古诗词助手：

1. 根据题目从知识库中检索相关的诗词
2. 利用检索到的诗词知识来回答问题

这样，即使是小模型，也能通过查阅知识库来准确回答古诗词问题！

什么是 RAG？
RAG（Retrieval-Augmented Generation）= 检索 + 生成：

```
问题 → 检索相关知识 → 将知识作为上下文 → 模型基于上下文生成答案
```

优势：

- 模型可以访问外部知识，不依赖预训练时的记忆
- 知识可以随时更新，无需重新训练模型
- 答案有依据，可追溯到具体的知识来源

任务目标

你需要在 `submission.py` 中完成**四个核心函数**：

1. 检索函数：`retrieve_relevant_poems(problem, knowledge_base, top_k) -> List[Dict]`

从唐诗知识库中检索与题目最相关的诗词。

输入：

- `problem`：题目文本（字符串）
- `knowledge_base`：唐诗知识库（列表），每首诗是一个字典，包含：
  - `author`：作者
  - `title`：诗名
  - `paragraphs`：诗句列表（每行诗包含两句，用逗号分隔）
  - `prologue`：解释（可能为空）
- `top_k`：返回最相关的前 k 首诗（默认 3）

输出：

- 包含最相关的 `top_k` 首诗的列表

实现提示：

1. 从题目中提取关键诗句（去掉"_"和选项部分）

T2 README.pdf

LMCC-2025+第二轮试题解析+-+青少年组.pdf

要点1：读懂题目要完成什么任务

什么是 RAG?

RAG（Retrieval-Augmented Generation）= 检索 + 生成:

```
1 问题 → 检索相关知识 → 将知识作为上下文 → 模型基于上下文生成答案
```

1. 检索函数: `retrieve_relevant_poems(problem, knowledge_base, top_k) -> List[Dict]`

2. 用户消息组装: `build_user_message(problem, retrieved_poems) -> str`

3. System Prompt 定义: `build_system_prompt() -> str`

编写 system prompt，引导模型使用参考知识来回答问题。

要点2：相关 Python 语法基础必须掌握

- 请记住一个核心原则：**编程的本质是逻辑，而不是背诵函数库。**
- 在 Python 中，几乎所有的高级函数，都可以用最基础的 for 循环、if 判断和基础数据类型（列表、字典、字符串）手写出来。
- 只要结果对，评测脚本（evaluate.py）根本不在乎你代码写得优不优雅。
- 只要你能用最基础的 for、if、+、== 把逻辑跑通，把正确的诗词找出来拼接到 Prompt 里，大模型就能给出正确答案，你就能拿满分。不要有语法焦虑，暴力破解也是一种智慧！

1. 忘记了怎么用正则表达式（`re.sub`）去标点？

- **高级写法：** `re.sub(r'[，。_]', '', text)`
- **基础自救写法：** 遍历字符串，遇到不要的字符就跳过，要的字符就拼起来。

```python
text = "床前明月光，_____。"
clean_text = ""
# 自己定义一个包含讨厌字符的列表
bad_chars = ['，', '。', '、', '？', '！', '_', 'A', 'B', 'C', 'D', '.']

for char in text:
    if char not in bad_chars:
        clean_text += char  # 只要不是标点，就拼接到新字符串里

print(clean_text) # 结果一样是 "床前明月光"
```

2. 忘记了怎么统计字符出现次数（`text.count('_')`）？

- **高级写法：** `if text.count('_') >= 3:`
- **基础自救写法：** 设一个计数器，用 `for` 循环数。

Python

```
text = "床前明月光,_____."
underscore_count = 0

for char in text:
    if char == '_':
        underscore_count += 1

if underscore_count >= 3:
    print("这是核心行")
```

3. 忘记了列表推导式怎么写 3-gram?

- **高级写法：** `[text[i:i+3] for i in range(len(text)-2)]`
- **基础自救写法：** 老老实实建一个空列表，用 `for` 循环 `append`。

Python

```
text = "床前明月光"
trigrams = []

for i in range(len(text) - 2):
    piece = text[i : i+3] # 切片截取3个字
    trigrams.append(piece)
```

4. 忘记了怎么用 `sorted` 对字典列表按分数排序？

- **高级写法：** `sorted(candidates, key=lambda x: x["score"], reverse=True)`
- **基础自救写法：** 既然只需要前 K 个（比如前 3 个），那就写个循环，每次找最高分，找 3 次。

Python

```python
# 假设 candidates 是 [{"score": 2, "poem": "A"}, {"score": 5, "poem": "B"}, ..
top_k = 3
retrieved = []

for _ in range(top_k):
    best_poem = None
    max_score = -1

    # 遍历找当前最高分
    for item in candidates:
        # 必须确保这首诗还没被选进去过
        if item["score"] > max_score and item["poem"] not in retrieved:
            max_score = item["score"]
            best_poem = item["poem"]

    if best_poem is not None:
        retrieved.append(best_poem)

# retrieved 就是得分最高的前 3 首诗
```

利用 Python 自带的“离线说明书”

- 如果你知道大概是哪个对象（比如字符串、列表），但忘了具体方法名（比如忘了切分字符串是 split 还是 cut），你可以利用 Python 解释器自带的求助功能。

1. 使用 `dir()` 查看所有可用方法

你想处理字符串，但忘了方法名：

Python

```
print(dir("hello"))
```

输出结果会列出一大堆单词，你扫一眼就能看到 `split`, `replace`, `count`, `find` 等，瞬间就能唤醒记忆。

2. 使用 `help()` 查看具体用法

你想起来是 `split`，但忘了参数怎么传：

```python
print(help("hello".split))
```

输出结果会告诉你：`split(sep=None, maxsplit=-1)`，并附带英文解释。

绝招三：面向 `print` 编程（打印调试法）

很多时候写不出代码，不是因为不会函数，而是**不知道传进来的数据长什么样。**

题目给了 `knowledge_base`，如果你不知道里面是啥，千万不要瞎猜。在你的函数第一行直接 `print`：

Python

```
def retrieve_relevant_poems(problem, knowledge_base, top_k=3):
    print("=== 题目长这样 ===")
    print(problem)

    print("=== 知识库第一条长这样 ===")
    print(knowledge_base[0])

    # 打印完后，运行一下 evaluate.py，看终端输出。
    # 看到真实数据后，你自然就知道该怎么用 for 循环去提取了。
    return []
```

绝招四：从提供的文件中“抄”代码

考场上提供了 `evaluate.py` 和 `submission.py` 等文件。

- **仔细看 `evaluate.py`**：评测脚本里通常包含了大量的数据读取、JSON 解析、字符串匹配的代码。如果你忘了怎么读写文件，或者忘了怎么处理字典，去 `evaluate.py` 里逛一圈，大概率能找到现成的代码片段，直接复制过来改改就能用。

在 T2 这种级别的考试中，算法的思路（RAG 的逻辑）
远比代码的优雅程度重要。