探索报告

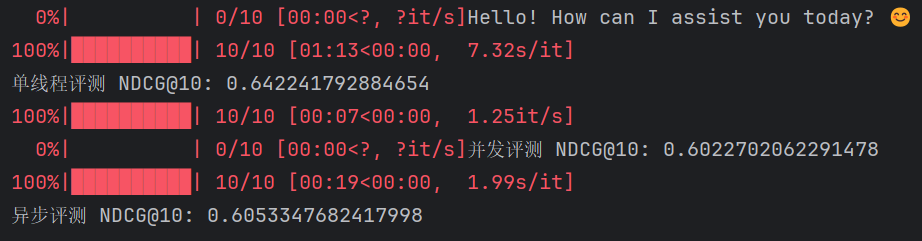
拿到问题后，首先划分确认任务：

1. 搭建大模型评测平台，高效，最好支持多线程并发查询3-8h
2. 修改精调prompt，
3. 留出时间调整修改答案，2h

阅读题目，确定研究思路。

1. 搭建好项目的文件框架，采用S2A，将题目要求抽取成markdown格式。删除关于提交方法的无关信息。
2. 调用API进行评测，非常地消耗时间，构建并发评测与异步评测函数，进行优化。

分析运行结果

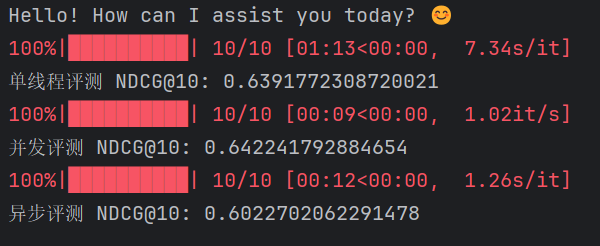


 **并发（线程池）最快的原因**：

* 使用线程池并发发送HTTP请求非常适合IO密集型任务（如API请求），且requests本身效率高，连接池复用使得效率很高。

 **异步（asyncio）略慢的原因**：

* aiohttp可能需要初始化和管理事件循环（event loop）的额外开销。
* 使用asyncio.as\_completed()方法逐个await任务完成的结果可能在少量任务情况下增加了调度开销，降低了速度。
* 异步并未充分发挥性能优势，因为任务数仅10条数据，不足以体现异步IO优势（一般异步方法在大量请求时优势明显）。
* 可能由于每个请求负载本身较低，并发线程池请求在少量任务场景下更直接、高效。



进一步加入日志记录等功能，完善评测框架。

评测体系搭建好后，开始进入PE阶段

从前期最简单的Prompt中，可发现平均得分在0.6左右，最低0.59，最高0.65.

首先进行简单的策略组合，筛选。

具体来说，有

系统角色，设定不同的角色

用户提示风格：指令式，问句式，表格化

候选项列表格式：

用户历史信息。

# 多策略提示生成器  
def construct\_prompt\_variant(d, strategy):  
 system\_roles = {  
 "expert": "你是一名电影推荐专家，目标是根据用户历史预测他们最可能观看的电影。",  
 "analyst": "你是一名数据分析师，请基于用户历史行为重排候选电影。"  
 }  
  
 styles = {  
 "instruction": "请根据用户的兴趣偏好，对下列候选电影重排，最可能观看的放最前，仅返回电影ID列表：",  
 "question": "用户接下来最有可能观看哪些电影？请从以下候选中排序，只返回ID列表："  
 }  
  
 candidates\_fmt = {  
 "id\_first": lambda c: "\n".join([f"{i[0]}: {i[1]}" for i in c]),  
 "name\_first": lambda c: "\n".join([f"{i[1]} (ID:{i[0]})" for i in c])  
 }  
  
 history\_fmt = {  
 "timestamped": lambda h: "\n".join([f"[{i+1}] {x[1]}" for i, x in enumerate(h)]),  
 "plain": lambda h: "\n".join([f"- {x[1]}" for x in h])  
 }  
  
 system\_prompt = system\_roles[strategy['role']]  
 user\_instruction = styles[strategy['style']]  
 history = history\_fmt[strategy['history']](d['item\_list'][-10:])  
 candidates = candidates\_fmt[strategy['c\_format']](d['candidates'])  
  
 prompt = f"用户最近观看的电影：\n{history}\n\n{user\_instruction}\n{candidates}"  
  
 return [  
 {"role": "system", "content": system\_prompt},  
 {"role": "user", "content": prompt}  
 ]



| **设计维度** | **模型偏好** | **原因** |
| --- | --- | --- |
| 角色 persona | expert 通常优于 analyst | “推荐专家”角色的语言风格与微调语料更贴合（如 ChatGPT 常见场景） |
| 表达风格 | instruction > question | 问句更开放，模型倾向生成解释性内容，不易集中排序本身 |
| 候选格式 | id\_first ≥ name\_first | 模型更容易根据 ID 构造排序输出，减少解析歧义 |
| 历史格式 | plain ≥ timestamped | 增加时间信息未必帮助，甚至可能增加噪声或分散注意力 |

这与你的得分高度一致：表现最好的策略依旧是 expert + instruction + name\_first + timestamped，但也未能超越基础设计。

然后我将采用APE技术（自动提示工程师），尝试构建一个前期的高标准。

具体来说，我计划用Agent技术来实现。

我将选用功能强大的模型作为思考规划模型，给出prompt，并根据在deepseek v3的回复与得分，进行评估与优化，然后多轮迭代。