

LLM 推荐方向

目录

1. 推理能力训练 (Reasoning)
 2. Agentic Search 训练
 3. Agent / Tool Use 训练
 4. 奖励模型 / 评估模型
 5. 对齐与安全 (Alignment & Safety)
 6. 多模态后训练 (Multimodal Post-Training)
 7. 后训练数据工程
 8. 长记忆训练 (Memory)
 9. Deep Research / Long-Horizon 规划
 10. 代码 Agent (Code Agent)
 11. 生成式搜索与推荐
-

1. 推理能力训练 (Reasoning)

今年 HC 最多的方向，没有之一。注意这是我给类似方向起的统一名字，反正意思就是利用和增强大模型推理能力的一些落地。

做什么： 训练模型的推理能力。RLVR 基本流程已经是标配了，现在卷的是更深层的问题。当然 pipeline 上不同位置的 reasoning 又有区别，但大部分课题是重叠的：

- **开放域推理的奖励设计：** 数学和代码的可验证奖励已经成熟，真正难的是开放域（逻辑分析、常识推理、复杂决策）——没有标准答案的场景怎么给奖励？生成式奖励 vs 判别式奖励 vs 过程验证器，各家都在探索。更前沿的是把多种奖励信号融合：可验证的部分用规则，主观偏好的部分用 RM，过程质量用 PRM，三者怎么加权、怎么避免互相干扰。
- **过程奖励与验证器：** 不再只看最终答案对不对，而是验证推理的每一步是否合理。过程奖励模型怎么训、验证器怎么和 RL 训练耦合、蒙特卡洛 rollout 估计中间步价值。更深的问题：PRM 标注数据从哪来？人工标注成本极高，自动化的过程标注方案（比如用模型自评 + outcome 回溯）是当前最活跃的研究点。
- **长链路推理的稳定性：** 推理链超过 10 步以后，策略崩溃、奖励稀疏、遗忘等问题全部涌现。渐进式训练（从短链路逐步扩展到长链路）、entropy 正则化、动态记忆状态等针对性方案。还有一个实际问题：长链路的 rollout 成本极高，一条 20 步的轨迹生成时间可能是短轨迹的 10 倍，怎么在有限 GPU 预算下高效训练。
- **推理预算分配 (Test-Time Compute Scaling)：** 不是所有问题都需要长推理。模型怎么自适应地决定“这个问题值得想多久”——简单问题快速出答案，复杂问题分配更多 token。这直接影响产品的推理成本和用户体验，是工程落地最关心的问题之一。
- **Reward Hacking 的深度对抗：** 模型学会了各种刷奖励的取巧策略，怎么检测和防范？多维度奖励冲突怎么解？分层门控和监控回调机制。实际训练中 reward hacking 的花样层出不穷——重复搜索刷检索奖励、模糊回答骗 LLM-Judge、格式正确但内容空洞等等。

- **RL 算法本身的迭代：**GRPO 不是终点，各家都在魔改——怎么做更好的 advantage 估计，怎么处理组内样本方差大的问题，online vs offline RL 的混合训练，group size 和采样策略的调参经验。这些工程细节论文上不写，但直接决定训练效果。
- **自我改进与数据飞轮：**用模型自己产出的高质量轨迹做 SFT 回训，再做 RL 继续提升，形成 RL → 筛选 → SFT → RL 的迭代循环。关键是筛选标准怎么定，多轮迭代后数据多样性怎么保持，什么时候该停。
- **推理能力的跨域迁移：**在数学上训出的推理能力能迁移到代码吗？代码上的能迁移到开放域吗？课程学习（curriculum）的设计，多任务混合训练的数据配比，以及“推理能力”到底是一个统一的能力还是多个独立能力的组合。

需求画像：不只是跑过 GRPO——要能设计和迭代多维度 reward，分析长链路训练中的各种崩溃模式，理解过程奖励和结果奖励的组合策略。

竞争激烈度：★★★★★ — 最热门也最卷，能背出 GRPO 原理的人很多，但真正在开放域奖励设计和长链路稳定性上有深度的人很少。

2. Agentic Search 训练

增长第二快的方向。注意：传统 RAG pipeline 的岗位已经基本消失了，现在都是 Agentic Search。

做什么：用 RL 训练模型“学会搜索”——什么时候该搜，搜什么 query，搜到的结果怎么用，搜到的信息是否可靠。本质上是把搜索能力内化为模型推理能力的一部分，而不是外挂一个检索模块。

目前有两种主要的产品形态：

- **独立的 AI 搜索入口：**比如淘宝 AI 万能搜，各家的 AI 搜索产品，用户直接对着 AI 问问题，模型自主规划搜索策略并生成答案。
- **App 内的 Search Tool 优化：**模型在对话/推理过程中调用搜索工具，核心是优化搜索 tool 的触发时机、query 质量、结果利用率（其实最好是负责整个 tool use 的优化，只不过 search 是最重要的 tool）。

细分任务：

- 搜索 Agent 的 RL 训练：知识边界感知、搜索触发决策、多轮搜索策略
- 检索质量奖励 + 证据奖励：不只是搜了，还要确保模型真的用了检索结果
- 信息提炼与事实核查：从检索结果中提取关键信息、过滤噪声、验证事实
- 推理-检索深度集成：搜索不再是“插入式”操作，而是和推理过程深度交织

需求画像：理解搜索引擎的工作原理 + RL 训练经验 + 信息检索评估方法。传统搜索/推荐背景的人转型做这个方向非常有优势。

竞争激烈度：★★★☆☆ — 方向很热但候选人池子小，懂搜索又懂 RL 训练的人不多，属于供不应求。

3. Agent / Tool Use 训练

从去年的概念验证进入生产化阶段。这块最大的好处是：基本所有重量级 App 都会投入资源做，而且相对容易做出可量化的成果。

做什么： 训练模型的工具调用能力——包括函数调用的格式、参数填充、多步工具链、错误处理和重试。

细分任务：

- 工具调用的 SFT 数据构造：单步/多步/并行/失败场景的数据合成
- 轨迹级 RL 训练：在真实工具环境中做 rollout，用最终任务完成率作为奖励
- 多 Agent 协作训练：规划者、执行者、验证者多个 Agent 的联合优化
- 垂直场景 Agent：搜索 Agent、代码 Agent、数据分析 Agent 各有不同的训练策略

需求画像： 有端到端 Agent 训练经验（不是 LangChain 套壳那种），理解多步决策的 reward 设计和 credit assignment。

竞争激烈度： ★★★★★ — 需求大，但面试官对“真 Agent 训练”和“套框架写 demo”区分得很清楚。

4. 奖励模型 / 评估模型 (Reward Model / Judge)

越来越独立的方向，从 RLHF 的附属品变成了独立的研究和工程方向。

做什么： 训练用于评估模型输出质量的判别器——可以是判别式 RM、生成式 Judge，或者针对特定维度的专项评估器。

细分任务：

- 判别式 RM 训练：偏好数据收集、训练流程、抗风格偏差
- 生成式 Judge：用大模型做评估，prompt 设计和校准
- 过程奖励模型 (PRM)：给推理的每一步打分，需要过程级标注数据
- 安全评估器：识别有害内容、过度拒绝、jailbreak 攻击
- 事实性评估器：原子声明拆解、逐事实核查

需求画像： 理解各类 RM 的训练方法和适用场景，有偏好数据工程经验，能设计和分析评估指标。

竞争激烈度： ★★★☆☆ — 方向重要但知名度不高，很多人不知道这是一个独立方向，竞争相对温和。

5. 对齐与安全 (Alignment & Safety)

稳定需求，不会爆发式增长但持续有 HC。

做什么： 让模型的输出符合人类价值观——不输出有害内容、不过度拒绝、遵循指令、保持一致性。

细分任务:

- RLHF / DPO / KTO 等对齐训练
- 安全偏好数据的构造和清洗
- Red teaming: 系统性地发现模型的安全漏洞
- 过度安全问题: 模型把正常问题也拒绝了, 怎么平衡安全和有用性

需求画像: 有对齐训练经验, 理解安全和有用性的 trade-off, 有 red teaming 方法论。

竞争激烈度: ★★☆☆☆ — 需求稳定但不大, 且很多团队把安全工作分散在各子方向内, 不单独设岗。

6. 多模态后训练 (Multimodal Post-Training)

增长中, 但对候选人的要求门槛较高。

做什么: 在视觉语言模型 (VLM) 上做后训练——图文理解、视觉推理、多模态 Agent。

细分任务:

- 视觉问答和推理的 SFT / RL 训练
- 多模态数据的质量控制 (图文对齐度、OCR 质量)
- 视觉 grounding 和定位能力训练
- 视频理解的长序列处理

需求画像: 有视觉语言模型训练经验, 理解视觉编码器和语言模型的对齐机制, 有多模态评估方法的经验。

竞争激烈度: ★★★☆☆ — 方向在增长但候选人池子更小, 有经验的人很容易拿到好 offer。

7. 后训练数据工程 (Data Engineering for Post-Training)

被严重低估的方向, 但实际上是各团队最核心的壁垒。

做什么: 后训练的数据构造、筛选、清洗、配比——这直接决定了训练效果的上限。

细分任务:

- SFT 数据合成: 用强模型生成轨迹、质量过滤、多样性控制
- RL 数据挖掘: 用 reward 信号定位模型的薄弱点, 定向补充训练数据
- 偏好数据的构造和清洗: 正负样本对的质量直接影响 RM 和 DPO 的效果
- 数据飞轮: 模型产出 → 筛选 → 回训 → 更好的产出, 形成正向循环
- 数据污染检测: 确保评测集不泄漏到训练集

需求画像: 强数据工程能力, 理解数据质量对训练效果的影响机制, 有大规模数据处理经验。

竞争激烈度：★★☆☆☆ — 大部分候选人觉得"做数据"不够高大上不愿意投，但这恰恰是团队最急需的人。

8. 长记忆训练 (Memory)

今年的好方向之一，算法和工程团队都在做，HC 在快速增长。

做什么： 让模型具备长期记忆能力——不是简单的上下文窗口扩展，而是让模型能够跨会话记住用户偏好、积累交互历史、在后续对话中利用之前的信息。

细分任务：

- 跨会话记忆机制：用户偏好记忆、对话历史压缩与检索、记忆的更新和遗忘策略
- 记忆与推理的结合：利用历史记忆辅助当前推理，而不是每次从零开始
- 记忆一致性：确保模型记住的内容不会互相矛盾、不会过时
- 记忆的 RL 训练：什么时候该存记忆，什么时候该调用记忆，记忆召回的准确性怎么作为奖励信号

需求画像： 理解长上下文和记忆机制的本质区别，有检索增强或知识管理的经验。

竞争激烈度：★★★☆☆ — 大部分人还没反应过来，先入场的人优势明显。

9. Deep Research / Long-Horizon 规划

和 Memory 相关但本质不同的独立方向。

做什么： 让模型能针对一个复杂问题做数小时级别的深度调研——自主规划研究路径、多轮迭代搜索-阅读-分析-总结，最终生成结构化的长篇报告。这不是简单的多轮搜索，而是一个 long-horizon 的规划和执行问题。

细分任务：

- 研究路径规划：面对一个开放性问题，模型怎么拆解子问题、决定调研顺序、判断何时信息充分
- Long-Horizon RL：轨迹可能长达数十步甚至上百步，传统 RL 的 credit assignment 完全失效，需要新的训练范式
- 中间状态管理：调研过程中的笔记、摘要、待验证假设怎么组织和更新
- 长报告生成与评估：输出几千字的结构化报告，事实性、逻辑性、完整性怎么评估
- **和 Agentic Search 的区别：** Agentic Search 解决的是"搜得准"，Deep Research 解决的是"研究得深"——前者是几轮搜索，后者是几十轮的迭代规划

需求画像： 有 long-horizon 规划或层次化 RL 的研究背景，理解长文本生成的评估方法，熟悉多步决策的建模。

竞争激烈度：★★☆☆☆ — 虽然类似的产品和代码有非常多了，但对深度研究来说还没真正全面好用的，也就是说还没有形成终极方案，学术界和工业界都在摸索，正是做突破的好时机。

10. 代码 Agent (Code Agent)

从"代码生成"升级为"自主编程 Agent", 独立性越来越强。

做什么: 训练模型能够自主完成软件工程任务——不是生成一段代码片段, 而是理解需求、规划方案、写代码、跑测试、调试修复、提交 PR, 完整闭环。

细分任务:

- 代码 Agent 的 RL 训练: 在真实代码仓库环境中做 rollout, 用测试通过率/CI 结果作为奖励
- 多文件编辑能力: 理解项目结构、跨文件依赖、增量修改而非全量重写
- 调试与自修复: 模型执行代码后遇到报错, 能分析原因并自主修复
- 代码审查和质量评估: 生成的代码不只是能跑, 还要符合规范、可维护

需求画像: 强软件工程背景 + RL 训练经验。纯 ML 背景但不懂工程的人做不了这个方向。

竞争激烈度: ★★★★★ — 需求很大, 但同时具备软件工程深度和 RL 训练能力的候选人非常稀缺。

11. 生成式搜索与推荐 (Generative Search & Recommendation)

传统搜索/推荐团队和大模型团队的交叉地带, 正在快速扩张。

做什么: 用大模型重构搜索和推荐的核心链路——不是在搜索结果上面套一层 LLM 做摘要, 而是用生成式模型直接参与召回、相关性判断、排序。

目前的落地现状:

- **召回侧已经可以落地:** 用大模型做 query 理解和语义召回, 效果显著优于传统方法, 延迟也在可接受范围内。
- **相关性和排序是最大的挑战:** 大模型做 pointwise/pairwise/listwise 排序的效果很好, 但 latency 是个巨大的瓶颈——传统排序模型几毫秒就搞定, LLM 排序动辄几百毫秒到秒, 在线服务根本扛不住。
- **当前的折中方案:** 用大模型离线做标注/蒸馏, 训练轻量级的排序模型上线; 或者只在 rerank 阶段对 top-k 结果用 LLM 精排。

细分任务:

- 生成式召回: LLM 直接生成候选文档 ID 或关键词, 替代传统的倒排/向量检索
- LLM 排序: 用大模型对候选结果做相关性打分和排序
- 延迟优化: 怎么把 LLM 排序的延迟从秒级压到百毫秒级——量化、缓存、投机解码、模型蒸馏
- 搜索/推荐专用 RM: 训练专门评估搜索结果质量的奖励模型
- 用户意图理解: 从短 query 中理解用户真实需求, 生成结构化的检索计划

需求画像: 传统搜索/推荐系统经验 + 大模型训练/部署经验。两边都懂的人极度稀缺。

竞争激烈度：★★★☆☆ — 方向很大但跨界门槛高，传统搜推的人不懂 LLM 训练，LLM 的人不懂搜推系统，能两边都搭上的人很抢手。

校招方向建议

优先**方向 1（推理训练）**，理由是技术成长和薪资上限都是最高的。推理训练方向的 tech lead 普遍级别高、资源充足，进去之后能接触到的技术深度和实验规模是其他方向比不了的。而且这个方向的能力是可迁移的——RL 训练、reward 设计、训练稳定性这些经验，换到 Agent、搜索、代码任何一个方向都能用上。

然后是**方向 2 和 3（Agentic Search / 各类 Agent Tool-Use 训练）**，就是上面讲的两种产品形态。这两个方向的好处是需求大但竞争没有方向 1 那么卷，有搜索或工具调用相关课题的同学很容易脱颖而出。

偏 research 背景的同学，可以投**方向 8（Memory）和方向 9（Deep Research / Long-Horizon）**——长记忆 RL、long-horizon 规划、Deep Research 等等。这些方向暂时还没有形成终极方案，学术界和工业界都在摸索，正是出论文和做突破的好时机。如果你的课题和这些方向沾边，现在入场的时间窗口非常好——再过一两年方向成熟了，先发优势就没了。

想赌未来的去**方向 10（代码 Agent）**。这个应该意识到了吧——代码 Agent 正在重塑整个软件工程的工作方式。目前这个方向的特点是：需求在爆发式增长，但学术界的研究还跟不上工业界的需求，组里真正能做事的人极度稀缺。如果你同时有 CS 工程能力和 ML 背景，这个方向的天花板非常高。

社招方向建议

首选**方向 2（Agentic Search）和方向 3（Agent 训练）**。这两个方向的特点是供不应求、薪资溢价高——懂搜索又懂 RL 训练的人太少了，懂端到端 Agent 训练（不是套框架那种）的人也不多。如果你在上一家做过搜索 Agent 或 Tool Use 的完整训练闭环，从数据构造到 reward 设计到训练调到上线评估都走过一遍，这种经验可以直接平移，面试评分通常不会低。

方向 11（生成式搜索推荐） HC 也够，但薪资溢价会低一些——因为这个方向更多是传统搜推团队在转型，薪资体系还是沿用搜推的标准，没有纯 LLM 方向那么激进。不过好处是竞争也相对温和，传统搜推背景的人只要补上 LLM 训练的短板就很有竞争力。

方向 10（代码 Agent） 对有强工程背景的社招候选人特别友好。很多做了几年后端/基础架构的工程师，转型做代码 Agent 反而比纯 ML 背景的人更有优势——因为这个方向需要的不只是模型训练能力，更需要对真实软件流程的深度理解（项目结构、依赖管理、CI/CD、代码规范）。

方向 7（数据工程） 不激进但非常稳。社招里有一类很吃香的画像：在大厂做过大规模数据处理、理解数据质量对训练效果的影响、有过从“数据问题”反推“训练问题”的经验。这种人不愁找工作，而且工作内容比想象中有意思——你实际上在决定模型能力的上限。

一个实际的建议：社招跳槽时，尽量选一个和你现有经验有交集的细分方向，而不是完全从零开始。比如你现在做搜推，去方向 2 或 11 比去方向 1 更容易拿到好评级；你现在做后端工程，去方向 10 比去方向 4 更自然。面试官看的是“你在这个方向上能不能快速产出”，有相关经验的人天然占优。

最后说一个趋势

从我最近面试的情况来看，有一个很明显的信号：**基础八股的考察权重在持续下降，项目深度才是真正的胜负手。**

今年秋招和社招，想拿到好一点的 offer，最忌讳的就是项目重复度高——换个数据集跑一遍、把开源 repo 拿来魔改一下、或者几个项目本质上是同一套流程换了个皮。面试官每周面几十个人，这些套路一眼就能识别。

搞清楚你投的岗位到底属于上面哪个子方向，面试前花时间研究一下这个方向目前在做什么、卡在哪儿、下一步可能往哪走。能在面试里展现出对方向的判断力，比背一百道八股题管用。