实现方案—4 层生成法

下面给你一份**通俗但尽可能详细**的「Step 2.0｜4 层生成法方案说明」（不含提示词），按你给的 Input Pack v2 / Output Spec v2 来讲清楚：**每一步在做什么、解决什么、重点是什么、优点是什么**。你可以直接放进 SOP 文档给团队用。

Step 2.0｜4 层生成法详细方案（通俗版）

这一步到底要解决什么问题?

一句话：**把"可能有人会搜的问题"尽可能穷举出来**，形成一个"问题宇宙"（Query 候选池），后面再做去重聚类、筛选、监控。

现实里你会遇到两个坑：

1. **漏**：只想到"推荐/对比"，漏掉"抱怨式人话、合规/安全、私有化部署、报价、案例"等关键流量入口
2. **乱**：生成一堆句子但没有结构，后面没法聚类、没法统计、没法复用

所以 Step 2.0 的核心不是"写几百条 query"，而是把它变成**可重复跑的生产过程**：有输入、有规则、有配额、有结构化输出。

0）准备阶段：把"生成条件一次说清楚"（Input Pack v2）

在真正生成前，先把输入包整理好（你提供的 YAML 就是标准）。

0.1 这一步在做什么?

把"模型需要知道的一切"一次性喂进去，让它生成时不乱猜、不跑题。

0.2 重点解决什么?

- **品牌是谁**：避免模型生成行业泛词、不带品牌语境
- **要覆盖什么标签**：避免只生成少数热门标签，漏掉关键能力/场景/风险
- **按什么分类体系产出**：避免产出无法落表、无法统计
- **要遵守什么规则**：避免广告腔、避免超泛词、避免明显重复

- **要多少量**：避免偏科（比如只输出 scenario，不输出 negative）

0.3 优点是什么？

- 以后换品牌/换行业，**只换 Input Pack，流程不改**
- 可追踪：有 run_id、可复跑、可对比
- 可控：配额+规则让输出稳定

1）L1：种子层（Seed）——“先把问题目录铺出来”

1.1 这一步在做什么？

用标签做“拼装”，生成一批**最基础、最典型**的搜索问题雏形（像目录）。

它不会追求语言花样，而是追求：

覆盖所有核心组合（品牌、行业、场景、能力、竞品、风险、价格、部署、合规、案例.....）

1.2 重点解决什么？

- **防漏**：先把“应该存在的问题类别”全部站住
- **搭骨架**：后面 L2/L3/L4 都会围绕这些骨架扩展

1.3 产出长什么样（通俗理解）

- 更像“关键词搜索”或“标准提问”：
  - “智能投研 公司 推荐”
  - “研报生成 工具 哪家好”
  - “xxx 私有化 部署”
  - “竞品A vs 竞品B”
  - “大模型投研 合规 风险”
- 每条会标好：它属于哪类 query_type（recommend/compare...）、属于哪个阶段（认知/考虑/决策/负面）

1.4 优点是什么？

- 后续好管理：因为 L1 本身就像**索引/目录**
- 方便查漏：你一眼能看出来缺了哪类（比如缺 pricing / deployment）

2）L2：问法扩展层（Paraphrase）——“同一个意思，用户会怎么搜？”

2.1 这一步在做什么？

对 L1 的每个“主题骨架”，生成多种不同说法，但**意思不变**。

换句话说：

L1 是“我要搜这个事”，L2 是“我可能用哪些中文句式/词汇来搜”。

2.2 重点解决什么？

- **覆盖真实搜索表达差异**：用户不会只用一种说法
- **解决“同义词/口语化”**：比如“靠谱吗/准不准/会不会胡编”其实是一类需求
- **提升后面监控命中率**：你用更少的簇覆盖更多真实输入

2.3 产出长什么样

同一语义的不同表达，例如都在问“研报生成工具推荐”：

- “研报生成工具推荐”
- “自动写研报的软件有哪些”
- “研报AI工具哪家好用”
- “有没有能生成研报的AI”
- “研报生成靠谱么”

同时它会保留“溯源”：

- 这条 L2 是从哪条 L1 扩出来的（seed 字段）

2.4 优点是什么？

- 你后面做聚类时更省力：同簇内容更完整

- 能覆盖“专业词 + 大白话”两种流量入口
- 很适合中文搜索环境（大量关键词式、口语式输入）

3）L3：角色真实问法层（Role Simulation）——“人话入口 & 抱怨式需求”

3.1 这一步在做什么？

让模型分别站在不同角色视角（行业分析师/基金经理/风控/IT/数据/普通投资者），生成他们真的会搜的问法。

L3 的关键词是：**真实 + 人话 + 场景压力**

经常会出现“抱怨+需求”这种表达。

3.2 重点解决什么？

- **解决 L1/L2 的“术语化偏差”**：L1/L2 容易太标准、太像咨询报告
- **补齐高价值流量入口**：很多商业价值最高的问题，恰恰是“人话”表达
- **把 painpoint/risk 拉满**：特别是负面/合规/安全类，靠 L3 补齐很有效

3.3 产出长什么样

更像工作中的真实困扰：

- 分析师：“公告太多看不过来，怎么自动总结重点？”
- 基金经理：“投研材料怎么快速汇总成结论？”
- 风控负责人：“大模型输出怎么做合规审计？会不会胡编？”
- IT/数据负责人：“私有化部署怎么接内部数据？权限怎么管？”
- 普通投资者：“财报解读有没有靠谱的AI工具？”

每条必须能落到至少一个标签上：

- 场景/能力/痛点/风险（否则就丢掉）

3.4 优点是什么？

- 很大概率产出你团队“想不到但真实存在”的 query

- 对负面与合规特别有帮助（自然语言更丰富）
- 对后续内容策略也很有价值（能直接变成 FAQ/选题）

4）L4：场景任务深挖层（Scenario Task Decomposition）——“把场景拆成可执行问题”

4.1 这一步在做什么？

针对你输入的 `scenario_tasks`（场景→任务→交付物→约束），把场景拆到“任务颗粒度”，生成微场景问题。

L4 的关键词是：**具体、可执行、可交付、带约束**

它会围绕“输入-处理-输出-约束-对接”去问。

4.2 重点解决什么？

- **让 Query 从“泛场景”变成“任务链问题”**

  例如不只“财报季解读怎么做”，而是：

  - “财报发布后怎么快速抓关键指标？”
  - “风险点能自动标注并分级吗？”
  - “生成的摘要能不能做到引用来源可追溯？”
- **把部署/合规/数据源/可复现等约束拉进来**（这些往往决定决策）

4.3 产出长什么样

围绕 deliverables/constraints 产生问题：

- 输出交付物：要点摘要/风险提示/关键指标表/研报大纲/图表
- 约束：时效性/可追溯/合规表述/可复现/数据口径一致
- 对接：数据源/权限/系统对接/私有化部署

而且 L4 会强制写 `scenario_task`：

- 例如：`财报季解读/快速解读财报`

4.4 优点是什么？

- 最贴近“产品能不能解决”的落地问题（非常适合后续销售/产品/内容）
- 对“监控”最友好：因为问题足够具体，信号更清晰
- 为后续 2.1 聚类打标提供高质量颗粒度语料

5）Step 2.0 最终“合并交付”（不是 2.1 去重）

5.1 这一步在做什么？

把 L1/L2/L3/L4 的结果合并成一个 JSON，严格符合 Output Spec v2，并做**最基础的校验**：

- 字段是否齐全（required_fields）
- 数量是否满足各 query_type 的最小配额（min_per_type）
- 总量是否超过 soft_limit（超过就优先裁剪低价值重复项）
- 每条是否带合理 related_tags（type:tag）
- L3 必填 role、L4 必填 scenario_task（口径一致）

5.2 重点解决什么？

- 让输出“能直接进入 2.1”，不需要人再清洗格式
- 让产物稳定：每次跑出来结构一致、可比较、可复用

5.3 优点是什么？

- 降低后续人工成本（不用再整理格式）
- 方便统计：你能马上看到哪个 query_type 或哪个 layer 覆盖不足

6）为什么 4 层比“一次性生成一堆 Query”更好？

如果只让模型“一次生成 2000 条”，常见问题是：

- 前 200 条质量还行，后面开始漂移、重复、跑题

- 负面/合规/部署类严重不足
- 结构乱，后面聚类打标成本大

4 层的优势是“分工明确”：

- **L1 防漏**（骨架）
- **L2 防单一表达**（同义与口语）
- **L3 防不真实**（人话入口）
- **L4 防不落地**（任务颗粒度）

最后得到的不是“句子堆”，而是一个**可进入流水线的候选池**。
