

首次LMCC青少年组交流活动-试题分析

命题主旨

本次活动旨在增加广大的青少年对大模型相关知识的兴趣以及对 LMCC 能力认证的了解，题目由易到难分为四题。

- T0 是入门题，旨在帮助考生了解使用大模型和使用传统算法解题的区别。考生需要同时使用两种方式解题，在解题过程中可以看到“隐藏测试集”，方便考生理解后续题目中可见的测试集不是全部的测试数据。
- T1 是对考生提示词编写能力的考查，这个是最常见的大模型使用技能。
- T2 是对考生模型训练能力的考察，对考生的要求更高，需要有一定的经验才能完成。
- T3 是情境题，是考生长文档阅读理解、新知识学习等多方面能力的考察。考生需要根据英文论文或中文说明文档来完成代码填空。

T0-CSP

💡 这里把 T0 设计成两道题，是出于教学考虑：T0-CSP 要求考生先把自然语言描述拆成可审计的规则表，并处理边界条件、优先级和稳定 JSON 输出；T0-LMCC 则要求把同一套手工规则翻译成模型可执行的提示词说明书。两题共同考察的是“先把规则讲清楚，再让程序或模型稳定执行”。

T0-CSP 本质上是一个规则归一化题。题面和评测器都明确要求 `gate_id` 的编号落在 `01` 到 `12` 之间，因此门号解析至少要覆盖公开与隐藏数据里实际出现的 `2号门`、`十号门`、`11号门`、`12号门` 这些情况，也建议完整覆盖 `十一号门`、`十二号门` 这类中文写法。

步骤	实现要点
第 1 步：准备区域映射、状态关键词和中文数字表	先把后面要用的规则全部整理成表，而不是把判断逻辑写散。区域映射只需要覆盖 E/W/N/S/I/O 六类；状态词分成 fault、closed、open 三组；中文数字至少覆盖一到九、十、十一、十二，阿拉伯数字也要处理，才能完整覆盖 01 到 12。
第 2 步：根据区域关键词确定 <code>gate_id</code> 的字母前缀	从自然语言里抽取“东环/西区/内侧/外环”等信息，归一化成标准前缀。按关键词表顺序匹配即可；若完全匹配不到，可以回退到默认值 <code>I</code> 。
第 3 步：提取“几号门”，并把数字统一转成两位编号	这是整个题最容易漏边界的部分，不能只处理个位中文数字。正则要同时兼容阿拉伯数字、中文个位数，以及 <code>十</code> 、 <code>十一</code> 、 <code>十二</code> 这类写法；最后统一补零成两位。
第 4 步：按 <code>fault > closed > open</code> 的优先级判定状态	状态判定不是“谁最后出现就算谁”，而是固定优先级短路。只要命中故障语义，就应直接输出 <code>fault</code> ；否则再判断 <code>closed</code> 和 <code>open</code> 。
第 5 步：组装标准输出，保证键名、顺序和格式都稳定	评测看的是精确输出，不只是语义接近。最终只输出 <code>gate_id</code> 和 <code>status</code> 两个键，键顺序固定，并保证 JSON 是单行字符串。

- “门号解析不要只写中文个位数”这个提醒是正确的，而且不是经验判断，而是题面约束和测试数据共同决定的：README 明确要求编号在 `01` 到 `12` 之间，公开数据里已经有 `11号门` 和 `12号门`。
- 如果只写 `一/二/三/.../九` 这种最短规则，那么 `十号门`、`十一号门`、`十二号门` 这样的输入会漏掉。
- 因此更稳的写法是：先用一个正则统一命中“X号门”，再把命中的 token 分情况转成整数。

代码实现请参考 `submission.ans.py`。

T0-LMCC

T0-LMCC 的核心不是写复杂 prompt，而是把 T0-CSP 的同一套规则用清晰、稳定、可约束的方式交给模型。目标不是“让模型理解大意”，而是让模型稳定输出一行符合格式的 JSON。

这题的难点主要有四类：输出格式、区域映射、门号标准化、状态优先级。Prompt 需要把这些规则直接写清楚，并明确禁止模型输出解释文字。

模块	应该说明什么	作用
输出格式	只能输出一行 JSON，只能包含 <code>gate_id</code> 和 <code>status</code> 两个键。	先保证格式稳定，避免模型输出解释、代码块、额外键或前后缀。
区域映射	明确列出东环/西区/北侧/主环/外环等说法分别映射到哪个字母。	把自然语言位置压到固定的 E/W/N/S/I/O，减少前缀漂移。
门号规则	说明编号必须补成两位，并至少覆盖 <code>2 号门</code> 、 <code>十号门</code> 、 <code>十一号门</code> 、 <code>十二号门</code> 、 <code>11号门</code> 、 <code>12号门</code> 。	避免模型只学会个位数补零，而漏掉两位编号或中文“十”。
状态优先级	明确写出 <code>fault > closed > open</code> ，并列每类关键词。	当一句话里同时出现多个状态词时，仍按固定优先级输出。

实现时建议先把输出格式写在 Prompt 最前面，再列区域映射、编号规则和状态规则。少量示例可以帮助模型学会 JSON 形状，但示例不应替代规则本身。

需要特别注意：

- `gate_id` 必须是 `X-YY`，编号范围是 `01` 到 `12`。
- 门号解析不要只覆盖中文个位数，`十号门`、`十一号门`、`十二号门`、`12号门` 都需要稳定处理。
- 状态不是按最后出现的词判断，而是按 `fault > closed > open` 的优先级判断。
- 生成参数保持简单即可，通常只需要限制输出长度，并关闭 thinking，让输出更短更稳定。

代码实现请参考 `submission.ans.py`。

T1

T1 考查的是 Prompt 控制能力。题目不要求训练模型，也不要求写复杂解析程序；核心是把一段自然语言调度口令稳定整理成三行固定格式。

这题的难点主要有三类：任务名抽取、时间归一化、泊位保留。Prompt 需要把这三类规则讲清楚，并明确禁止模型输出解释文字。

模块	应该说明什么	作用
输出格式	必须恰好三行，前缀固定为 <code>任务：</code> 、 <code>执行时间：</code> 、 <code>泊位：</code> 。	先保证格式稳定，不让模型输出解释、代码块或额外前后缀。

时间规则	写明基准日 2026-03-23，以及周几、明天、今晚、晚 x 点、x 点半、月日补年份等规则。	把相对时间转成唯一的 YYYY-MM-DD HH:MM，避免模型自由推断。
任务抽取	优先使用明确任务代号；其次使用引号内任务；否则提取核心动作短语。	避免把时间、泊位或泛化后缀混进任务字段。
泊位抽取	保留原文中最具体的位置短语，不要缩写或改写。	保证 银湾-04泊位、东侧检修桥2号泊位、三号补给臂C-6 这类地点能原样输出。

实现时建议先把格式要求写在 Prompt 前半部分，再写时间规则和字段抽取规则。最后放少量示例即可，示例的作用是让模型学会三行输出和字段边界，不需要堆很多。

需要特别注意：

- 执行时间 必须是真实合法时间，格式是 YYYY-MM-DD HH:MM。
- 任务 字段不要带“任务/作业/事项/安排”这类泛化后缀，除非它本身就是任务名的一部分。
- 泊位 字段不要自行规范化，例如不要把原文地点压缩成自造缩写。
- 生成参数保持简单即可，通常只需要控制 max_new_tokens，并关闭 thinking，让输出更短更稳定。

T1 的评分由字段正确和格式正确两部分组成。格式错不一定意味着字段全错，但会丢格式分；字段内容采用严格字符串匹配，所以 Prompt 中要尽量减少模型改写。

代码实现请参考 submission.ans.py。

T2

T2 的核心是训练一个可离线提交的 Qwen3-0.6B 工具调用模型。



T2 分数分为数据和模型两部分。只要完成数据合成，并提交格式正确的 tool_agent_train.jsonl 与 tool_agent_sft_train.jsonl，就可以在 A 项“数据分布和多样性”中获得最高 10/30 的分数。后续利用这份数据训练出可用模型，才继续影响公开集和隐藏集的性能分。

数据合成思路

需要先判断哪些工具经常会在同一条请求中一起出现。这个步骤可以人工设计，也可以借助大模型辅助归纳。确定工具组合后，再围绕每一种组合生成不同说法的用户请求。为了提高数据多样性，可以从三个维度扩展样本：工具组合、潜在用户类型、用户使用工具的具体场景。通过这三类维度交叉组合，可以系统性地放大数据规模，同时避免训练集只是在同一句话上做表面改写。例如：

- 工具组合：hotel_booking + weather + email
- 潜在用户：出差的项目经理
- 使用场景：下周去上海开客户会，需要提前确认住宿、天气和通知团队

具体的数据合成流程建议如下：

1. 先枚举工具组合，覆盖单工具、双工具、三工具和四工具并行请求。
2. 对每一种工具组合，可以换几个不同的用户身份、使用场景和表达方式来改写。比如同一组工具可以写成办公场景、出差场景，也可以写成简短指令或口语化请求。每类准备少量变化，就能形成足够大的候选池。
3. 每条候选先构造结构化答案，再生成自然 query，不让生成模型自由决定工具名和参数。

- 4. 做严格后处理，要求关键参数直接出现在 query 中。
- 5. 从候选池中选约 100 条作为正式训练集；超过 100 条不会继续提高 A 项数量分，只有修复明确错误时才值得增加。
- 6. 生成 SFT 数据时使用 Qwen 工具调用格式，assistant 目标是连续的 `<tool_call>...</tool_call>`。
`wait_for_result` 应固定绑定到工具类型，不建议让生成模型自由决定：
 - `weather / translation / stock_quote`: `false`
 - `calendar / calculator / map_route / email / hotel_booking`: `true`

满分数据样例

样例统一用 semantic_cluster_ratio 的值命名数据文件，不再额外使用独立别名。因此 artifacts/ 下共有 8 个 jsonl 文件：每个 ratio 对应一份原始训练集和一份 SFT 训练集。

对应关系固定为：

代码块

```
1 tool_agent_train.<semantic_cluster_ratio>.jsonl
2 tool_agent_sft_train.<semantic_cluster_ratio>.jsonl
```

例如 semantic_cluster_ratio 为 0.43 时，对应文件就是：

代码块

```
1 tool_agent_train.0.43.jsonl
2 tool_agent_sft_train.0.43.jsonl
```

下面只保留训练后达到 A=10, B=12, C=8, T2=30/30 的数据。每一行的 semantic_cluster_ratio 都可以按上面的规则映射到一组 tool_agent_train 和 tool_agent_sft_train 文件。

semantic_cluster_ratio	数据合成思路
0.32	先从公开评测题中抽取 12 组工具组合种子；每组按 2 个用户画像 × 2 个使用场景 × 2 种表达方式扩展，共 96 个设置。每个设置生成 2 条 query 候选，最后根据分数自动筛成 100 条。
0.38	先整理 18 组覆盖单工具与多工具链路的工具组合；每组按 2 个用户画像 × 2 个使用场景 × 2 种表达方式扩展，共 144 个设置。每个设置生成 4 条 query 候选，再从候选池筛成 100 条。
0.42	沿用 0.38 的 18 × 2 × 2 × 2 × 4 的初始数据，在最终 100 条里只替换 6 条语义过近的用户请求；工具、参数和等待标注保持不变。
0.43	沿用 0.38 的 18 × 2 × 2 × 2 × 4 的初始数据，在最终 100 条里替换 10 条低风险语义改写样本，在保持参数严格的前提下进一步提高多样性。

易错点：

- `email` 的 `subject/body` 必须在 query 中说清楚，否则小模型容易把正文做语义压缩。
- `stock_quote` 应直接给股票代码和日期，避免把公司名到 ticker 的外部知识作为评测前提。
- `wait_for_result` 必须与 `arguments` 同级，不能写进 `arguments` 内。

- 并行场景要连续输出多个 `<tool_call>`，不要输出解释文字或 markdown。
- 只追求更高 `semantic_cluster_ratio` 可能伤害 B/C，参数严格性比表达花样更重要。

训练代码填空

💡 这里的训练代码看起来比填空本身复杂，是出于教学考虑：它展示了一个真实可运行的小模型 full fine-tuning 脚本需要哪些工程保护。考生真正要掌握的是稳定超参、全参数 optimizer/scheduler、以及正确的 loss 缩放和反向传播。

这一节只需要补三个短函数：默认训练参数、optimizer/scheduler 构造、以及单步 forward/backward。外层训练循环已经提供，下面按填空位置说明每一处应返回什么、以及它和完整训练流程的关系。

填空位置	解答要点	为什么这样写
TODO-1	返回已验证的默认训练参数和 checkpoint 目录。答案中使用 <code>batch_size=4</code> 、 <code>grad_accum=4</code> ，有效 batch size 为 <code>16</code> ； <code>max_seq_length=1024</code> ，每 2 个 epoch 保存一次 checkpoint。	这一步把可复现的训练规模固定下来，并允许通过环境变量覆盖。答案里给 <code>20</code> 个 epoch 作为训练上限，实践中可根据 loss 和评测表现选择 6-10 个 epoch 附近的较优权重。
TODO-2	用 <code>AdamW(model.parameters(), lr=learning_rate, weight_decay=weight_decay)</code> 构造 optimizer，再用 <code>LambdaLR(optimizer, lr_lambda=lr_lambda)</code> 构造 scheduler。	题目要求 full fine-tuning，因此 optimizer 应覆盖全部模型参数；scheduler 使用外层训练循环已经定义好的 warmup + decay 学习率函数。
TODO-3	在 autocast 上下文中执行 <code>model(**batch)</code> 得到 loss，将 <code>loss / grad_accum</code> 用于 backward；如果 scaler 启用则走 <code>scaler.scale(...).backward()</code> ，否则直接 <code>backward()</code> ，最后返回未缩放的 loss 数值。	除以 <code>grad_accum</code> 是为了让梯度累积后的尺度等价于更大的 batch；optimizer step、clip grad 和 scheduler step 已经在外层循环统一处理，不应在这个函数里重复执行。

训练超参数

前一节的三个填空分别接通默认训练参数、optimizer/scheduler 和单步反向传播；这些代码最终服务于下面这组 full fine-tuning 配置。这里的目标不是把训练规模做大，而是在显存可控、结果可复现的前提下，让 `Qwen3-0.6B` 稳定学会 Qwen 工具调用格式、`wait_for_result` 标注和参数复制。

代码块

```
1 base_model = Qwen3-0.6B
2 fine_tuning_type = full
3 precision = bf16
4 per_device_train_batch_size = 4
5 gradient_accumulation_steps = 4
6 effective_batch_size = 16
7 learning_rate = 2e-5
8 weight_decay = 0.01
9 warmup_ratio = 0.03
10 max_seq_length = 1024
11 num_train_epochs = 20
12 save_every_epochs = 2
13 gradient_checkpointing = true
```

预期：

- 默认训练上限是 20 个 epoch
- 训练过程中建议保留每 2 个 epoch 的 checkpoint，并优先在 6-10 个 epoch 之后选择公开集表现更稳定的一版
- `target_loss <= 0.001` 可作为较理想的收敛参考；实际选择仍以公开集评测和输出格式稳定性为准

T3



做 T3 时，可以把它当成一次“把论文公式写成代码”的小练习：你不需要复现完整的 DIVERSED 系统，只需要理解题目已经拆好的动态验证核，并补全 4 个 Python 函数中的缺失逻辑。建议先读 `README.md` 建立整体认识，再读 `代码与论文公式对照.md` 对齐每个公式和函数；完成这两步后，再查看 `2604.07622v1.pdf`，把题目里的简化实现放回论文背景中理解。

这道题的解题顺序可以概括为：先用 verifier head 算当前步动态权重 `w_t`，再用 `w_t` 混合 target/draft 两个分布，最后按 relaxed accept rule 做接受或回退。题目已经给出归一化、接受概率、回退分布和全连接层前向等辅助函数，考生只需要补全验证链路中最关键的四个接口。

文件结构

类别	文件	作用
论文背景	<code>2604.07622v1.pdf</code>	DIVERSED 原论文。考生不需要复现完整论文系统，但需要知道本题来自 relaxed speculative decoding 与 dynamic ensemble verification。
公式桥梁	<code>代码与论文公式对照.md</code>	最重要的辅助资料。它解释 Equation (3)、Equation (4)、Equation (6) 分别如何映射到题目函数。
题面说明	<code>README.md</code>	说明题目背景、输入输出、样例、本题做过的确定性化简，以及 <code>T3_real_demo/</code> 与正式评测的关系。
考生填空	<code>submission.py</code>	只需要补全 4 个 TODO：动态权重、动态混合、单步验证、序列验证。
固定参数	<code>ensemble_head.json</code>	题目给出的已训练 verifier head。输入维度 12，隐藏维度 8，激活函数为 ReLU，输出一个 scalar logit。
评测代码	<code>evaluate.py</code>	评测分为核心函数单测和公开 case，相同逻辑也会用于检查最终输出结构与数值。

解题主线

这一节可以按一条验证链路来读：先用两路 hidden 特征预测当前步给 target 分布的权重，再用这个权重把 target/draft 两个概率分布混成验证分布，最后根据验证分布决定接受草稿 token 还是进入回退。四个 TODO 分别对应“算权重”“算分布”“做单步决策”“汇总多步结果”。

1. `dynamic_ensemble_weight`：从特征预测 target 权重

输入是当前步的 `h_q` 和 `h_p`。把二者拼接后送入题目给定的两层 verifier head：

- 第一层用 `_dense_forward(x, W1, b1)` 和 ReLU 得到 hidden
- 第二层用 `W2` 与 `b2` 得到一个 scalar logit，最后经过 sigmoid 得到 `w_t`。这里的 `w_t` 就是论文 Equation (3) 中动态预测出的 target 权重。

2. `ensemble_distribution`：用动态权重构造验证分布

先把 `target_probs` 和 `draft_probs` 都归一化，再逐 token 计算 $w_t * target + (1 - w_t) * draft$ 。得到的 mixed 分布再做一次归一化后返回，对应论文 Equation (4) 的 v_t^{θ} 。

3. `verify_single_step`：完成一次接受 / 回退决策

这一层负责把前两个函数串起来：先得到 `w_t` 和 `verify_probs`，再调用题目已给出的 `acceptance_probability` 与 `residual_distribution`。

- 若 `uniform_u <= accept_prob`，说明草稿 token 被接受；
- 否则不能随机采样，必须从 `residual_probs` 中取 `argmax` 作为回退 token。这一步对应论文 Equation (6)。

4. `run_dynamic_verification`：把单步验证串成序列输出


最后一层只做编排，不再引入新公式。按各输入序列的最短长度循环调用 `verify_single_step`，并把每一步的最终 token、接受标记、动态权重和接受概率分别收集到 `tokens`、`accepted_mask`、`weights`、`accept_probs` 中。

评分结构

评分项	分值	说明
A：四个核心函数单测	16/20	每个 TODO 函数 4 分。评测会检查动态权重、混合分布、单步验证结果和序列验证结果，浮点数按约 <code>1e-6</code> 的精度比较。
B：公开评测集	4/20	运行 <code>run_dynamic_verification</code> 处理公开 cases，主要比较最终 token、接受 mask、动态权重和接受概率。

因此，这题只要四个函数逻辑正确，公开集也会自然拿满。不要额外改数据、改 `ensemble_head.json`，也不要随机采样引入答案。

T3_real_demo



这个目录的教学作用，是帮助考生把“函数填空”放回真实双模型推理链路中理解；其中的代码也提供了解题线索。但它仍是纯 Python 小型机制演示，不代表论文级或生产级加速实现。

论文中的方法若要转化为大幅端到端加速，还需要结合 block-level 并行验证、高效 KV cache 复用、底层 attention/kernel 优化，以及推理 runtime 的调度与工程实现。

前面的题解只讨论正式评测所需的 `submission.py`：它把 DIVERSED 的验证核抽成确定性的函数填空，输入已经是整理好的概率分布和低维特征。`T3_real_demo/` 则是一个额外的教学案例（考试过程中也提供给了考生），用来展示这些函数背后更接近真实系统的双模型推理流程。正式评分仍然只看 `T3/submission.py` 和评测脚本，这个目录不参与计分。

文件	建议关注点
<code>readme.md</code>	先读这个文件，了解 demo 的定位、运行方式、输出指标，以及它和 T3 评测的区别。
<code>run_qwen_diversed_benchmark.sh</code>	一键运行入口。它会设置 draft/target 模型路径，必要时训练 dynamic ensemble head，然后运行 benchmark。

run_real_qwen_diversed.py	主脚本，支持 <code>train</code> 、 <code>generate</code> 、 <code>benchmark</code> 。这里能看到真实双模型前向、分布混合、接受 / 回退决策等和填空题对应的实现线索。
diversed_train_small.jsonl	训练 dynamic ensemble head 的小样本，用来演示如何从任务奖励中学习动态权重。
diversed_benchmark_small.jsonl	小型 benchmark 数据，用来比较普通 speculative decoding 和 DIVERSED 的耗时、吞吐、接受率与奖励指标。

如果只是想试跑流程，可以在准备好模型路径后进入该目录运行：

代码块

```
1 cd T3_real_demo
2 FORCE_TRAIN=1 bash ./run_qwen_diversed_benchmark.sh
```

运行后建议重点看 `T3_real_demo/T3_artifacts/benchmark_时间戳/report.md` 和 `sample_outputs.jsonl`。前者给出汇总指标，后者能看到每条样本的真实生成结果。这里的目标不是追求考试分数，而是观察：draft model 先提出 token，target model 再验证，dynamic ensemble head 如何影响接受率和最终输出。

如果是为了反推正式题的实现，可以按下面顺序读代码：

1. 先在 `run_real_qwen_diversed.py` 中找 dynamic ensemble head 如何输出 `w_t`，对应 `dynamic_ensemble_weight`。
2. 再看 target/draft 概率如何按 `w_t` 混合，对应 `ensemble_distribution`。
3. 接着看接受概率和 residual fallback 如何进入单步验证，对应 `verify_single_step`。
4. 最后看 benchmark/generate 如何把多个 token 步串起来，对应 `run_dynamic_verification`。

文件结构

路径 / 文件	作用
.	赛后答案包根目录。
考试说明.md	整套考试说明：题型、数据命名、环境路径、交卷要求和赛后复现注意事项。
result_store.py	统一写出各题分数与明细，生成 JSON/Markdown 结果报告。
./data/	公开数据、种子数据与最终评测数据。
T0_test_data.jsonl	T0 公开评测数据。
T0_test_data.final.jsonl	T0 最终评测数据。
T1_visible_test_data.jsonl	T1 公开评测数据。
T1_visible_test_data.final.jsonl	T1 最终评测数据。
T2_seed_tasks.jsonl	T2 数据合成种子样本。
T2_public_eval_queries.jsonl	T2 公开评测 query 与标注。

T2_public_eval_queries.final.jsonl	T2 最终评测 query 与标注。
T3_public_cases.jsonl	T3 正式 evaluate.py 默认读取的公开案例。
T3_public_cases.final.jsonl	T3 赛后补充案例和答案对照材料。
./img/	题目配图。
星港概念图.png	星港主题概念图。
./T0_CSP/	T0 传统规则解法。
README.md	T0-CSP 题面、评分和运行说明。
evaluate.py	T0-CSP 官方评测入口。
submission.py	考生原始实现文件，核心函数是 normalize_gate_report_csp。
submission.ans.py	T0-CSP 赛后参考答案。
./T0_LMCC/	T0 大模型 Prompt 解法。
README.md	T0-LMCC 题面、评分和运行说明。
evaluate.py	T0-LMCC 官方评测入口，加载模型并测试 Prompt 输出。
submission.py	考生原始实现文件，核心是 build_messages 和生成参数。
submission.ans.py	T0-LMCC 赛后参考答案。
./T1/	Prompt 控制与结构化输出题。
README.md	T1 题面、时间规则、评分和运行说明。
evaluate.py	T1 官方评测入口。
submission.py	考生原始实现文件，用于构造 system prompt 与生成参数。
submission.ans.py	T1 赛后参考答案。
./T2/	工具调用数据合成与 full fine-tuning 题。
README.md	T2 题面、数据合成思路、评分结构、训练和评测说明。
evaluate.py	T2 官方评测入口，检查数据质量并加载模型评测公开/最终集。
tool_agent_common.py	T2 工具 schema、wait_for_result 规则、数据解析与模型目录检查。
train_tool_agent.py	T2 训练骨架，考前版本只在少量函数处留空。
train_tool_agent.ans.py	T2 训练代码参考答案。
./T2/artifacts/	T2 数据与模型产物。
README.md	T2 artifacts 目录说明。
tool_agent_train.jsonl	原始训练数据格式样例，1 条。
tool_agent_sft_train.jsonl	Qwen SFT messages 格式样例，1 条。

tool_agent_train.0.32.jsonl	semantic_cluster_ratio=0.32 的满分原始训练数据。
tool_agent_sft_train.0.32.jsonl	与 0.32 原始数据配套的 SFT 数据。
tool_agent_train.0.38.jsonl	semantic_cluster_ratio=0.38 的满分原始训练数据。
tool_agent_sft_train.0.38.jsonl	与 0.38 原始数据配套的 SFT 数据。
tool_agent_train.0.42.jsonl	semantic_cluster_ratio=0.42 的满分原始训练数据。
tool_agent_sft_train.0.42.jsonl	与 0.42 原始数据配套的 SFT 数据。
tool_agent_train.0.43.jsonl	semantic_cluster_ratio=0.43 的满分原始训练数据。
tool_agent_sft_train.0.43.jsonl	与 0.43 原始数据配套的 SFT 数据。
model/	训练后完整模型目录；无模型答案包中可能不包含权重。
./T3/	DIVERSED 论文机制填空题。
README.md	T3 题面、阅读顺序、评分结构和 real demo 说明。
2604.07622v1.pdf	DIVERSED 论文原文。
代码与论文公式对照.md	代码变量与论文公式对照说明。
ensemble_head.json	T3 动态权重头参数。
evaluate.py	T3 官方评测入口。
submission.py	考生原始实现文件，需要补全动态验证核四个函数。
submission.ans.py	T3 赛后参考答案。
./T3_real_demo/	T3 真实双模型小型演示，不属于正式评分入口。
README.md	demo 定位、数据格式、训练和 benchmark 运行说明。
run_real_qwen_diversed.py	真实加载 Qwen3-8B + Qwen3-0.6B 的 train/generate/benchmark 主脚本。
run_qwen_diversed_benchmark.sh	一键运行脚本，支持 PYTHON_BIN、QWEN_ROOT、TARGET_MODEL、DRAFT_MODEL。
diversed_train_small.jsonl	dynamic ensemble head 训练小样本。
diversed_benchmark_small.jsonl	普通 speculative decoding 与 DIVERSED 对比 benchmark 小样本。