多模态模型“破甲”与回归测试

Model Prompt Armor & Regression Playbook

Silvite Translate Lab · mimo-v2.5 漫画模式排序修复实战沉淀

目标

把这次被实测验证有效的提示词技术与测试方法论，沉淀成可迁移的模型接入模板。

核心结论

单靠自然语言规则无法稳定收敛；需要组合“提示词定位 + 结构化补丁 + 代码确定性兜底 + 回归测试”。

来源：docs/model-prompt-armor-and-regression-playbook.md

速览：这份 Playbook 解决什么

这份文档来自 Silvite Translate Lab 的真实多模态调试过程。目标不是追求“神 Prompt”，而是建立一套可复用的工程方法：先识别模型的稳定失败模式，再把语义依赖结构化，由代码执行确定性约束，并用机器判定的回归测试防止过拟合与误判。

问题本质	模型空间先验、规则服从与输出格式都可能成为故障源，不能把所有失败都归因于“模型没理解”。
最高收益	把规则写进对应 schema 字段描述；用 id + reply_to/depends_on 把语义依赖显式化。
代码职责	验证结构、执行依赖约束、确定性渲染与 fallback；代码不猜语义。
测试原则	机器判定 + 原始响应留痕 + 对照组 + N 次批次统计；单次 PASS 不算收敛。

一、问题背景：为什么要“破甲”

任务：让多模态模型输出“真实阅读顺序”的结构化漫画 segments（panel / order / speaker / type），而不是按 OCR 空间顺序平铺。

- 空间先验极强：同侧多个气泡按空间连读，语义关系被忽略；实测约 50% 概率把“回应”排到“质疑”之前。
- 抽象规则执行力差：写了“必须检查问答关系”，模型确认理解，但生成时可能不执行。
- 规则过度服从：硬规则过强会在别的样本上过矫正，例如把质疑句提前到一切之前。
- 自洽的错误：错误输出内部也能自圆其说，事后仅靠“再强调一次”很难解决。

结论：单靠自然语言规则无法稳定收敛（实测约 50% 正确率天花板），必须组合提示词定位、结构化补丁和代码确定性兜底。

二、破甲技术清单（按实测效果排序）

2.1 规则写进 schema 字段描述 [5/5]

把硬规则嵌入 JSON schema 对应字段的说明，而不是单独成段。模型在生成该字段的瞬间会读到它。

```
"order": "全局真实阅读顺序，从 1 开始递增。硬规则：回应永远排在它所回应的句子之后，即使回应的气泡在画面上更靠右或更靠下；质疑句若本身是对前句（解释/致歉）的回应，也仍须排在该句之后"
```

实测对比：同样的话写在正文 bullet 里 adherence 约 50%；写进字段 description 后明显提升。

模板：每条硬规则都问一句“这条规则在生成哪个字段时最该被想起？”，把规则塞进那个字段的 description。

2.2 结构化补丁：模型声明依赖，代码强制执行 [5/5]

把抽象语义变成显式结构：给每个元素分配 id，用 reply_to 声明“我在回应谁”，代码只执行声明。

```
若 A.reply_to = B.id，则 order(A) > order(B)；违反时交换两者 order。
```

这是把“模型猜语义”变成“模型声明语义 + 代码执行声明”。声明正确时可以得到确定性执行；声明方向错仍是残留缺陷。

模板：任何“X 必须在 Y 之后”的需求，优先设计成 id + depends_on/reply_to 字段，再由代码做拓扑修正。

2.3 可证伪的不变量检查 [4/5]

“请检查对话连贯性”太模糊；“如果出现可检测的异常形态，则执行确定动作”更有效。

```
悬空检查：若排序结果以未获回应的问句/质疑句收尾，几乎总是排错了；把在语义上回应它的陈述句移到质疑之后。
```

模板：领域常识尽量写成“异常形态 + 确定动作”。异常必须能从输出直接判定，动作必须具体。

2.4 表面线索启发式 [4/5]

把需要语用推理的判断降维成模型可直接执行的表面模式，例如词形、标点、关键词复现。

```
词语回声：回应句常复现被回应句的关键词，例如“为什么”->“因为”，“真的吗？”->“是真的”。当问句与含其关键词的陈述句并存时，问句应在前。
```

2.5 WRONG -> RIGHT 对比例 [4/5]

给出一个与真实失败拓扑相似、但文本不同的小例子，同时明确标注错误链与正确链。

```
错误：对不起 -> 今天是真的 -> 真的吗?
正确：对不起 -> 真的吗？ -> 今天是真的
```

2.6 证据优先级链 [3/5]

模型会抓错信号，需要显式给出权重顺序，并把容易误导的信号压到最低。

```
回应语义关系 > 问答关系 > 质疑/辩解关系 > 气泡尾巴归属 > 画格视觉流 > 阅读方向 > speaker 是否交替（最低权重参考）
```

2.7 软信号框定，硬规则只留给可证伪项 [3/5]

有反例的信号不能写成硬排序规则，只能作为“触发复核”的软信号。

```
同一说话人连续多个气泡时，不自动改序；只触发连贯性复核。若两句本身确实连续，必须保留同 speaker 连续。
```

2.8 已知残留缺陷

- 声明方向错：模型偶尔把依赖方向写反；代码无法在不猜语义的前提下纠正。
- 互为 reply_to：可能出现 A.reply_to=B 且 B.reply_to=A。代码侧应对无序对只执行一次，并优先执行“被违反”的那条约束。
- 上游波动：偶发空响应（约 1/10，重试即可）；偶发把整个 JSON 双重编码塞进 translation 字段，需要代码解包防御。

三、已验证无效 / 反效果的做法

做法	结果
长篇抽象规则堆 system prompt	~50% 天花板；继续堆更多条也不涨
降低 temperature（0.3 -> 0.1）+ 精简指令	更差：出现过矫正，质疑被提前到一切之前
“两人轮流说话”硬规则	真实漫画存在合法连续同角色；且实测不稳定
事后重复强调同一规则	无效；错误输出内部可自洽
复杂 responsive class 复用同一套 DOM	前端侧同构教训：可读性与布局稳定性下降

四、回归测试模板

4.1 机器判定，不靠人眼

写 verdict 脚本：断言相对顺序、断言不变量、断言格式，并输出一行 JSON PASS/FAIL。人眼查看 CJK 日志容易误判，机器判定才适合稳定回归。

```
{"verdict":"PASS","orders":{"a":3,"b":4,"c":5,"d":6},"envInDialogue":false,"labels":true}
```

4.2 对照组防过拟合（必须）

对照	目的
同角色连续两句确实合理的样本	防止修复把合法连续强行重排
最简问答样本	防止基础能力被新规则破坏
与主样本同拓扑但不同文本	验证修复是模式级的，不是样本级的

合成样本可行：SVG 画气泡 / 人物 / 尾巴 -> sharp 转 PNG。画完必须先人眼检查渲染，再送测。

4.3 稳定度用批次说话

单次 PASS 可能只是运气。每轮改动后同输入运行 N>=3-4 次并报告正确率；正确率低于 3/4 视为未收敛，继续迭代或降级为“已知不稳定”。

4.4 原始响应留痕

verdict 脚本同时把 raw response 追加写入 jsonl。失败分类需要依赖原始日志；本次关键发现（互为 reply_to、双重编码、方向反写）都来自原始响应，而不是最终 UI。

4.5 失败模式分类表

失败模式	现象	归属
空间翻转	同侧气泡连读，回应排在质疑前	模型默认行为
过矫正	质疑被提到一切之前	规则过强 / temperature
方向反写	reply_to 指错对象	模型声明错误，代码不可救
互为 reply_to	A<->B 互相声明	代码需配对去重 + 违反者优先
空响应	上游偶发	重试

双重编码	JSON 塞进 translation 字符串	代码解包
order 缺失 / 为 0	环境文字 order=0	代码回退到数组原序

4.6 职责边界：最终收口

模型：看图、分结构、判语义、声明依赖（id/reply_to）、给初版 order。

代码：验证结构、执行声明约束、确定性渲染、fallback。

原则：代码不猜语义，模型不负责最终排版稳定性。

五、迁移到其他模型 / 任务的检查单

1. 需求里所有“必须 X 在 Y 前” -> 能否改成 id + 依赖字段 + 代码拓扑执行?
2. 每条硬规则 -> 有没有反例？有反例的降级为软信号。
3. 每条硬规则 -> 它在生成哪个字段时最该被想起？塞进那个字段的 schema description。
4. 领域语感点 -> 能否降维成表面模式（词形 / 标点 / 复现）？
5. 写一个 WRONG -> RIGHT 对比例，拓扑贴近真实失败案例。
6. 列证据优先级链，显式压制最误导的信号。
7. 写至少 3 个对照样本（含一个“修复不能破坏”的反例）。
8. verdict 脚本 + raw 留痕 + N 次批次，报正确率，不报单次。
9. 代码兜底链：结构执行 -> fallback 到模型自由输出 -> 错误态；任何结构失败都不能导致空结果。

附：建议的模型接入分层

以下分层是对本文方法的结构化呈现，用于迁移时快速定位“应该改哪一层”。内容均来自本文职责边界与回归方法。

层	职责
Core / Mode Prompt	描述任务目标与风格，不承担模型特定怪癖
Provider-specific Rules	只放该模型已验证的行为修正，避免把旧模型病历迁移到新模型
Schema Contract	在字段生成点放硬规则；优先显式依赖字段
Output Normalizer / Parser	处理空响应、双重编码、格式漂移等协议层异常
Deterministic Code	执行依赖约束、排序、渲染与 fallback
Regression Harness	机器 verdict、raw 留痕、控制组、批次稳定度

一句话收口：先把模型的“稳定失败模式”变成可观察、可证伪的结构，再决定由 Prompt、Schema、Parser 还是代码接管。