

[return_dim_tag_20251...](#)

文件

[return_fact_details_202...](#)

文件

你是一名资深的亚马逊美国站点的产品总监，同时也是运营总监，你让一名运营小白根据return_dim_tag的标签定义和边界进行退货留言的打标，return_fact_details是运营小白交付的打标结果，现在请你对交付结果的准确度逐条进行打分

已思考 7m 19s >

已思考 1m 18s >

先给你一个整体结论，然后逐条打分。

评分规则（我这边统一定义一下）：

- **1 分**: 完全符合这个维度的定义和边界，是“理想答案”
- **0.5 分**: 方向大体对，但和定义的边界有偏差 / 有更合适的主标签
- **0 分**: 不符合该标签定义，不推荐打这个标签

整体来看：

- **1 分: 45 条**
- **0.5 分: 5 条**
- **0 分: 8 条**
- 严格按“完全正确=1”的准确率约 **78%**，把 0.5 也算作半对，整体有效率约 **82%**。

标签定义来自 `return_dim_tag`，评分依据为定义与边界说明。 `return_dim_tag_20251114`

打标结果来自 `return_fact_details`。 `return_fact_details_20251114`

逐条打分与点评（按文件顺序）

序号	review_id	tag_code	得分	简短点评
1	R1LTUUWSTZ9AG G	INSTALL_FIX	0.5	只抱怨“组装很烦”，没说调好后易位移/需频繁紧固，更像“安装复杂”，和INSTALL_FIX 定义有偏。
2	R1O02YVMHQ8M NS	MATERIAL_RUST	1	“the screws are not rustproof”=螺丝不防锈，完全符合“材质/防锈失效”。
3	R87ZB4398VG0G	STABILITY_TIP	1	“Cannot stand my dish” 理解为架不住盘子，对应“稳定性/防倾倒不足”合理。
4	RCYO6MIQT6IE1	SCENE_ADAPT	0.5	问题是“占满水槽右侧”，属于场景使用不理想，但未出现倾斜/下坠/回流水等典型故障，勉强算场景适配不足。
5	RCYO6MIQT6IE1	GUIDE_VARIANT	0.5	“I should have checked the specs.” 有误选成分，但用户也自认是自己没看清，是否归因于指引不清不确定。
6	R1CXWTGPPXE6R Q	PAGE_MISSING	1	“Was unaware it needed to be put together” = 不知道要组装，可视为页面对安装要点提示不够。
7	R1EUI60POM0L4D	MATERIAL_RUST	1	用了 5 个月生锈，典型材质/防锈失效。

序号	review_id	tag_code	得分	简短点评
8	R1FKKN3V47NYGP	SCENE_ADAPT	0	文案里是“fit almost any sink”正向评价，不是“场景适配不足”。这个标签不该打。
9	R1FKKN3V47NYGP	STABILITY_TIP	1	“a little wobbly when stretching it out full length”是晃动问题，完全符合稳定性不足。
10	R1GCBKRVQ36CPT	FIT_COMPAT	1	“did not fit”+退货，典型“尺寸/兼容性不符”。
11	R1GCBKRVQ36CPT	GUIDE_VARIANT	0.5	“Much smaller than anticipated”有预期管理/选型误差，可能和指引相关，但也可能是买家自己没看尺寸。
12	R1OTE4Q3RSD124	CAPACITY_EFF	1	“Muy pequeño/非常小”，直接指向容量/有效面积偏小，可视为容量不足。
13	R2CZ4AZ0E543P5	INSTALL_FIX	1	“The screws fell out numerous times”需要不断处理，典型固定性差。
14	R2UQOC992TAEO 7	CS_POOR	0	客服让寄回，用户抱怨“去 UPS 太麻烦”，更多是退换流程/自身不便，不是客服态度或处理不达标。
15	R2UQOC992TAEO 7	INSTALL_FIX	1	用一个月螺丝开始脱落、要用铁丝绑住，完全是安装/固定性问题。

序号	review_id	tag_code	得分	简短点评
16	R38AZO0RUC1G17	PAGE_MISSING	1	"I did not realize that you would have to assemble once you received it." 典型"页面未清晰提示需要组装"。
17	R3H0XMUHSAO28	FIT_COMPAT	1	"too small for my kitchen sink" 明确尺寸不合。
18	R3H0XMUHSAO28	SCENE_ADAPT	0	根因是"太小", 按边界说明: 尺寸不合应归 FIT_COMPAT, 不应再打 SCENE_ADAPT.
19	R3J3G4WC0X1E1V	DOA_DAMAGE	1	出厂配件没装好, 一部分橡胶件只是丢盒子里且无法自己安装 = 开箱即功能不完整, 典型 DOA/缺件。
20	R3J3G4WC0X1E1V	PARTS DESIGN	0	问题核心是"少装/装错", 不是结构设计本身不合理, 因此不建议打 PARTS DESIGN.
21	RI20ZLLKL9DYG	FIT_COMPAT	1	"Didnt fit my sink at all and I have a standard size sink." = 标称应适配标准水槽却没适配, 完美命中。
22	R1P8UT2IK6F4E7	MATERIAL_RUST	1	两个月就生锈, 典型防锈失效。
23	R26Z7K293JZOHP	STABILITY_TIP	1	"Flimsy" 可视为结构发软、不稳, 归为稳定性不足合理。

序号	review_id	tag_code	得分	简短点评
24	R3T9650HZPHPC0	SOCIAL_GAP	0	"Not what I expected" 是预期落差，没有提高评分/热卖等社会证明，与定义不符。
25	R3T9650HZPHPC0	STABILITY_TIP	1	"not sturdy very wobble on top of sink" 是非常典型的晃动/不稳定问题。
26	R1DIR5UMMQVM 8V	LOOK_COLOR	0	"its not stainless steel" 是材质/规格问题，不是外观色差，应考虑SPEC_MISMATCH 或材质相关标签。
27	R1DIR5UMMQVM 8V	INSTALL_FIX	1	"little knobs ... move around" 位置锁不住，属于固定性/调节性差。
28	R1DIR5UMMQVM 8V	PARTS DESIGN	1	餐具筐浅、感觉廉价，是典型“刀叉篮/杯架设计不合理”。
29	R1DIR5UMMQVM 8V	VALUE_WEAK	1	"not worth what I paid""should be \$10 or less" 直接表达价格价值感不佳。
30	R1H5889KTI8IHE	MATERIAL_RUST	1	不到一年就生锈，符合材质/防锈失效。
31	R18VMLQIBLIK2W	MATERIAL_RUST	1	用 3 天就生锈，更是典型防锈失效场景。
32	R2AVPEX1OYOUY Q	INSTALL_FIX	1	需要不断重新拧螺丝、复位橡胶件，完全符合INSTALL_FIX 定义。

序号	review_id	tag_code	得分	简短点评
33	R2AVPEX1OYOUYQ	VALUE_WEAK	1	“会去找替代品”=明确有换购倾向，符合VALUE_WEAK。
34	R2EKMFA0HY6EZ7	CAPACITY_EFF	1	“不能很好最大化空间”“像缩小版的大架子”=有效面积不足。
35	R2EKMFA0HY6EZ7	PARTS DESIGN	1	碗盘在排水区“坐不稳”，是分区结构导致的使用问题。
36	R2IJTUIDOBGGTR	FIT_COMPAT	0.5	“mostly fits... sides still open”只是两侧有空，不是明显无法放置，勉强算兼容性略差。
37	R2IJTUIDOBGGTR	PARTS DESIGN	1	餐具架过大，应更短才能稳固，是典型配件设计不合理。
38	R3441UHDAP5UL8	VALUE_WEAK	1	“看起来很低调又廉价，不喜欢”=整体价值感偏低，归 VALUE_WEAK 合理。
39	R3GDDPAC4WALFE	INSTALL_FIX	0	问题是“需要组装”和“Flimsy”，对应 PAGE_MISSING + STABILITY_TIP，更不是“调好后固定性差”。
40	R3GDDPAC4WALFE	SOCIAL_GAP	0	仅“Not what I expected”，同样没提社交证明，高评分/热卖等。
41	R3LFGFO6Z93AP5	SPEC_MISMATCH	1	“it said rust resistance...that's why I bought it”且实际生

序号	review_id	tag_code	得分	简短点评
				锈，规格宣传与实物不符。
42	R3LFGFO6Z93AP5	MATERIAL_RUST	1	"turns rusty bad" 是典型材质/防锈失效。
43	RD7VR09KC85WW	VALUE_WEAK	1	"Very cheaply made. Do not recommend" = 质量感差且明确不推荐，符合价值/对比劣势。
44	R2UPS2YYUD7M70	FIT_COMPAT	1	"rack didn't fit my sink properly because my faucet... too wide" = 与实际水槽结构不兼容。
45	R2UPS2YYUD7M70	VALUE_WEAK	1	材料感觉便宜、质疑耐用性与价格匹配度，是典型价值感偏低。
46	R2UPS2YYUD7M70	STABILITY_TIP	1	"It's also a bit flimsy... don't feel confident with heavy dishes" 典型稳定性/承重可靠性不足。
47	R2Z4UK8H91SCGR	CAPACITY_EFF	1	伸展后有大缝，东西会掉下去 = 有效可用面积被浪费，容量效果不达标。
48	R2Z4UK8H91SCGR	PARTS_DESIGN	1	刀叉杯太小且太高挂不上，是典型配件几何/结构问题。
49	R3DK7EH9J86G2C	LOOK_COLOR	1	"a little different than the illustration shows" = 实物和图示不一致，归外观/颜色不符合合理。

序号	review_id	tag_code	得分	简短点评
50	RG4YQ1P976ZJY	INSTALL_FIX	1	螺丝位置刚性、无法灵活调整外侧，属于安装/调节位置受限。
51	RGOK60EVGW06H	CAPACITY_EFF	1	"Not very big" 直指容量/有效面积偏小。
52	RGOK60EVGW06H	VALUE_WEAK	1	"Strangely made and very awkward. I dont recommend." 虽未提价格，但整体体验差+明确不推荐，可归为价值/对比劣势。
53	R38XZO2C596REH	DOA_DAMAGED	1	缺所有连接件，无法使用，是标准缺件/DOA场景。
54	RO4T32WPBCGRD	PARTS DESIGN	1	没有防倒的横杆导致高杯子容易倾倒，典型杯架结构设计缺失。
55	RO4T32WPBCGRD	STABILITY_TIP	1	"Cups fall over..." 是直接的防倾倒/稳定性问题。
56	R384TSBX2ZQOS	PARTS DESIGN	1	刀叉杯因横杆阻挡卡不上，是配件设计问题。
57	R384TSBX2ZQOS	STABILITY_TIP	1	伸缩机构造成台阶，餐具“不能平放”，属于放置稳定性问题。
58	R384TSBX2ZQOS	DOA_DAMAGED	1	"doesn't sit straight; bent out of shape" 说明到货即变形，不是正常结构，符合损坏到货。

总结&训练建议（给这位运营小伙伴的）

1. 容易混淆的几个标签：

- **SOCIAL_GAP**: 必须出现“评分/热卖/大家都说好”等“社会证明”线索，否则只是“和预期不同”，不该用这个标签。
- **SCENE_ADAPT vs FIT_COMPAT**: 如果根因是尺寸不合（太大/太小），优先 **FIT_COMPAT**；只有尺寸 OK，但在某场景会倾斜、回流水、下坠等，再用 **SCENE_ADAPT**。
- **INSTALL_FIX vs STABILITY_TIP / PAGE_MISSING**:
 - 只说“难装”“需要组装”→更接近 **PAGE_MISSING**（页面没写清）
 - 说“晃、软、flimsy、not sturdy”→优先 **STABILITY_TIP**
 - 说“螺丝总掉、需频繁拧紧、位置锁不住”→才是 **INSTALL_FIX**
- **LOOK_COLOR vs SPEC_MISMATCH / MATERIAL_RUST**:
 - “不是不锈钢/材质不对”→考虑 **SPEC_MISMATCH** 或 **MATERIAL_RUST**（如伴随生锈）
 - “颜色和图不一样/涂层不均”→才是 **LOOK_COLOR**

2. 做得不错的地方：

- 所有跟 **MATERIAL_RUST**、**DOA_DAMAGED**、**VALUE_WEAK**、**PARTS DESIGN**、**STABILITY_TIP** 相关的大部分案例，判断都很稳。
- 对于多标签场景（同一条 review 打多个维度），整体覆盖较完整，很少漏掉重要维度。

如果你后面想，我也可以帮你把这些错误样本单独整理出来，当作运营培训题库，再配几道“纠错练习题”给新人用。

来源



假设把误差原因都归属于我们自己，请你根据评分结果（<1分），哪些标签的定义和边界需要进一步的完善，是否需要新增标签，但是新增和定义的时候注意全局的MECE原则

已思考 36s >

已停止思考 >

假设把误差原因都归属于我们自己，请你根据评分结果（<1分），哪些标签的定义和边界需要进一步的完善，是否需要新增标签，但是新增和定义的时候注意全局的MECE原则

已思考 7m 10s >

先把结论说清楚：

从所有打分<1的样本看，真正暴露问题的不是操作员，而是我们自己的标签设计在几个地方不够 MECE（边界有重叠/空白）。

需要重点优化的标签（按优先级）：

INSTALL_FIX、SCENE_ADAPT、FIT_COMPAT、GUIDE_VARIANT、CS_POOR、SOCIAL_GAP、LOOK_COLOR、PARTS DESIGN (和 DOA_DAMAGED 的边界)
建议新增 1 个标签：“组装复杂/耗时高”(暂名：INSTALL_COMPLEX)

下面我按“问题簇”来讲：哪里不 MECE、怎么改定义&边界、是否需要新标签。

一、安装相关：INSTALL_FIX 太“窄”，导致被乱用

涉及误差样本：

- “very frustrating to put together” (组装很烦) → 被打成 INSTALL_FIX
- “Needed to be put together. Flimsy” → 被打 INSTALL_FIX，而不是 PAGE_MISSING + STABILITY_TIP

1) 现在的问题

- INSTALL_FIX 的定义是**“装配虽易，但调好后易位移、需要频繁紧固或无法锁定位置”**。
- 但评论里大量是：“**步骤多、螺丝多、说明难懂、自己装起来很痛苦**”，并没有说“调好后还老是松”。

也就是说：

“过程痛苦”和“结果锁不稳”目前被塞在一个标签里，而定义又写得很偏“结果”，操作员只能猜。

2) 建议：新增一个安装维度标签（必要）

新增标签（建议）： INSTALL_COMPLEX

- **tag_name_cn:** 组装复杂/耗时高
- **建议归类：** 也在结构/使用成本相关大类里（和 INSTALL_FIX 同一层级）
- **定义（草案）：**

“安装/组装步骤繁琐、耗时长、需要额外工具或说明书难懂，导致用户明显表达‘装起来很痛苦’。”

- **边界建议：**
 - 若抱怨是“装好了之后还总要拧螺丝/位置老是跑” → **INSTALL_FIX**
 - 若抱怨是“页面没说要自己组装，我以为是整装” → **PAGE_MISSING**
 - 若抱怨是“Flimsy/not sturdy/wobbly” → **优先 STABILITY_TIP**，而不是 **INSTALL_FIX**

同时微调 `INSTALL_FIX` 定义：

加一句：“不包含‘仅因步骤多、说明不清导致安装过程挫败’的情况，这类归入‘组装复杂/耗时高’。”

这样安装相关就变成一个 MECE 的小树：

1. 页面没讲清要自己装 → **PAGE_MISSING**
2. 说明写了，但安装过程太折腾 → **INSTALL_COMPLEX** (新)
3. 装好后老是松/位置锁不住 → **INSTALL_FIX**
4. 结构本身软/晃/易倒 → **STABILITY_TIP**

二、适配&场景：**FIT_COMPAT / SCENE_ADAPT / GUIDE_VARIANT** 三兄弟边界不够锐

典型误差样本：

- 想要更长/更大：“I would have preferred that it could expand ... Completely on me though - I should have checked the specs.”

被打：`SCENE_ADAPT + GUIDE_VARIANT`

- "Much smaller than anticipated and did not fit. Returned."
- 被打：**FIT_COMPAT + GUIDE VARIANT**
- "too small for my kitchen sink / smaller than my RV sink"
- 同时被打：**FIT_COMPAT + SCENE_ADAPT**

1) 现在的问题

- FIT_COMPAT 定义里写“**在标称适配范围内仍无法稳固放置**”，容易让人误解成：“评论里没提‘页面写了适配范围’，就不敢打 FIT_COMPAT。”
- SCENE_ADAPT 定义是“某/多场景下出现倾斜、下坠、回流水等”，但实际使用时也被用在纯尺寸问题上（太小/太大）；
- GUIDE VARIANT 的定义是“说明不足导致误选”，但样本里很多只是“我主观以为会更大/更长”，并未说“页面不清楚”。

2) 建议：优化这三个的定义&边界（不一定要新标签）

(1) 放宽 FIT_COMPAT 的前提表述

调整 FIT_COMPAT 定义为：

“实际与用户目标位置/设备（如水槽尺寸、柜体尺寸）不匹配，导致**无法放入/无法跨放，或放入后间隙过大严重影响使用**。”

不要求评论里必须提‘页面宣称适配’。”

边界补充一句：

- 若只是“放得下，但某个姿势略有不便”，无明显功能问题 → 不打 FIT_COMPAT，按照 SCENE_ADAPT 或其他来。

(2) 收紧 SCENE_ADAPT

强调：

“SCENE_ADAPT 仅用于：尺寸本身能放下，但在特定场景（双水槽、RV、窄台面）下出现倾斜/下坠/回流水等现象。”

根原因为尺寸完全不合的情况 → 一律 FIT_COMPAT。”

(3) 收紧 GUIDE VARIANT 应用条件

在定义和边界里加“必要条件”：

- “出现类似字眼：“listing confusing / pictures misleading / sizes all on one page not clear / hard to tell which size to choose / should be clearer which size fits””
- “或用户明确把责任归因于**页面/卖家指引**，而不是“完全怪我自己”。”

并在边界里写清楚：

- 只说“smaller than expected / I should have checked the specs” → **默认不打 GUIDE_VARIANT, 仅打 FIT_COMPAT.**

这样三者就更接近 MECE：

- 完全放不下/明显尺寸不合 → FIT_COMPAT
 - 尺寸 OK, 但在某特殊场景下有倾斜/回流 → SCENE_ADAPT
 - 多规格/多尺寸商品, 页面指引混乱 → GUIDE_VARIANT
-

三、服务/退货：CS_POOR vs RETURN_POLICY_BAD 分工不清

典型误差样本：

- “客服让寄回, 但 UPS 很远, 我懒得去” → 被打成 CS_POOR

1) 现在的问题

- CS_POOR (品牌/店铺服务不达标) 定义偏客服响应/态度；
- 但样本里抱怨的是：**退货渠道对偏远用户不友好 (要开车去 UPS, 很麻烦)** —— 更像退换流程/政策设计问题。

2) 建议：不新增标签，强化边界

强化 RETURN_POLICY_BAD 定义：

明确包含：

- “退货地点选择少、离家太远”
- “必须指定快递点 (如 UPS) 对某些地区非常不方便”
- “包装、打印标签、预约取件等流程复杂”

同时把 CS_POOR 边界写死：

仅当用户抱怨：

- “客服慢、不回消息”
 - “承诺的补偿/换货没有兑现”
 - “给错信息/推诿等
- 才打 CS_POOR。

对于“流程设计本身对我不友好”，一律归 RETURN_POLICY_BAD。“

四、感知类：SOCIAL_GAP 的“社会证明”要写死，不然容易被当成“预期落差”

典型误差样本：

- “Not what I expected.”
- “I thought it would be a little bit different...”

都被打成 SOCIAL_GAP，但评论里完全没有提“高评分/热卖/好评”等社会证明。

1) 现在的问题

SOCIAL_GAP 目前定义：

“热卖/高评等社证与实际体验差距大，削弱信任。”

但实际操作中，“Not what I expected”太像“预期落差”，容易被误扔到这里。

2) 建议：不新增“预期落差”标签，只把 SOCIAL_GAP 收紧

- 在定义中加一句**必须出现社会证明相关线索**：如‘评分 4.8 却很一般’、‘评论都说很好结果一般’、‘bestseller 但体验不如其他’等。
- 在边界写明：

仅有“not what I expected / I thought it would be different”且无法从文本推断“是被评分/评论影响预期”的，**不打 SOCIAL_GAP**，只按具体问题打功能标签（如 STABILITY_TIP、CAPACITY_EFF 等），没有就可以不打“体验维度”的标签。

这样避免新增一个非常模糊、极易跟所有维度重叠的“预期落差”标签，保持 MECE。

五、外观/材质：LOOK_COLOR 与 SPEC_MISMATCH 的边界不清晰

典型误差样本：

- “Its sturdy but its not stainless steel.” → 被打成 LOOK_COLOR

1) 现在的问题

- LOOK_COLOR 目前是**“色差/涂层不均/边角发灰”**，本质是“视觉效果与图不符”；
- 而“不锈钢 / 材质”其实是**客观规格**，更接近 SPEC_MISMATCH；但 SPEC_MISMATCH 定义现在偏“伸缩范围/承重/重量/容量等硬指标”。

2) 建议：扩大 SPEC_MISMATCH 范围，不新增新标签

修改 SPEC_MISMATCH 定义：

“伸缩范围、承重、重量、容量、材质（如是否不锈钢）、是否可洗碗机等**客观规格**与页面宣称不符。”

在 `LOOK_COLOR` 边界加：

- “若投诉点是“材质/是否不锈钢/是否金属/是否塑料” → **SPEC_MISMATCH**”
- “`LOOK_COLOR` 仅用于“颜色/光泽/图案/视觉效果”的不一致。”

这样“外观色差”和“材质规格不符”就被拆开，互斥清晰。

六、设计 vs 质量：**PARTS DESIGN** 与 **DOA DAMAGED** 的拆分

典型误差样本：

- 出厂没把橡胶件装好，用户自己装不上：“It is something that has to be done during manufacture.”
→ 被打：`DOA DAMAGED` + `PARTS DESIGN`

1) 现在的问题

- `PARTS DESIGN` 定义是真正的“结构几何问题”（太高/太窄/缺防倾结构）；
- 上述案例其实是“生产装配缺失/导致到货即不可用”，**本质是 DOA**，不是“设计不合理”。

2) 建议：不新增标签，只补充 DOA 的边界

在 `DOA DAMAGED` 边界里明确写入：

- “若用户描述“某个配件原本应该在出厂时装好/焊好/固定好，但现在是松的/没装/装错，导致开箱即无法正常安装或使用”，
→ 归 **DOA DAMAGED**，不再打 `PARTS DESIGN`。”

`PARTS DESIGN` 保持只关注：

- 形状/高度/宽度/孔位导致杯子/刀叉放不上、太浅、易倒等纯“几何结构”问题。
-

七、**FIT COMPAT** 的“半对样本”：什么时候不该打？

有一条是：

“Mostly fits the sink, but there is still space left open on the sides... I understand

it won't be perfect because this depends on the sink."

被打了 FIT_COMPAT + PARTS DESIGN。

从 MECE 角度看：

- 他自己说 "I understand it won't be perfect" → 说明“侧面有空隙”是可接受范围内的兼容性缺点，且没有说因此导致严重功能问题。
- 真正的痛点在“utensil holder too large to fit well / should be shorter so it stays in place” → 这是很典型的 PARTS DESIGN。

建议在 FIT_COMPAT 边界加一条：

- “若仅为“轻微缝隙/不完美贴合”，且用户主要抱怨集中在某个配件设计上（如餐具篮尺寸不对），
→ 优先打 PARTS DESIGN，不打 FIT_COMPAT，避免重复和泛化。”

总结：要改的 & 要新增的

1. 建议重点修改定义/边界的标签

- INSTALL_FIX：和安装过程/稳定性要重新划清界线
- FIT_COMPAT、SCENE_ADAPT、GUIDE_VARIANT：三者之间的分工要写得更“像决策树”
- CS_POOR vs RETURN_POLICY_BAD：谁管流程、谁管客服，边界要写死
- SOCIAL_GAP：必须出现“社会证明”线索，否则不打
- LOOK_COLOR vs SPEC_MISMATCH：材质/是否不锈钢统一纳入 SPEC_MISMATCH
- PARTS DESIGN vs DOA DAMAGED：出厂装配缺失一律 DOA，不算设计问题
- FIT_COMPAT 的适用条件：弱问题不打，避免到处挂 FIT_COMPAT

2. 建议新增 1 个标签（保持全局 MECE，又不过度拆分）

- INSTALL_COMPLEX （组装复杂/耗时高）
 - 补上“仅抱怨安装过程痛苦”这块空白，避免大家把所有“安装相关抱怨”都塞进 INSTALL_FIX 或 PAGE_MISSING 里。

如果你愿意，下一步我可以帮你把这些改动整理成一份**“打标决策树 + 反例说明”**，方便你直接拿去培训运营同学。



return_dim_tag_20251...
文件

附件是v1版本的标签库，请你在v1的基础上对标签定义和边界进行优化，被改动标签对应的version改成v2，最后生成JSON文件供我下载

已思考 28s >

已思考 1m 31s >

我已经在你给的 v1 标签库基础上，按照前面讨论的 MECE 原则做了这些调整，并生成了一个新的 JSON 文件：

👉 **下载链接：**

[return_dim_tag_v2_20251117.json](#)

本次具体改动一览（均已在 JSON 中落地）

一、新增标签 (version=1)

1. INSTALL_COMPLEX 安装/组装复杂 (耗时高)

- **definition:**

安装/组装步骤繁琐、耗时明显偏长，需额外工具或说明难以理解，用户明确表达“装起来很痛苦/费劲”等感受。

- **boundary_note:**

若已安装完成且主要问题为螺丝/卡扣反复松动或位置锁不住→“安装/调节/固定性差”；仅因页面未说明需要自行组装导致心理落差→“页面信息缺失/误导”。

用于承接“安装过程很烦/很难”这类抱怨，和 INSTALL_FIX 做拆分。

二、修改且 version=2 的标签

以下标签的 `version` 已全部从 **1 改为 2**，并更新了 `definition / boundary_note` 以及 `updated_at`：

1. **FIT_COMPAT 尺寸/兼容性不符**
2. **SCENE_ADAPT 场景适配不足**
3. **GUIDE_VARIANT 变体/选型指引不清**
4. **CS_POOR 品牌/店铺服务不达标**
5. **RETURN_POLICY_BAD 退换信息/政策体验不佳**
6. **SOCIAL_GAP 社会证明与体验不符**
7. **SPEC_MISMATCH 规格参数与宣传不一致**
8. **LOOK_COLOR 外观/颜色与实物不符**
9. **INSTALL_FIX 安装/调节/固定性差**
10. **DOA_DAMAGED 损坏到货 (DOA/受损)**
11. **PARTS DESIGN 分区/配件设计不合理**
12. **PAGE_MISSING 页面信息缺失/误导**

这些改动的核心逻辑是：

- **安装相关：**
 - `INSTALL_COMPLEX` 专门接住“安装过程复杂/耗时高”；
 - `INSTALL_FIX` 只管“装完后还锁不住/总要拧螺丝”；
 - 和 `STABILITY_TIP`、`PAGE_MISSING` 的边界在文案里都写清楚了。
- **适配相关 (FIT_COMPAT / SCENE_ADAPT / GUIDE VARIANT) :**
 - `FIT_COMPAT`: 只要是**尺寸不匹配导致基本放不下/缝隙巨大**就归它，不再强依赖“标称适配范围”这句话；
 - `SCENE_ADAPT`: 强调“尺寸基本 OK，但在 RV/双槽/窄台面等特定场景出现倾斜/回流”；
 - `GUIDE VARIANT`: 必须能看出用户把“选错型号”归因到**页面指引不清**，而不是单纯“我没看清”。
- **服务/退货分工 (CS_POOR vs RETURN_POLICY_BAD) :**
 - `CS_POOR`: 只认客服响应/态度/个案处理；
 - `RETURN_POLICY_BAD`: 流程复杂、必须跑远处 UPS 才能退、规则不友好等，全部归它。
- **预期 vs 社会证明 (SOCIAL_GAP) :**
 - 必须出现“高评分/热卖/大家都说好”这类**社会证明**字眼才打；
 - 仅“not what I expected”不再归这里。
- **外观 vs 规格 (LOOK_COLOR vs SPEC_MISMATCH) :**

- 材质/是否不锈钢/是否塑料等 → SPEC_MISMATCH；
 - 真正的“色差/光泽度/涂层不均”等 → LOOK_COLOR。
 - **设计 vs 出厂缺陷 (PARTS_DESIGN vs DOA_DAMAGED) :**
 - 几何/结构设计问题 → PARTS_DESIGN；
 - 出厂装错/少装导致开箱就不能正常安装/使用 → DOA_DAMAGED。
 - **信息缺失 (PAGE_MISSING) :**
 - 明确把“未说明需要自行组装”写进了定义；
 - 若信息已清晰写出，仅是用户没仔细看 → 不打 PAGE_MISSING。
-

你可以直接下载上面的 `return_dim_tag_v2_20251117.json` 替换到你的系统里使用。

如果后面你想，我也可以基于这套 v2 标签库帮你生成一份**「运营打标训练题 + 答案」**，方便培训新人。 </>



请你重新审视V2版本，整体来看，是否有删减标签的空间

已思考 1m 16s >

有空间，但不需要“大动刀子”。

从现在这套 V2 来看，**整体已经比较 MECE**，但如果您的核心诉求是：

「一线运营更好上手、单条 review 打标更快」，
那确实有几类标签可以**合并/下沉成“衍生维度”**，不必让新人手动选。

我按「推荐保留 / 建议合并或删除」给你一个清单。

一、我建议坚决保留的“主干维度”（不要动）

这些是你现在这套体系里，**高频 + 高价值 + 边界清晰**的标签，不建议删减：

- **适配&场景类**
 - FIT_COMPAT 尺寸/兼容性不符

- SCENE_ADAPT 场景适配不足
- GUIDE_VARIANT 变体/选型指引不清
- **页面&规格类**
 - PAGE_MISSING 页面信息缺失/误导
 - SPEC_MISMATCH 规格参数与宣传不一致
- **安装&使用类**
 - INSTALL_COMPLEX 安装/组装复杂 (耗时高)
 - INSTALL_FIX 安装/调节/固定性差
 - STABILITY_TIP 稳定性/防倾倒不足
 - CAPACITY_EFF 容量/有效面积不足
 - PARTS DESIGN 分区/配件设计不合理
 - CLEAN_MAINT 清洁/维护成本高
- **产品本体质量类**
 - DOA_DAMAGED 损坏到货 (DOA/受损)
 - MATERIAL_RUST 材质/防锈失效
- **物流&售后类**
 - SLA_DELAY 物流时效异常
 - RETURN_POLICY_BAD 退换信息/政策体验不佳
 - CS_POOR 品牌/店铺服务不达标
- **价格&感知价值类**
 - VALUE_WEAK 价值/对比劣势

这些构成了“问题是什么”的主干树，删掉会明显损失分析颗粒度，我不建议动。

二、明确有删减/合并空间的标签（重点看这几个）

1) SOCIAL_GAP 社会证明与体验不符 → 建议并入 VALUE_WEAK

现状：

- 使用条件苛刻：必须同时出现“高评分/热卖/大家都说好” + “体验不如预期”。
- 在实际样本里出现频率不会太高，而且它往往**同时伴随其他真实问题**（如质量差、稳定性差）。

问题：

它本质是一个“情绪 + 信任”维度，而不是独立的“问题类型”。

从运营决策角度看：

- 把它单独做一个标签，
- 和“到底哪里不好”那一堆标签并列，
收益有限，认知负担反而上升。

建议：

- **删除 SOCIAL_GAP 这个标签；**
- 逻辑上并入：
 - 用户觉得被“评分/评论骗了” → 依然靠
 - **根因标签** (如 STABILITY_TIP、MATERIAL_RUST、PAGE_MISSING 等)
 - **价值维度**: VALUE_WEAK 来体现负面体验。

简单说：

“社会证明 ≠ 实际体验”这个信息，可以通过
「高星 + 吐槽 + VALUE_WEAK」
在后续分析阶段算出来，没必要让一线手动打一个专门标签。

2) PRICE_PROMO_OPAQ 价格/促销不透明 → 建议合并到 VALUE_WEAK / RETURN_POLICY_BAD

现状：

- 覆盖“价格波动/买贵了/活动解释不清/价格很套路”这类反馈。
- 实战上，抱怨价格不透明的评论，**几乎总是伴随着“觉得亏/不值”。**

问题：

- 它和 VALUE_WEAK 的边界非常“概念化”，一线打标很难稳定区分：
 - “觉得卖得贵” → VALUE_WEAK
 - “觉得被活动套路了” → PRICE_PROMO_OPAQ
但在用户嘴里往往混在一起。

建议：

- **删除 PRICE_PROMO_OPAQ 作为独立标签；**
- 处理方式：
 - 纯“贵、不值” → 仍然用 VALUE_WEAK
 - 明确指向“退换差价、活动规则坑、退款不到位”等 → 用 RETURN_POLICY_BAD (退换/政策体验)
- 后期如果想看“价格相关问题”，可以通过关键词 (price / deal / discount / coupon / promotion) 和 VALUE_WEAK / RETURN_POLICY_BAD 组合筛出来，无需多一个标签。

3) SURFACE_PROTECT 台面/器具保护不足 → 建议合并进 PARTS DESIGN / DOA DAMAGED

现状：

- 专门描述“没有防滑/防刮脚垫、划伤台面/器具”等。
- 但本质上是**设计或装配问题**的一种表现形式。

问题：

- 与 PARTS DESIGN 、 DOA DAMAGED 高度重叠：
 - 如果本来就该有橡胶脚垫但没有设计 → PARTS DESIGN
 - 如果说明书/图示里有脚垫，但实际到货缺失或装错 → DOA DAMAGED

建议：

- **删除 SURFACE PROTECT 单独标签；**
- 在 PARTS DESIGN 的例子里补一句：
“包含因缺少防滑/防刮脚垫导致划伤台面或器具的情况（设计缺失）；”
- 在 DOA DAMAGED 边界里保留现在这句：
“出厂应预装的胶脚/防滑件未装或装错，开箱即无法按正常形态使用。”

这样既不损失信息，又能让打标决策更简单：

划伤台面？→ 先看是“设计没做”，还是“配件少装/装错”。

三、争议较小但可以视业务情况再简化的标签

下面这几个**不是必须删**，但如果你未来要进一步压缩标签数，可以当作“第二批候选”。

4) WEIGHT_MOVE 自重过大/移动不便

- 对当前这类水槽架产品来说，**出现频率不高**；
- 但在更大件的品类（比如带水箱、带轮、金属家具类）会很有价值。

建议：

- 如果你的标签体系会复用到更多大件品类 → **保留**；
- 如果从战略上只打算服务“轻量级厨卫收纳”等品类 → 可以考虑未来把它合并进一个更大的“使用便利性”类（比如和 CLEAN MAINT 一起，做一个大类维度，在上层分析做拆分，而不是一线打两个小标签）。

5) PACK_WEAK 包装防护不足

- 和 DOA_DAMAGED 的关系是：
 - PACK_WEAK = **保护手段弱**（但商品可能完好）；
 - DOA = **结果坏掉了。**
- 对供应链/仓储优化很有价值。

从 MECE 看，两者是“原因 vs 结果”，并不重合。

建议：

- 如果你有打算把“包装改善”单独作为一个项目（比如换填充物、改纸箱强度）→ **保留**；
- 如果你只是想简化运营打标，而后续诊断就不会深入到“包装 vs 质检”层面 → 可以考虑只留 DOA_DAMAGED，把 PACK_WEAK 作为**后续用文本规则自动判断的“隐形特征”**，不让运营手动选。

四、整体删减建议小结

在不牺牲核心洞察的前提下，我会给你两档方案：

方案 A：轻度精简（推荐）

- **删除 3 个标签：**
 1. SOCIAL_GAP
 2. PRICE_PROMO_OPAQ
 3. SURFACE_PROTECT
- 其它标签保持不变。
- 总数从 23 → **20 个**，一线认知负担下降明显，但维度结构还是很完整。

方案 B：更激进一点（视业务选择）

在方案 A 基础上，再视品类范围决定是否：

- 合并/隐藏 WEIGHT_MOVE（并入“使用便利性”大维度，在后台做文本识别）；
- 合并/隐藏 PACK_WEAK（把包装问题仅通过 DOA_DAMAGED + 关键字分析体现）。

如果你愿意，下一步我可以帮你：

- 按**方案 A**生成一版精简后的标签 JSON（自动调好 version / 边界说明），

- 顺便输出一张「老标签 → 新标签」映射表，方便你做历史数据迁移。



按方案 A生成一版精简后的标签 JSON（自动调好 version / 边界说明），此外，在调整的过程中，可以思考一下标签中文名称与对应定义的表达，目的是让运营人员看到标签名称就大概猜到定义是什么

已思考 28s >

已思考 40s >

已思考 1m 8s >

已思考 31s >

我已经按**方案 A**帮你生成了一版精简后的标签库 JSON，并在调整过程中顺带优化了部分中文标签名，让运营同学一看名字大概就能猜到含义。

👉 下载文件：

[return_dim_tag_v2_slim_20251117.json](#)

本次精简 & 调整概览

一、删除了 3 个标签（不再出现在 JSON 中）

1. SOCIAL_GAP
2. PRICE_PROMO_OPAQ
3. SURFACE_PROTECT

对应的信息，会通过组合其它标签 + 文本分析在后续分析层面还原，不再让一线手动选。

二、中文名称做了“小手术”的标签（更贴近定义）

这些标签的 **tag_code** 不变，只是中文名字更口语直观：

1. PACK_WEAK

- **原名：**包装防护不足
- **新名：**包装太薄/防护不足
- 运营看到就知道是“包装太单薄、容易磕碰”的那种吐槽。

2. VALUE_WEAK

- **原名：**价值/对比劣势
- **新名：**性价比差/不值这个价
- 明确就是“贵”“不划算”“不值”的场景。

3. STABILITY_TIP

- **原名：**稳定性/防倾倒不足
- **新名：**整体稳定性差/易晃动
- 一眼就联想到“晃、软、摇、容易倒”。

其余大部分标签中文名本身已经比较直观，我就没有硬改，比如：

- FIT_COMPAT：尺寸/兼容性不符
- GUIDE_VARIANT：变体/选型指引不清
- SCENE_ADAPT：场景适配不足（槽内/跨槽/台面/RV 等）
- INSTALL_FIX：安装/调节/固定性差
- CAPACITY_EFF：容量/有效面积不足（含扩容效果不达标）等。

三、定义 & 边界更新并升版 (version=2) 的标签

以下这些标签，**定义/边界有改动**，version 已统一从 1 升到 2，方便你在系统里区分 v1/v2：

- FIT_COMPAT 尺寸/兼容性不符
- SCENE_ADAPT 场景适配不足
- GUIDE_VARIANT 变体/选型指引不清
- CS_POOR 品牌/店铺服务不达标
- RETURN_POLICY_BAD 退换信息/政策体验不佳
- SPEC_MISMATCH 规格参数与宣传不一致（已纳入“材质是否不锈钢”等客观规格）
- LOOK_COLOR 外观/颜色与实物不符（只管“看起来不对”，材质不对归 SPEC_MISMATCH）
- INSTALL_FIX 安装/调节/固定性差（明确和“安装复杂”的拆分）

- DOA_DAMAGE 损坏到货 (DOA/受损) (补充了“出厂少装/装错配件”的场景)
- PARTS DESIGN 分区/配件设计不合理 (与 DOA、稳定性拆分更清楚)
- PAGE MISSING 页面信息缺失/误导
- PACK WEAK 包装太薄/防护不足
- VALUE WEAK 性价比差/不值这个价
- STABILITY TIP 整体稳定性差/易晃动

整体思路就是：

- 文字里直接点出**常见话术** (比如“贵/不值/不划算”，“晃、摇、容易倒”，“包装太薄”)
 - 边界里尽量写成“**如果是 A 场景就打 X, 如果是 B 场景就打 Y**”这种运营能直接照抄的决策规则。
-

四、新增标签 (仍为 version=1)

保留之前加的安装维度标签：

- INSTALL_COMPLEX : **安装/组装复杂 (耗时高)**
 - 独立承接“步骤多、很难装、装到崩溃”这类吐槽；
 - 定义和边界里已经写清楚：
 - 只要是“过程很痛苦” → INSTALL_COMPLEX
 - “装好后总要拧螺丝/锁不住” → INSTALL_FIX
 - “根本没说要自己装” → PAGE_MISSING
-

五、保持不变的标签 (version 仍为 1)

这些标签逻辑上已经够清晰，这次没动定义与名称：

- SLA_DELAY 物流时效异常
 - CLEAN_MAINT 清洁/维护成本高
 - MATERIAL_RUST 材质/防锈失效
 - CAPACITY_EFF 容量/有效面积不足 (含扩容效果不达标)
 - WEIGHT_MOVE 自重过大/移动不便
-

如果你后面希望，我可以基于这份 **v2_slim 标签库** 再帮你出一份：

- 每个标签 2-3 条**典型好样本 + 易错反例**

- 外加一张「老标签 → 新标签（含删除/合并逻辑）」映射表

方便你直接拿去做运营打标培训 PPT 或文档。 </>



你觉得“场景适配不足（槽内/跨槽/台面/RV 等）”的标签名称是否可改成更直白点，倾向于“沥水”

已思考 30s >

我也觉得现在这个名字有点“产品经理味儿”，不够一眼秒懂 😅。

但如果直接改成“沥水问题”，会有点缩窄了范围——因为这个标签本来不仅管“水怎么流”，还管“在 RV、双槽、台面这种场景下的使用不顺”。

所以我会这样处理：把“沥水”放到前面当记忆锚点，但保留“场景”两个字，既直白又不丢边界。

建议的改名方案（推荐用第 1 个）

推荐名称：

「沥水/场景不适配（槽内/跨槽/RV 等）」

- 一线看到**“沥水”**：会想到回流水、积水、滴到台面这些典型抱怨；
- 看到**“场景不适配”**：还能 cover 掉
 - RV 里高度不合
 - 双水槽跨不过去
 - 台面太窄导致一边悬空

这类非“水路问题”但也是“场景相关”的吐槽。

你的 SCENE_ADAPT 定义本来就是：

在尺寸基本适配的前提下，在特定场景（双槽、RV、窄台面等）出现倾斜、下坠、回流水等问题

所以用「沥水/场景不适配」刚好把核心关键词都挂在名字里，运营一看就大概知道：

“不是单纯尺寸不合，而是放到某种场景里沥水/姿态出了问题”。

备选两个你可以对比看看

1. 「沥水不顺/场景不适配」

- 更口语一点，但“沥水不顺”略长，UI 上可能有点挤。

2. 「水槽场景不适配（沥水/倾斜等）」

- 强调是“水槽相关场景”，但如果你后面把这套标签复用到非水槽类品（比如台面置物架），就有点局限了。

如果你觉得「沥水/场景不适配（槽内/跨槽/RV 等）」OK，

下一步我可以帮你把 JSON 里 SCENE_ADAPT 的 tag_name_cn 更新成这个版本，并重新导出一版。



我觉得倾斜，下坠问题在其它标签已能体现出来，可以把这个标签的定义收窄为沥水相关的问题

好，这个收窄思路我很赞同：

“场景适配不足”→ 改成只管“水往哪儿流、流得顺不顺”，其它倾斜/下坠类都交给稳定性、尺寸这些标签。

我已经在你上一版精简 JSON 的基础上，完成了以下调整：

1. SCENE_ADAPT 标签改名 & 收窄定义

tag_code 不变： SCENE_ADAPT

- 新中文名称：

👉 「沥水/排水问题」

- 新定义 (definition)：

沥水方向或排水路径设计不佳，导致水流不到位或去向异常，如滴到台面/地面、流不到水槽、易积水不易干等。

- **新边界说明 (boundary_note) :**

若主要问题为架体在特定场景下倾斜、下坠或一侧悬空→优先按“整体稳定性差/易晃动”或“尺寸/兼容性不符”打标；

若仅为局部挡水条/接水盘几何设计导致局部不稳或空间浪费→可考虑“分区/配件设计不合理”；

本标签仅聚焦水路/沥水方向相关问题（如水滴外流、积水难排、滴在台面上等）。

- **version:** 已确保至少为 2，方便你在系统里区分新版定义。
- **updated_at** 也同步更新为当前时间。

这样一来：

- **各种“晃、倾斜、下坠、一边悬空”** → 都去 STABILITY_TIP （整体稳定性差/易晃动）或 FIT_COMPAT；
- **“沥水方向不对、滴到台面、积水不干”** → 一律用 SCENE_ADAPT = 沥水/排水问题，一线看到名字就知道是水路问题。

2. 新文件下载

已经生成了新的 JSON 文件，供你直接使用：



[return_dim_tag_v2_slim_drain_20251117.json](#)

如果你愿意，下一步我可以帮你写一小段“运营使用说明”，专门讲清楚：

- **沥水/排水问题 vs 整体稳定性差/易晃动 vs 尺寸/兼容性不符**
用几句“傻瓜式记忆口令 + 示例句”，方便培训新人。 </>



另外，这套标签体系中，与页面描述不符相关的我觉得可以酌情删除，因为后续我会再做依次页面描述匹配来做评估，

不在这做打标，但是你要区别于期望落差相关的，期望落差还是需要保留的

已思考 38s >

已思考 31s >

已思考 31s >

理解你的思路了：

- “**页面描述是不是对的**” → 以后用独立的页面描述匹配体系来评估（自动化）；
- “**用户的主观预期 vs 实际体验**” → 仍然希望在这套人工标签体系里保留一个维度来抓。

我已经按这个原则，把标签做了一轮结构调整，并生成了一份新的 JSON 文件。

1. 做了哪些结构调整？

① 删除了“页面描述不符”类标签（从打标体系中拿掉）

从当前这版中 移除 了这两个标签（不再给运营打标用）：

1. SPEC_MISMATCH：规格参数与宣传不一致
2. LOOK_COLOR：外观/颜色与实物不符

它们都是典型的“**实物 vs 页面描述/图片不一致**”类型，后续交给你的“页面描述匹配评估”模块来做就好。

注意：只是**不让运营手动打这俩标签**，概念和问题本身你还能通过别的算法去捕获。

② 保留但重写“信息类”标签的角色

保留了这两个：

- PAGE_MISSING 页面信息缺失/误导
- GUIDE_VARIANT 变体/选型指引不清

并重新明确它们的定位：

- **PAGE_MISSING**: 只管“该说的没说清/没说全”（缺关键尺寸、缺安装说明、没讲要自己组装等等）；
 - 新边界里写清楚：
 - 页面数值错误、图片失真这类**客观描述不符** → 不再靠打标，交给页面匹配评估；
 - 用户只是没仔细看/预期太高 → 可考虑打“预期落差”，**不要打 PAGE_MISSING**。
 - **GUIDE_VARIANT**: 只管“多尺寸/多版本下的指引不清”，例如：
 - “哪个尺寸适合我的水槽看不出来”
 - “图片/选项里看不明白差异”等
 - 边界中也删除了对 SPEC_MISMATCH 的依赖，改成：
 - 规格数值错误这类**客观“描述不符”** → 后台页面匹配模块处理。
-

③ 新增一个“预期落差”标签，专门抓用户主观期待

新增标签：

- EXPECT_GAP : 预期落差/没有想象中好

字段设置：

- category_code : CAT_INFO_EXPECT
- category_name_cn : 信息与预期管理
- version : 1 (新标签)

定义 (definition) :

页面信息本身未必错误，但用户主观预期（如尺寸、质感、整体体验等）明显高于实际表现，

在评论中使用“没有想象中那么好/不如预期/Not what I expected/Smaller than I expected”等表达。

边界 (boundary_note) :

- “若能明确指向具体功能/质量缺陷（如不稳、易生锈、容量太小等），应优先打对应功能性标签，可视情况叠加本标签；”
- “单纯认为“贵/不值”且与价格强相关→“性价比差/不值这个价”；”
- “因页面规格错误、图片失真等导致的客观“描述不符”问题，将在后续页面匹配评估中处理，本标签不再区分此类客观错误。”

也就是说：

- 用户说：“**Not what I expected / smaller than I expected / not as nice as I thought**”
 - 没有明显指向“页面写错、图骗人”等 → 打 EXPECT_GAP。
 - 如果他既说“没达到我的预期”，又具体说“很晃、会生锈”
 - 先打稳定性/材质等功能标签，**可选叠加一枚 EXPECT_GAP**。
-

2. 同步调整了其他标签的边界描述（避免提到已删除的标签）

为了不让运营看到“指到一个不存在的标签”，我也顺手改了几处边界说明：

- FIT_COMPAT **尺寸/兼容性不符**
 - 去掉“→ 规格参数与宣传不一致”的引用，改成：
 - 由页面规格错误引起的认知偏差 → 由页面描述评估模块处理，本标签不管这块。
 - GUIDE_VARIANT **变体/选型指引不清**
 - 边界中不再出现 SPEC_MISMATCH；
 - 增加：
 - “仅是自己没仔细看/预期更大，但没怪页面” → 用 FIT_COMPAT 或 EXPECT_GAP。
 - MATERIAL_RUST **材质/防锈失效**
 - 边界中不再提 LOOK_COLOR，改成泛化描述：
 - 单纯色差/喷涂不均等外观问题 → 交给视觉/页面质量评估，不用在这里打。
 - PAGE_MISSING **页面信息缺失/误导**
 - 边界强调：
 - “页面没错，只是用户预期太高或没仔细看” → 打 EXPECT_GAP 不打 PAGE_MISSING；
 - 页面数值/描述错误 → 页面匹配模块处理，不再靠打标。
-

3. 新文件下载

我已经把这些变更写入一份新的 JSON 文件：



[return_dim_tag_v3_expect_20251117.json](#)

这一版的特点是：

- 不再要求运营区分各种“页面描述对不对”；
 - 专注在“实际体验是什么” + “用户主观预期是否落差很大”；
 - 便于后续你用算法独立评估“Listing 文案/图片”这条链路。

如果你愿意，下一步我可以帮你：

- 用几条真实/模拟的评论，示范：
 - 哪些只打功能标签
 - 哪些加上 EXPECT_GAP
 - 哪些原本会打 SPEC_MISM“块” </>

< 2/2 > ...

PAGE_MISSING和GUIDE_VARIANT是否可以整合到一个标签中即可，另外，保留LOOK_COLOR，EXPECT_GAP是否可以当成一个大类放到category_code即可，因为具体是什么不如预期就得看细分标签了，基于这些修改，你可以重新审视调整标签大类的分类以及标签命名

已思考 50s >

已思考 1m 36s >

太好了，这一步基本把体系「收敛成纯问题维度 + 预期靠细分标签」了，我已经按你的思路做完调整，并生成了新 JSON。

👉 新文件下载:

return_dim_tag_v4_mergedInfo_20251117.json

下面是关键变更说明，方便你 review：

1. 合并 PAGE MISSING & GUIDE VARIANT → 一个信息/指引标签

保留 tag_code: PAGE_MISSING

(减少历史数据迁移成本)

- **新中文名称:**

「页面信息/选型指引不清」

- **新定义 (definition) :**

商品页面在尺寸、适配范围、安装要点（如需自行组装）、使用场景或多尺寸/多版本差异等关键信息上缺失、表达不清或难以理解，导致用户对是否适配、安装难度或版本选择产生误判。

- **新边界 (boundary_note) :**

核心规则写成运营可以直接照抄的判定逻辑：

- 只说“**没有想象中那么好/以为会更大**”，没有提「页面/选型看不懂」

→ 按具体功能 + 价格标签处理，**不打这个标签**

- 评论里直接吐槽：

“尺寸对比看不懂 / 列表看不出差别 / 不知道该选哪个尺寸/型号”

→ 打**页面信息/选型指引不清**，不用再单独分变体标签

- 规格数值写错、图片严重失真这类**客观描述不符**

→ 不在打标侧区分，交给你后续的“页面描述匹配评估模块”

- 页面已经写得很清楚，只是用户没看

→ 不打这个标签

- **所属大类 (category) :**

- category_code : 仍用 CAT_INFO_EXPECT

- category_name_cn : 改为 「**信息说明与选型指引**」

2. 删除 EXPECT_GAP 标签 (预期落差不再做成一个单独 tag)

- 删除了 tag: EXPECT_GAP

- 预期落差今后不当成“一个标签”，而是：

- 通过**具体问题标签 + 星级/情绪**来理解

- 或在分析层面通过 star、review 文本、功能标签组合 去识别

换句话说：

运营只打“具体哪里不好”，而不是打一个抽象的“预期落差”。

3. 保留并重构 LOOK_COLOR → 只管「到手实物外观不满意」

根据你“页面描述不符交给后续算法”的要求，我把 LOOK_COLOR 重新加入体系，但完全脱敏于“与页面是否一致”：

新/现 tag_code: LOOK_COLOR

- **中文名称:**

👉 「外观颜色/做工不满意」

- **定义:**

用户对到手实物的颜色、质感、涂层、焊点/焊缝等外观做工不满意，如明显色差、看起来很廉价、涂层不均、焊接粗糙等。

- **边界:**

- 重点是“**到手实物长得丑/看着廉价/做工糙**”，
- 无论页面图片是不是一样，这里都不去判断。
- 耐久性问题：生锈、掉皮、涂层寿命差 → MATERIAL_RUST
- 外观问题只是支撑“这价不值” → 可叠加 VALUE_WEAK
- 与页面图片/描述是否一致 → 交给你的 **页面评估模块** 做，不靠这个标签区分。

- **归类:**

- category_code : CAT_MATERIAL_LOOK
- category_name_cn : **材质与外观**

4. 现在的标签大类 & 命名结构 (v4)

当前这版共有 **18 个标签**，大类和命名已经比较清晰，运营一眼看名字大致就知道干嘛的：

A. 结构与承载 (CAT_STRUCT_LOAD)

- CAPACITY_EFF : 容量/有效面积不足 (含扩容效果不达标)
- PARTS DESIGN : 分区/配件设计不合理 (刀叉篮/杯架等)
- STABILITY_TIP : 整体稳定性差/易晃动
- WEIGHT_MOVE : 自重过大/移动不便

B. 材质与外观 (CAT_MATERIAL_LOOK)

- CLEAN_MAINT : 清洁/维护成本高
- MATERIAL_RUST : 材质/防锈失效
- LOOK_COLOR : 外观颜色/做工不满意

C. 履约与服务 (CAT_FULFILL_SERVICE)

- PACK_WEAK : 包装太薄/防护不足
- DOA_DAMAGED : 损坏到货 (DOA/受损)
- SLA_DELAY : 物流时效异常
- CS_POOR : 品牌/店铺服务不达标

D. 适配与场景 (CAT_FIT_SCENE)

- FIT_COMPAT : 尺寸/兼容性不符
- SCENE_ADAPT : 沥水/排水问题

E. 安装与使用体验 (CAT_INSTALL_USE)

- INSTALL_COMPLEX : 安装/组装复杂 (耗时高)
- INSTALL_FIX : 安装/调节/固定性差

F. 信息说明与选型指引 (CAT_INFO_EXPECT)

- PAGE_MISSING : 页面信息/选型指引不清

(以后如果你想继续细分预期类，也可以把新的预期相关标签挂在这个大类下)

G. 价格与政策 (CAT_PRICE_POLICY)

- RETURN_POLICY_BAD : 退换信息/政策体验不佳
- VALUE_WEAK : 性价比差/不值这个价

5. 整体效果小结

现在这套 v4 体系的特点：

1. 运营端只打“具体问题是什么”

- 不再要求判断“是不是页面写错/图片骗人”，这些交给你的后续模块。

2. 跟“预期相关”的，全部通过细分标签来体现

- 比如：晃 → STABILITY_TIP；生锈 → MATERIAL_RUST；觉得不值 → VALUE_WEAK；信息看不懂 → PAGE_MISSING。

3. 标签名尽量口语化、可视化

- 看名字就有画面感：晃、重、贵、不值、包装太薄、外观不满意、沥水有问题、页面信息/选型指引不清……

再次给你新文件链接，方便直接接入系统：

👉 v4 标签库 JSON:

[return_dim_tag_v4_mergedInfo_20251117.json](#)

如果你接下来准备上模型 / 规则打标，我也可以基于这 18 个标签帮你写一份：

- 每个标签 **关键词/句式示例**（适合做 prompt 或规则）
- 再配一套 **运营训练题（评论 → 正确标签）**，可以直接拿去做培训。 ↵/»

□ ⬆ ⬇ ⬈ ⬉ ...

A,D两大类的大类名称是否可以合并到一起？统一叫什么比较好呢？INSTALL_FIX的标签中文名称是否可以更倾向于使用过程调节起来困难？

已思考 38s >

可以合并，而且你这个直觉很对：A+D 本质上是在描述**“这个架子本身长什么样 & 放上去好不好用”**，拆成两个大类对一线来说有点学术了。

我已经按你的想法做了两件事：

1. 合并 A、D 两个大类

原来：

- A: CAT_STRUCT_LOAD 结构与承载
 - CAPACITY_EFF, PARTS DESIGN, STABILITY_TIP, WEIGHT_MOVE
- D: CAT_FIT_SCENE 适配与场景
 - FIT_COMPAT, SCENE_ADAPT (沥水/排水问题)

✓ 现在统一成一个大类：

- **category_code:** CAT_STRUCT_FIT
- **category_name_cn:** 「结构与摆放/适配体验」

所有原来在 **结构与承载** 和 **适配与场景** 下的标签，都归到这个大类下面，比如：

- CAPACITY_EFF 容量/有效面积不足 (含扩容效果不达标)
- PARTS DESIGN 分区/配件设计不合理 (刀叉篮/杯架等)
- STABILITY_TIP 整体稳定性差/易晃动
- WEIGHT_MOVE 自重过大/移动不便
- FIT_COMPAT 尺寸/兼容性不符
- SCENE_ADAPT 沥水/排水问题

命名逻辑：

- “结构” → 架子本身怎么长、分区/承重/稳定性
- “摆放/适配” → 放在什么水槽/台面/场景下好不好用
- “体验” → 让一线知道这是“使用层面”的问题，而不是售后/价格/页面信息

这样，运营看到这个大类名就能大概知道：

这是关于“架子本体 + 放哪儿 + 怎么放”的问题集合。

2. INSTALL_FIX 中文名称微调，更偏向“使用过程调节不省心”

你提到：

“标签中文名称是否可以更倾向于使用过程调节起来困难？”

在不和 INSTALL_COMPLEX (安装步骤复杂) 混淆的前提下，我把 INSTALL_FIX 改成了一个更贴近你意思、又兼顾原定义的说法：

INSTALL_FIX 新中文名称：

「使用中位置调节/固定不省心」

• 保持的核心含义：

不是“第一次安装很难”，而是：

- 调好了后还老跑位、
- 老要拧螺丝、
- 某些卡扣/支撑调节起来老是锁不牢。

• 与 INSTALL_COMPLEX 的分工会更清晰：

- INSTALL_COMPLEX：装起来很折腾 (步骤多、说明难懂、要找工具)
- INSTALL_FIX：用着不省心 (经常要重新调/紧固，位置不稳，固定不牢)

原来的 definition / boundary 我没有改逻辑，只是同步更新了 updated_at：

- 仍然是：

“按说明装好之后，位置仍易位移、需频繁紧固或关键部件无法可靠锁定，影响使用稳定性。”

3. 最新 JSON 文件

所有这些修改，我已经写进了新版本的标签库 JSON：

👉 下载（已合并大类 + 修改 INSTALL_FIX 名称）：

[return_dim_tag_v5_structfit_20251117.json](#)

如果你接下来希望，我可以给这 18 个标签做一张 「大类-子标签-一句话解释」对照表，方便你拿去做运营培训或当系统内的 hover 提示文案。 </>



就目前的标签体系来说，请你帮我把搭建标签体系的主线逻辑提炼出来，我想在日后其它品类继续复用这个思路

已思考 37s >

我先用一句话总结一下这套体系的“灵魂”：

不是在问“谁的错”，而是在用用户视角，把“具体哪里不好用”拆成若干互斥的小问题，覆盖完整的退货旅程。

你后面换品类，只要抓住这条主线，基本就能复用。

一、三层结构：大框架先定死，再长细节

这套标签体系本质是 3 层结构：

1. 顶层：用户旅程维度

- 商品本体：结构/适配、材质/外观
- 安装 & 使用过程

- 信息说明 & 选型指引
- 履约 & 服务
- 价格 & 退换政策

2. 中层：问题大类 (category_code)

当前大类是：

- CAT_STRUCT_FIX : 结构与摆放/适配体验
- CAT_MATERIAL_LOOK : 材质与外观
- CAT_INSTALL_USE : 安装与使用体验
- CAT_INFO_EXPECT : 信息说明与选型指引
- CAT_FULFILL_SERVICE : 履约与服务
- CAT_PRICE_POLICY : 价格与政策

3. 底层：具体问题标签 (tag_code)

每个 tag 都是用户能说出来的一句抱怨，如：

- “容量太小” → CAPACITY_EFF
- “架子晃/不稳” → STABILITY_TIP
- “调好总跑位” → INSTALL_FIX (现在叫：使用中位置调节/固定不省心)
- “装起来太折腾” → INSTALL_COMPLEX
- “外观颜色/做工不满意” → LOOK_COLOR
- “页面信息/选型指引不清” → PAGE_MISSING
- ...

主线逻辑就是：

先把用户从“看到页面 → 收货 → 使用 → 售后/退货”的路径拆成几个「阶段维度」，再在每个维度下，用用户语言定义「一个个可观察的问题点」。

二、核心设计原则：以后做新类目也沿着这几条

1. 只打“体验问题”，不打“页面对不对”

我们刻意把这两块拆开了：

- **打标体系：**只负责记录
 - 到手的东西是什么样
 - 用起来哪里不顺
 - 服务/物流/价格体验如何
- **页面描述是否准确：**

- 交给你后面单独的“页面描述匹配评估”去做。

所以我们把“**规格参数与宣传不一致 / 图片不符**”这类标签从人工体系里拿掉，避免运营在那边纠结：

“到底是他预期太高，还是页面真写错了？”

之后所有“**描述不符**”由算法/规则判定；
人工标签只关注“**事实上的体验问题**”。

2. 区分“过程痛苦”和“结果不稳”

你在水槽架这个品类里，其实有两种典型抱怨：

- “装起来心很累” → INSTALL_COMPLEX
- “装完了还老要拧螺丝/总跑位” → INSTALL_FIX
- “放上去就晃/一边悬空” → STABILITY_TIP / FIT_COMPAT

我们给每种「不同性质的痛点」一个独立标签，并在边界里写成**决策树**：

- 装之前就痛苦 → 安装复杂
- 装完还要折腾 → 位置调节/固定不省心
- 自身结构软、晃 → 整体稳定性差
- 根本放不下/太大太小 → 尺寸/兼容性不符

以后换品类（比如小家电）：

同样可以沿着这条思路，把「装/调/用」拆开，而不是一个大“安装问题”糊掉。

3. 一切标签都围着“用户话术”命名

你可以看到，我们调整后的中文名都尽量是一句“**吐槽语句的压缩版**”，例如：

- **结构与摆放/适配体验**（大类）
 - 容量/有效面积不足
 - 分区/配件设计不合理
 - 整体稳定性差/易晃动
 - 自重过大/移动不便
 - 尺寸/兼容性不符
 - 沥水/排水问题
- **安装与使用体验**
 - 安装/组装复杂（耗时高）
 - 使用中位置调节/固定不省心

- **材质与外观**

- 材质/防锈失效
- 外观颜色/做工不满意
- 清洁/维护成本高

- **信息说明与选型指引**

- 页面信息/选型指引不清

- **履约与服务**

- 包装太薄/防护不足
- 损坏到货 (DOA/受损)
- 物流时效异常
- 品牌/店铺服务不达标

- **价格与政策**

- 性价比差/不值这个价
- 退换信息/政策体验不佳

以后做新类目，只要问自己一句话：

“这个标签名字能不能直接复读成用户的一句抱怨？”

如果不能，就再口语化一点。

4. MECE 思路：每个问题只属于一个“最小格子”

这套体系里有几个关键拆分，你可以在别的品类照抄：

- **稳定 vs 容量 vs 适配 vs 分区设计：**

- 放不稳、会晃 → 稳定
- 放得下但太小/浪费面积 → 容量
- 根本放不下、太大太小 → 适配
- 明确空间够，但分区/挂件几何奇怪 → 分区设计

- **安装过程 vs 使用过程：**

- 安装步骤多、说明看不懂 → 安装复杂
- 装好后老要调 → 位置调节/固定不省心

- **包装 vs DOA：**

- 包装看着很危险/纸箱软/缓冲少，但东西还能用 → 包装太薄/防护不足
- 东西本体真坏了/少件 → 损坏到货 (DOA)

- **价格 vs 预期 vs 具体问题：**

- 贵/不值/不划算 → 性价比差

- 哪里不好用 → 具体功能标签（容量/稳定/材质/清洁难...）
- 预期相关不单做一个标签，而是靠**功能标签 + 星级 + 文字组合**，在分析层面感知。

这样就避免一个抱怨被打到三个维度里，后面分析会很干净。

三、复用到其他品类的步骤建议

以后你换成别的品类（例如：锅具、小家电、浴室收纳、办公椅...），可以按这个步骤来：

步骤 1：沿用顶层大类不动

直接继承这 6 个维度作为「骨架」：

1. 结构与摆放/适配体验（结构 & 功能表现 & 场景适配）
2. 材质与外观
3. 安装与使用体验
4. 信息说明与选型指引
5. 履约与服务
6. 价格与政策

不管什么品类，用户退货原因 90% 都能归进这六格。

步骤 2：对新类目，先用这 6 格聚类真实退货原因

拿一批这个类目的退货评论，做一个简单的“人工聚类”：

- 属于「本体结构/功能问题」的 → 丢进 结构与摆放/适配体验
- 属于「外观/材质/做工」的 → 丢到 材质与外观
- 属于「安装/使用过程」的 → 丢到 安装与使用体验
- ... 依此类推。

然后看看每个大类里，反复出现的、可清晰命名的痛点有哪些，就形成新的 `tag_code`。

步骤 3：能复用的直接复用，只在必要时新增

- 类似“性价比差”“物流慢”“客服差”“包装太薄”等是**跨品类通用** 的，直接沿用。
- “外观颜色/做工不满意”“清洁难”也是高度通用，可以原样复用。
- 对于每个品类特有的问题，只在**频次够高**时再加新标签，比如：
 - 电器类可能需要新增“噪音大”“异味严重”“发热严重”等；
 - 家具类可能需要“坐感不舒服”“支撑不够/塌陷”等。

步骤 4：为每个新标签写清楚“边界三句话”

每个新标签都写一个 mini 决策树：

1. 这个标签“是什么”（1 句话定义）
2. “不是它而是另一个”的例子（对容易混淆的标签写清这里不归我管）
3. 典型用户话术示例（1-2 个例句）

运营培训的时候，就拿这三句话说清即可。

四、总结一句话的主线逻辑

先用用户旅程定“几个大抽屉”，再用用户吐槽语定义“每个抽屉里的小格子”。

小格子永远描述“事实上的体验问题”，

页面对不对、预期高不高交给别的模块处理。

照着这条主线走，你在别的品类里，只需要替换“小格子”，大框架几乎不用重造。



↳ **主线逻辑就是：** 先把用户从“看到页面 → 收货 → 使用 → 售后/退货”的路径拆成几个「阶段维度」，再在每个维度下，用用户语言定义「一个个可观察的问题点」。

按照这个主线逻辑，目前这套标签体系的映射关系是？

已思考 32s >

好，我们就按你说的这条主线——

「看到页面 → 收货 → 使用 → 售后/退货」

来把现在这套标签体系的映射关系梳理一遍。

0. 先回顾一下当前的大类 & 标签 (v5 版)

大类 (category_code → category_name_cn)

- **CAT_STRUCT_FIT**: 结构与摆放/适配体验
- **CAT_MATERIAL_LOOK**: 材质与外观
- **CAT_INSTALL_USE**: 安装与使用体验

- **CAT_INFO_EXPECT:** 信息说明与选型指引
 - **CAT_FULFILL_SERVICE:** 履约与服务
 - **CAT_PRICE_POLICY:** 价格与政策
-

1. 阶段一：看到页面（决策前）

用户通过页面信息 & 选型做购买决策

对应大类

- **CAT_INFO_EXPECT:** 信息说明与选型指引

对应标签

- **PAGE_MISSING — 页面信息/选型指引不清**

- 页面在尺寸、适配范围、安装要点（是否要自己装）、多尺寸/多版本差异等信息上说得不清楚 / 看不懂，
- 导致用户选错型号 / 误判是否适配 / 误判安装难度。

这个阶段对应的是“**我看了页面再下单**”这一步里，信息是否足够清晰。

真正的“描述不符（规格写错/图片欺骗）”已经从标签里剥离出去，交给后续页面匹配模块。

2. 阶段二：收货（物流 & 包装 & 到货状态）

从发货 → 运输 → 开箱这一段体验

对应大类

- **CAT_FULFILL_SERVICE:** 履约与服务

对应标签

- **SLA_DELAY — 物流时效异常**
 - 到货晚、卡单、跟承诺时效差距大。
- **PACK_WEAK — 包装太薄/防护不足**
 - 箱子很薄、内衬少、看着就容易磕碰；
 - 或已经造成轻微外观损伤但功能还可以用。
- **DOA_DAMAGED — 损坏到货 (DOA/受损)**

- 开箱就发现缺件/变形/坏了/无法正常安装使用。
- 包括本应出厂就装好的关键配件（脚垫、连接件）缺失/装错，导致一开箱就不正常。

这部分是典型的“收货体验”维度，跟产品设计本身分开看。

3. 阶段三：安装 & 使用过程（核心体验）

我把它拆成两个层次：

3.1 产品本体：结构 & 适配 & 使用表现

「东西本身长得怎么样、放哪儿、好不好用」

大类：**CAT_STRUCT_FIT — 结构与摆放/适配体验**

- **CAPACITY_EFF — 容量/有效面积不足（含扩容效果不达标）**
 - 能放的东西比想象中少、空间浪费严重、扩展后有效面积依然不够用。
- **PARTS_DESIGN — 分区/配件设计不合理（刀叉篮/杯架等）**
 - 刀叉篮/杯架高度、宽度、开孔设计不合理，导致放不稳/难拿/压缩空间。
- **STABILITY_TIP — 整体稳定性差/易晃动**
 - 架子整体发软、晃、易倒、一边悬空等。
- **WEIGHT_MOVE — 自重过大/移动不便**
 - 太重、抓不住、从一个位置挪到另一个位置很费劲。
- **FIT_COMPAT — 尺寸/兼容性不符**
 - 跟水槽/台面尺寸不配套：放不下/太小/缝隙太大严重影响使用。
- **SCENE_ADAPT — 沥水/排水问题**
 - 水不往水槽里流，滴到台面/地面，容易积水、干得慢等水路问题。

3.2 安装 & 使用过程中的操作负担

「装的过程 / 用的过程中调节是不是很折腾」

大类：**CAT_INSTALL_USE — 安装与使用体验**

- **INSTALL_COMPLEX — 安装/组装复杂（耗时高）**
 - 步骤多、说明难懂、要找各种工具，用户典型感受是：“装起来太折腾 / 费劲 / 费时间”。

- **INSTALL_FIX — 使用中位置调节/固定不省心**

- 好不容易装好了，结果：
 - 架子总移位
 - 螺丝/卡扣总要重新拧
 - 高度/位置调节不顺滑/锁不牢

这里的主线逻辑是：

- “**过程痛苦** → INSTALL_COMPLEX”
 - “**结果不省心** (老要调) → INSTALL_FIX”
 - “**自身结构不稳** → STABILITY_TIP”
 - “**尺寸根本不合** → FIT_COMPAT”
-

3.3 材质 & 外观 & 清洁

「看上去怎么样、摸起来/洗起来怎么样」

大类：CAT_MATERIAL_LOOK — 材质与外观

- **MATERIAL_RUST — 材质/防锈失效**
 - 用没多久就生锈、掉皮、材质明显不耐用。
- **LOOK_COLOR — 外观颜色/做工不满意**
 - 颜色不好看、显廉价、焊点粗糙、喷涂不均等**主观嫌弃长相/做工的反馈**。
- **CLEAN_MAINT — 清洁/维护成本高**
 - 缝隙多、积水/积垢难清理，擦洗成本高。

这些都是“实际拿在手上 & 日常使用时”才会暴露的体验。

4. 阶段四：售后/退货 & 价格感受

4.1 履约 & 售后体验

大类：CAT_FULFILL_SERVICE — 履约与服务

- **CS_POOR — 品牌/店铺服务不达标**
 - 客服慢、态度差、推诿、不按承诺处理等。

收货时的 DOA/包装/时效也在这个大类里（上面第二段已经列出）。

4.2 价格与退换政策

大类: CAT_PRICE_POLICY — 价格与政策

- VALUE_WEAK — 性价比差/不值这个价
 - 用户明确表达“贵/不值/不划算/不会再买”。
- RETURN_POLICY_BAD — 退换信息/政策体验不佳
 - 退换流程复杂、必须去很远的网点、只支持某快递点、周期长等。

这一段就是用户在「考虑退不退 / 怎么退」时，对钱 & 规则的真实感受。

5. 映射关系总结成一张“路径 → 大类 → 标签”骨架

你以后做其它品类，可以直接照着这个骨架：

1. 看到页面 (决策前)

- 信息说明与选型指引 (CAT_INFO_EXPECT)
 - 页面信息/选型指引不清 (PAGE_MISSING)

2. 收货阶段 (物流 & 包装 & 到货状态)

- 履约与服务 (CAT_FULFILL_SERVICE)
 - 物流时效异常 (SLA_DELAY)
 - 包装太薄/防护不足 (PACK_WEAK)
 - 损坏到货 (DOA_DAMAGED)

3. 安装 & 使用阶段 (核心体验)

- 结构与摆放/适配体验 (CAT_STRUCT_FIT)
 - 容量/有效面积不足 (CAPACITY_EFF)
 - 分区/配件设计不合理 (PARTS DESIGN)
 - 整体稳定性差/易晃动 (STABILITY_TIP)
 - 自重过大/移动不便 (WEIGHT_MOVE)
 - 尺寸/兼容性不符 (FIT_COMPAT)
 - 沥水/排水问题 (SCENE_ADAPT)
- 安装与使用操作体验 (CAT_INSTALL_USE)
 - 安装/组装复杂 (INSTALL_COMPLEX)
 - 使用中位置调节/固定不省心 (INSTALL_FIX)
- 材质与外观 (CAT_MATERIAL_LOOK)
 - 材质/防锈失效 (MATERIAL_RUST)
 - 外观颜色/做工不满意 (LOOK_COLOR)
 - 清洁/维护成本高 (CLEAN_MAINT)

4. 售后/退货 & 价格阶段

- 履约与服务 (CAT_FULFILL_SERVICE)
 - 品牌/店铺服务不达标 (CS_POOR)
 - 价格与政策 (CAT_PRICE_POLICY)
 - 性价比差/不值这个价 (VALUE_WEAK)
 - 退换信息/政策体验不佳 (RETURN_POLICY_BAD)
-

你以后在别的品类，只要：

1. 先确定这四个阶段还成立（基本都会成立），
2. 再看每个阶段下 **哪些标签可沿用、哪些要新增**，

这套主线逻辑就能“像模具一样”复用下去。需要的话，我也可以帮你按这个骨架，画一张可放进 PPT 的「标签体系概览图」。



沥水/排水问题，的tag code SCENE_ADAPT是否需要修改一下，这个不太贴合

已思考 52s >

我分两层跟你说：“**要不要改**”和“**如果改，怎么改最稳妥**”。

1. 要不要改 SCENE_ADAPT 这个 tag_code?

现在这个标签的中文名和定义已经收窄成：

- **中文名：**沥水/排水问题
- **含义：**水往哪儿流、流得顺不顺（滴台面、积水、流不到水槽）

SCENE_ADAPT 这个 code 对应的是“场景适配”，确实已经**和现在语义不贴了**。

从两个角度看：

1. 工程 & 数据稳定角度：

- tag_code 一般当作“主键 / ID”，改名会涉及：

- 历史数据迁移
- 报表、埋点、模型特征等的映射
- 所以**不建议“直接改 code”**，除非你确定相关链路都能一起改。

2. 长期语义 & 可复用角度：

- 你未来打算这套体系**跨品类复用**，tag_code 如果语义歪了，会让后续人接手很痛苦。
- “**沥水/排水问题**”跟 SCENE_ADAPT 的词根已经没什么关系了，从**规范化视角**确实值得修一修。

我的建议是：

**不要在原记录上硬改 code，
而是走“新增一个语义干净的 code + 废弃老 code”的路径。**

2. 建议的新 tag_code 以及迁移方式

2.1 推荐的新 tag_code

参考你现有命名风格 (CAPACITY_EFF, MATERIAL_RUST, PACK_WEAK 这种“名词+弱点”的结构），我会推荐：

- DRAIN_FLOW
 - 英文含义：Drainage flow (排水流向/排水表现)
 - 对应中文名：**沥水/排水问题** (保持不变)

如果你更想强调“外漏”，可以选 DRAIN_LEAK，
但考虑到你这个标签还管“积水不干”， DRAIN_FLOW 覆盖面更大。

2.2 迁移建议 (最稳的一条路)

1. 新增一条标签：

- tag_code : DRAIN_FLOW
- tag_name_cn : 沥水/排水问题
- category_code : 仍然挂在 CAT_STRUCT_FIT (结构与摆放/适配体验)
- definition & boundary_note : 直接沿用现在 SCENE_ADAPT 的文案

2. 将 SCENE_ADAPT 标记为历史标签：

- 保留它在表里，但：
 - is_active : 设为 0

- `effective_to` : 填一个截止日期
- `definition` 里可以补一句：
“已由 DRAIN_FLOW (沥水/排水问题) 替代，仅用于历史数据兼容。”

3. 数据/系统侧的做法：

- 历史数据里：
 - 可以在报表层做一个简单映射： SCENE_ADAPT → DRAIN_FLOW
- 新的数据流：
 - 运营界面只展示 DRAIN_FLOW，不再展示 SCENE_ADAPT
- 模型或规则：
 - 新规则/模型全部产出 DRAIN_FLOW，
 - 老数据在离线分析时统一映射，不影响线上逻辑。

这样你既能保证：

- 新体系的 `tag_code` 与含义对齐、跨品类更“干净”，
 - 又不会因为硬改 code 把历史数据搞乱。
-

3. INSTALL_FIX 中文名这次改得也对了

你前面希望它更侧重“使用过程中调节困难”，现在我们改成：

- `INSTALL_FIX` 中文名：「使用中位置调节/固定不省心」

这跟 `DRAIN_FLOW` (沥水问题) 一样，都走的是：

code 倾技术 / 稳定，
中文名极度贴用户话术。

所以整体风格上是统一的。

小结

- 是的，从语义角度看，`SCENE_ADAPT` 这个 code 已经不贴现在“沥水/排水”的含义了，值得调整。
- 但不要直接改 code，而是：
 - 新增 `DRAIN_FLOW` (沥水/排水问题)，
 - 将 `SCENE_ADAPT` 标记为历史兼容标签。

如果你愿意，我下一步可以帮你给出一个具体的 JSON 片段示例（包含新 DRAIN_FLOW + 旧 SCENE_ADAPT 标记为 is_active=0 的写法），你可以直接拷进你现有的标签库里。



你觉得可以把性价比差/不值这个价归到哪个大类呢？我把退换信息/政策体验不佳归属到了履约与服务，所以不想就价格自己再创一个大类

已思考 1m 6s >

我直接先给结论：

我建议把「性价比差/不值这个价」归到 CAT_INFO_EXPECT 这个大类下面，并把这个大类改名为「预期与价值感知」，这样就不用单独搞一个“价格”大类，又在逻辑上是说得通的。

1. 为什么不建议塞进其他大类？

先看几个候选，看起来可以塞，但其实都别扭：

- **结构与摆放/适配体验 (CAT_STRUCT_FIT)**

这里是“东西本身怎么放、稳不稳、容量够不够”这种**客观功能维度**，“不值这个价”显然不是结构问题。

- **材质与外观 (CAT_MATERIAL_LOOK)**

虽然“看起来廉价”会影响性价比感受，但 VALUE_WEAK 本身是一个**综合判断 (价格 × 功能 × 品质)**，不能变成“外观子问题”。

- **安装与使用体验 (CAT_INSTALL_USE)**

这类是“装起来累不累、用着调不调得动”，也还是具体问题维度。

- **履约与服务 (CAT_FULFILL_SERVICE)**

你已经把「退换信息/政策体验不佳」放这里了，再把“性价比差”塞进来，会让这个大类同时承担“服务体验 + 价值感”，过于混合。

所以：这些大类都更适合放**客观/具体问题**，

而「**不值这个价**」是用户站在高处，对整体体验做的**价值判断**，横跨所有问题维度。

2. 把 VALUE_WEAK 放在哪个大类最顺？

在你现有的大类里，有一个天生就带“心理/预期”味道的：

- **CAT_INFO_EXPECT (现在叫：信息说明与选型指引)**
 - 下面是： PAGE_MISSING 页面信息/选型指引|不清

如果我们把这类大类理解为：

“用户脑子里的预期是怎么形成的 & 最终觉得值不值”

那就很自然可以挂两种东西在一起：

1. 预期形成阶段：

- 页面信息/选型指引不清 → 导致预期错误或选型错误

2. 预期兑现 & 价值感阶段：

- 性价比差/不值这个价 → 体验完以后，对「付出的钱 vs 得到的东西」的综合判断

所以推荐做法

- 保留 CAT_INFO_EXPECT 这个 category_code 不变（方便技术上兼容）
- 但把 category_name_cn 改成类似：

「预期与价值感知」

或「预期管理与价值感知」

- 这个大类下包括：
 - PAGE_MISSING：页面信息/选型指引不清（影响预期形成）
 - VALUE_WEAK：性价比差/不值这个价（预期兑现后的价值感判断）

这样有几个好处：

1. 不用额外新建“价格大类”（满足你不想为价格单独创 category 的诉求）。

2. 概念上成立：

- 页面说明 & 选型指引 → 决定用户“预期什么”
 - 性价比感受 → 决定用户“最后觉得值不值”
- 都是同一条“**预期-体验-价值感**”链路上的点。

3. 分析时也好用：

- 想看“预期管理出了什么问题” → 看 this 大类整体
- 想拆细：信息没说清 vs 觉得不值这个价 → 看 PAGE_MISSING / VALUE_WEAK 各自占比。

3. 映射关系 (调整后的简表)

如果按这个思路，你的映射会变成：

- **CAT_INFO_EXPECT — 预期与价值感知**
 - PAGE_MISSING：页面信息/选型指引不清
 - VALUE_WEAK：性价比差/不值这个价
- **CAT_FULFILL_SERVICE — 履约与服务**
 - RETURN_POLICY_BAD：退换信息/政策体验不佳
 - SLA_DELAY / PACK_WEAK / DOA_DAMAGED / CS_POOR ... (原来那一堆)

其它大类不变。

如果你觉得「预期与价值感知」这个名字 OK，下一步你要是需要，我可以按这个映射帮你改一版 JSON 的大类中文名 + VALUE_WEAK 的归属，让你直接落库用。



return_dim_tag_v2_202...

文件

重新审视一遍tag code和标签中文之间是否贴合

已思考 56s >

我帮你把整套 tag_code 和中文标签名又对了一遍，结论是：

整体是贴合的，运营看到中文就能理解，code 作为“技术 ID”也基本能看出语义。
真正有一点点偏差的只有 3~4 个，可以视你“强迫症等级”选择是否要再抠。

return_dim_tag_v2_20251117

我按「完全贴合」vs「轻微偏差」来拆给你。

一、整体非常贴合的一批 (可以直接认定 OK)

大类层面

- CAT_STRUCT_FIT → **产品结构/适配体验**
- CAT_MATERIAL_LOOK → **材质与外观**
- CAT_INSTALL_USE → **安装与使用体验**
- CAT_INFO_EXPECT → **预期与价值感知**
- CAT_FULFILL_SERVICE → **履约与服务**

code 和中文大类名是一一对应、含义也对得上的。 □ return_dim_tag_v2_20251117

标签层面

这些 code 与中文名+定义基本是“直译/近义”关系：

- WEIGHT_MOVE → **自重过大/移动不便**
- PARTS DESIGN → **分区/配件设计不合理 (刀叉篮/杯架等)**
- FIT_COMPAT → **尺寸/兼容性不符**
- DRAIN_FLOW → **沥水/排水问题**
- CAPACITY_EFF → **容量/有效面积不足 (含扩容效果不达标)**
- MATERIAL_RUST → **材质/防锈失效**
- CLEAN_MAINT → **清洁/维护成本高**
- INSTALL_COMPLEX → **安装/组装复杂 (耗时高)**
- VALUE_WEAK → **性价比差/不值这个价**
- PAGE_MISSING → **页面信息/选型指引不清 (结合定义，现在是“缺失+不清楚”的合并标签)**
- SLA_DELAY → **物流时效异常**
- RETURN_POLICY_BAD → **退换信息/政策体验不佳**
- PACK_WEAK → **包装太薄/防护不足**
- DOA_DAMAGED → **损坏到货 (DOA/受损)**
- CS_POOR → **品牌/店铺服务不达标** □ return_dim_tag_v2_20251117

这些都属于 code 看英文大概是啥，中文一眼也能懂 的状态，我不建议再动。

二、有轻微语义差异，但问题不大的一批

1) STABILITY_TIP → 整体稳定性差/易晃动

- code: **TIP** 明显是 “tipping (倾倒)” 的缩写，偏重「倒/翻」；
- 中文名和定义：已经扩展到

“下弯、晃动、位移、杯盘易倒”等整体稳定性问题。 return_dim_tag_v2_20251117

评价：

- 对技术同学：看到 STABILITY_TIP 会理解成“稳定性/防倾倒”，和现在的中文名“整体稳定性差/易晃动”是一个家族的词，不至于跑偏。
- 对运营：完全不看 code，只看中文，不受影响。

是否有必要改 code？

- 如果你追求“极致对齐”，可以在将来新版本里把 code 改成 STABILITY_WEAK 或 STABILITY_ISSUE；
 - 但这会牵涉历史数据迁移，我个人建议：**保持 code 不动，只把 code 的英文解释在内部文档里写清楚即可。**
-

2) LOOK_COLOR → 外观颜色/做工不满意

- code：从字面看是 **Look + Color**，偏「视觉效果」。
- 中文 + 定义：已经涵盖“颜色、质感、涂层、焊点/焊缝等外观做工不满意”，把“做工粗糙”也包含进来了。 return_dim_tag_v2_20251117

评价：

- 从用户视角：“外观颜色/做工不满意”是一个非常自然的吐槽合并；
- 从 code 视角：**LOOK** 本身也可以含「做工看上去很糙」的意思，不算明显错位。

要不要动？

- 要求非常严的话，可以把 code 改成 LOOK_QUALITY 之类；
 - 但目前这个轻微差异，我觉得完全在可接受范围内，而且中文名现在对运营是最友好的，就保持现状更好。
-

3) INSTALL_FIX → 使用中位置调节/固定不省心

- code：INSTALL_FIX 原始含义类似 “installation fixity / 固定性”；
- 中文名：强调的是

“使用中位置调节/固定不省心”——即**装完之后还老要调、老要拧螺丝**。

return_dim_tag_v2_20251117

评价：

- 语义上还是一个方向：**安装后的固定性问题**；
- 中文名更贴运营和用户口语，这是我们刻意做的强化。

是否需要改 code?

- 只要你们内部文档明确：`INSTALL_FIX` = “in-use fixity / post-install stability adjustments”，不会有人理解成“维修”，就没什么问题；
 - 同样，如果将来有大版本升级，可以考虑换成`FIX_UNSTABLE` 或类似更明显的词，但现在没有硬性必要。
-

4) PAGE_MISSING → 页面信息/选型指引不清

- code：**MISSING** 更偏“缺失”；
- 中文名+定义：把“缺、说不清、看不懂”都合在了一起，说明是个**“信息不完整 + 不好懂”的综合标签。** return_dim_tag_v2_20251117

评价：

- 若从纯语义洁癖角度，`PAGE_MISSING` 更适合只表示“缺少信息”；
- 但你现在的业务决策是：
 - 不打算再拆一个“`PAGE_CONFUSING`”之类的小标签，
 - 所以用一个 code 承载“缺 & 不清楚”，中文明确写全。

要不要动？

- 如果未来其他品类也沿用“缺 + 不清”合并思路，可以暂时认定`PAGE_MISSING` 的含义就是“信息不完整/不明确”；
 - 真要极致对齐，可以换成`PAGE_INFO_WEAK`，但同样会牵涉历史数据，我不会建议现在改。
-

三、整体结论 & 建议

1. 主流标签（超过 80%）是高度贴合的：

- 尤其是：`FIT_COMPAT` / `DRAIN_FLOW` / `CAPACITY_EFF` / `MATERIAL_RUST` / `INSTALL_COMPLEX` / `VALUE_WEAK` / `DOA_DAMAGED` / `CS_POOR` 这几个，code 和中文基本是“带点缩写的直译”，语义几乎 100% 对得上。

return_dim_tag_v2_20251117

2. 有轻微语义差异的几个 (`STABILITY_TIP`, `LOOK_COLOR`, `INSTALL_FIX`, `PAGE_MISSING`)，都是中文更贴运营 & 用户话术，code 稍偏抽象。

3. 在当前阶段，我的建议是：

- **tag_code 暂时不动：**

- 作为内部 ID 已经比较规整，改动代价大；
- 中文名已经很好地服务了运营，不建议为了和 code 对齐再“变更得更生硬”；
- 真要对齐，可以在你的内部文档里，给每个 tag_code 附一句**英文长名称/解释**，比如：
 - STABILITY_TIP → *Overall stability / tipping issue*
 - LOOK_COLOR → *Appearance color & visible workmanship*
 - INSTALL_FIX → *Post-install position fixing issues*
 - PAGE_MISSING → *Missing or unclear key info on page*

这样一来：

- 对技术 & 数据同学：看 code + 英文解释就懂；
- 对运营：看中文名称就懂；
- 对你自己：结构干净、迁移成本低，以后复用到其他品类也很顺。

如果你愿意，下一步我可以直接帮你生成一份 “**tag_code ↔ 中文名 ↔ 英文解释 ↔ 一句话场景**” 的映射表，用来当内部 wiki 或培训手册的底稿。

来源



I like the one this replaced better. I have to use the smaller setting and there does not appear to be a good understanding of being able to maximize space. It just feels like a small version of a bigger one. The bowls, plates and other items do not sit well in the drain area. Was somewhat disappointed.
应该归属到哪里？

已思考 22s >

就按我们现在这版标签体系，这条评论我会这么打：

1 主标签：CAPACITY_EFF — 容量/有效面积不足（含扩容效果不达标）

对应句子：

"there does not appear to be a good understanding of being able to maximize space. It just feels like a small version of a bigger one."

这些话的核心就是：

- 空间没用好
 - 实际能用的面积，比心理预期的小
→ 标准的“**容量/有效面积不足/扩容效果不达标**”场景。
-

2 辅标签：PARTS DESIGN — 分区/配件设计不合理（刀叉篮/杯架等）

对应句子：

"The bowls, plates and other items do not sit well in the drain area."

这里是在说：

- 碗、盘子在沥水区“坐不稳、放不好”，
根因更像是槽位/分区/支撑结构设计得不合理，而不是整体晃动。

所以：

- 不是 STABILITY_TIP (整体架子晃)
 - 也不是 SCENE_ADAPT (沥水方向/水路问题)
而是局部几何设计问题 → PARTS DESIGN 更贴。
-

3 不建议打的标签

- FIT_COMPAT (尺寸/兼容性不符)
 - “have to use the smaller setting” 更像是使用习惯/场景选择，并没有说「放不下/太大太小导致不能用」，所以我不会打这个。
 - VALUE_WEAK (性价比差/不值这个价)
 - 虽然最后一句 “Was somewhat disappointed” 有情绪，但没提价格/贵/不值，
所以不单独打价值感标签。
-

总结：

这条评论推荐打：

- CAPACITY_EFF 容量/有效面积不足（主）
- PARTS_DESIGN 分区/配件设计不合理（辅）



你觉得为什么deepseek会把它归属收到了DRAIN_FLOW

已思考 13s >

我也猜了一下 deepseek 的脑回路，其实还挺“可以理解但不合规”的 😅。大概率是这几个原因叠加：

1. 关键词误导：看到“drain area”就条件反射成沥水问题

原句里有一句：

*“The bowls, plates and other items do not sit well in the **drain area**.”*

对一个模型来说，如果标签里有一个叫「沥水/排水问题（DRAIN_FLOW）」的维度，它很可能做了词面匹配：

- 用户说了 **drain area**
- 标签有 **drain / 沥水 / 排水**
→ 推断：“这是在吐槽排水区有问题，所以选 DRAIN_FLOW。”

但我们现在对 DRAIN_FLOW 的定义是：

水往哪儿流、流不流得出去 —— 滴台面、积水、流不到水槽、干得很慢等等。

这条评论里其实完全**没有提水流向/积水**，只是说“碗盘在沥水区坐不稳”，根因更像是：

- 槽位几何设计不好（PARTS_DESIGN）
- 空间利用差（CAPACITY_EFF）

也就是说，deepseek 很可能是被“**drain**”这个词给骗了，没真正理解用户是嫌弃**“放东西的结构设计”**，而不是嫌弃“水往哪流”。

2. 旧语义残留：SCENE_ADAPT 原来就叫“场景适配”，涵盖面比较广

你回想一下我们一开始的体系：

- SCENE_ADAPT 最初叫“**场景适配不足**”，定义里面既有
 - 双槽/台面/RV 场景
 - 也有 **沥水、倾斜、下坠** 这一堆东西混在一起。

后来我们才把它收窄成**纯沥水/排水问题**，并计划用 DRAIN_FLOW 替代。

deepseek 如果是拿**旧版定义/旧版训练数据**来的，那它心中的这个标签含义可能还是：

“跟水槽/沥水区相关的问题”，
而不是我们现在这种很严格的“只管水路”的 DRAIN_FLOW。

在这种语境下，它看到：

“items do not sit well in the drain area”

就会觉得：“啊，这是**沥水区场景适配问题**”，于是打成 SCENE_ADAPT / DRAIN_FLOW 也就不难理解了。

3. 模型在多标签场景里的“保守策略”——宁可多打一个“相关”标签

很多模型在多标签分类时有一个倾向：

只要提到跟这个标签“看上去有点关系”的词，就倾向于打上去，以免漏标。

在你的这条评论里，模型可能的内部逻辑是：

- 已经识别出：
 - “空间没用好” → CAPACITY_EFF
 - “碗盘在 drain area 放得不舒服” → 有结构/分区问题
- 看到 “drain area”，并联想到 “沥水/排水问题”
→ 就顺手把 DRAIN_FLOW 也打上了。

从模型角度，这样**Recall (召回)** 高；

但从运营角度，这是我们现在在用标签边界精修时要**刻意修正**的地方。

4. 怎么在后续训练里帮它“改掉这个习惯”

如果你想在后面的模型训练/规则优化里减少这种误判，可以有几条策略：

1. 在 DRAIN_FLOW 的负样本里，专门放类似这条的案例：

- 文本里出现 “drain / drain area / drying area”，
- 但用户抱怨的是“碗盘坐不稳、容量小、几何设计怪”，
- 标签只给： CAPACITY_EFF + PARTS DESIGN，**不打 DRAIN_FLOW。**

2. 在标注指南里写“反例规则”：

- 明写：

“仅出现 drain/drying 相关词不代表沥水问题；必须出现水往哪儿流、滴到哪儿、积水/不易干等水路线索，才打 DRAIN_FLOW。”

3. 在模型层加一点规则兜底：

- 如果句子包含 “do not sit well / can't stand / difficult to place” 等结构摆放词，而没有 “drip / leak / water / dry / wet / pooling” 等水相关词，
→ 优先考虑 PARTS DESIGN / STABILITY_TIP，禁止单独触发 DRAIN_FLOW。

小结一句话

deepseek 把这条归到 DRAIN_FLOW，
多半是因为 看到了 “drain area” 这个词，又继承了旧的 SCENE_ADAPT 语义，
把“沥水区的结构问题”混淆成了“排水问题”。

从业务视角，我们更希望：

- **结构问题** → PARTS DESIGN + CAPACITY_EFF
- **水路问题** → DRAIN_FLOW

你这套新标签边界其实就是在帮模型「戒掉这种词面联想」，逼它回到**真实问题本身**。



下面是我给到deepseek的提示词，如果是你，你会在哪里多加一句，杜绝只做词面匹配呢：

【角色】

你是一名亚马逊美国站的退货分析专家，负责解读用户留言并输出结构化结果。

【任务】

阅读传入的 review 信息和标签库 (tag_library) , 根据标签定义与边界对留言进行打标; 留言可能涉及多个吐槽点, 因此可能需要输出多个标签。

【要求】

1. 输出的 JSON 必须只包含以下字段, 含义如下:
 - review_id: 字符串, 对应留言 ID
 - review_source: 数字, 留言来源 (0=return, 1=voc, 2=review)
 - review_en: 英文留言原文
 - review_cn: 中文翻译, 若无法翻译可返回空字符串
 - sentiment: 情感标签 (-1=负面, 0=中性, 1=正面)
 - tags: 数组。每个元素包含:
 - * tag_code: 标签编码
 - * tag_name_cn: 标签中文名
 - * evidence: 英文原文中触发该标签的句子/短语
2. tag_code, tag_name_cn 都只能来自下方的 tag_library, 若找不到匹配标签则返回空数组。
3. evidence 需为英文原文中触发该标签的简短片段。
4. 禁止返回多余字段或解释性文本, 严格遵循 JSON Schema。

已思考 27s >

我会在【任务】那一段后面直接补一句总则, 把“不能只看关键词”写死成硬规则, 这样所有标签 (不只是 DRAIN_FLOW) 都会受到约束。

你原文是:

text

复制代码

【任务】

阅读传入的 review 信息和标签库 (tag_library) , 根据标签定义与边界对留言进行打标; 留言可

我会改成 (新增那一句我给你标出来) :

text

复制代码

【任务】

阅读传入的 `review` 信息和标签库 (`tag_library`)，根据标签定义与边界对留言进行打标；留言可在选择标签时必须基于语义理解和标签的“**定义/边界**”来判断问题类型，而不能仅凭是否出现了与标签

这样几个好处：

- **位置对**：紧跟在“根据标签定义与边界打标”后面，逻辑连贯，也提醒模型“先看定义，再看词”。
- **语义上管全局**：不仅仅是 `DRAIN_FLOW`，所有会被词面误导的标签都会被这句话约束。
- **给了一个明确反例**：专门点名 “`drain area → DRAIN_FLOW`” 这一类错误，让模型知道这种词面联想是禁止的。

如果你想更简洁一点，也可以用短版：

text

□ 复制代码

在选择标签时必须基于语义理解和标签的“**定义/边界**”来判断问题类型，而不是做简单的关键词/标签

然后专门在你给 `deepseek` 的 **tag_library** 开头再加一句总说明，例如：

说明：不要仅凭评论中出现与标签名称相似的单词就打标签，必须确认评论描述的问题确实符合该标签的“**定义**”和“**边界**”。

两处一起用，基本能把“只看 `drain` 看到 `DRAIN_FLOW`”这类词面匹配错误压下去不少。



下面是亚马逊平台给的退货标签，哪些标签其实意义不大呢？如果没有额外留言的情况下：

APPAREL_STYLE
APPAREL_TOO_LARGE
APPAREL_TOO_SMALL
CR-TOO_SMALL
DAMAGED_BY_CARRIER
DAMAGED_BY_FC
DEFECTIVE
DID_NOT_LIKE_COLOR

DID_NOT_LIKE_FABRIC
EXTRA_ITEM
FOUND_BETTER_PRICE
MISORDERED
MISSSED_ESTIMATED_DELIVERY
MISSING_PARTS
NEVER_ARRIVED
NOT_AS_DESCRIBED
NOT_COMPATIBLE
NO_REASON_GIVEN
ORDERED_WRONG_ITEM
POOR_FIT
QUALITY_UNACCEPTABLE
SWITCHEROO
UNAUTHORIZED_PURCHASE
UNDELIVERABLE_CARRIER_MISS_SORTED
UNDELIVERABLE_FAILED_DELIVERY_ATTEMPTS
UNDELIVERABLE_INSUFFICIENT_ADDRESS
UNDELIVERABLE_REFUSED
UNDELIVERABLE_UNCLAIMED
UNDELIVERABLE_UNKNOWN
UNWANTED_ITEM
WRONG_SIZE

已思考 1m 47s >

先说结论：

如果只有亚马逊这一个“退货原因 code”，没有任何额外留言，而且你的关注点是「商品&页面优化」而不是「物流/风控」的话，下面这些标签的决策价值其实比较低，可以视为“噪音/用户原因”，只做大类统计就够了。

一、几乎没有可操作价值的（可以当“用户原因/噪音”）

1 完全不知道用户哪里不满意

- NO_REASON_GIVEN
- UNWANTED_ITEM

“没说原因”和“不想要了”，本质都是 **用户后悔/情绪型退货**，

没有任何指向：不是质量问题，也不是尺寸/页面/物流问题。

👉 对商品、页面、运营都几乎没法动作，只能当背景噪音看整体占比。

2 纯选错/下单错误（用户操作失误）

- **MISORDERED**
- **ORDERED_WRONG_ITEM**
- **WRONG_SIZE** (在已经有 APPAREL_TOO_LARGE/SMALL、POOR_FIT 的前提下)

这些都在说：**我自己点错了**。

没说是因为页面引导差，还是用户没看清；

👉 对商品优化、文案优化的直接价值非常有限，更多是「用户教育」或「下单确认流程」层面的事。

3 模糊的 size/fit 备选 code (信息重复)

- **CR-TOO_SMALL**

你已经有：

- APPAREL_TOO_SMALL
- APPAREL_TOO_LARGE
- POOR_FIT
- WRONG_SIZE

再来一个 **CR-TOO_SMALL**，在没有文本的前提下，很难知道它和 APPAREL_TOO_SMALL / WRONG_SIZE 有什么实质差异，基本只是「另一个太小」而已。

👉 实际分析时，完全可以和其他 size/fit 类原因合并统计。

二、更多是平台/风控/异常流程，和商品本身关系不大

从“商品&页面优化”的视角看，这些标签**更属于平台侧/风控侧信号**：

- **EXTRA_ITEM** (多发、发错件，偏履约/仓操作问题)
- **SWITCHEROO** (调包/换货欺诈)
- **UNAUTHORIZED_PURCHASE** (未授权购买，账户/支付风控问题)

这些当然对“风控、仓配流程优化”有价值，但如果你是在做：

- 商品设计
- 详情页优化
- 供应商管理

👉 它们传递的“为什么用户不满意”信息很有限，只能当特殊场景标记，不宜过度解读。

三、物流「未妥投」类型，对商品优化价值有限

这一串都属于「货物没真正到用户手里/没收货就退」：

- **UNDELIVERABLE_CARRIER_MISS_SORTED**
- **UNDELIVERABLE_FAILED_DELIVERY_ATTEMPTS**
- **UNDELIVERABLE_INSUFFICIENT_ADDRESS**
- **UNDELIVERABLE_REFUSED**
- **UNDELIVERABLE_UNCLAIMED**
- **UNDELIVERABLE_UNKNOWN**
- **NEVER_ARRIVED**
- **MISSED_ESTIMATED_DELIVERY**

如果你是在做“端到端体验”或“物流 SLA 管理”，这些信息当然有意义；

但单看商品/页面：

- 用户根本没体验到产品本身
- 所以这些原因对“换供应商/改结构/改文案”帮助都很有限

👉 建议它们作为「物流运营维度」单独看，而不要混在“商品问题”里一起分析。

四、比较尴尬的几类：有一点价值，但需要大颗粒看

1 价格相关

- **FOUND_BETTER_PRICE**

如果你做**定价/竞品监控**，这个 tag 是有价值的；

但在没有任何额外信息（没说是哪里便宜、便宜多少、平台内还是平台外）的情况下：

- 对单品优化：作用很有限
- 更适合用来看「这个 ASIN 在大盘里的价格敏感度」

2 主观喜好但粒度较粗

- APPAREL_STYLE (不喜欢款式)
- DID_NOT_LIKE_COLOR
- DID_NOT_LIKE_FABRIC

这些标签可以大致说明：“这款在审美/触感上卖得不讨喜”，

但如果沒有留言细节：

- 是因为太花？太素？布料太厚？太薄？
- 完全看不出来。

👉 适合做高层判断（比如这条线整体风格不受欢迎），但对「怎么改设计」帮助有限，建议配合你自己的细分标签或文本挖掘使用。

五、相对“高价值”的标签（如果只有 reason code 也值得保留）

对比起来，下面这些即便没有留言，也能直接给你「比较清晰可执行的方向」——这些不建议你放弃：

商品/质量类

- DEFECTIVE (功能性故障)
- QUALITY_UNACCEPTABLE (质量不达标，虽笼统但方向明确)
- MISSING_PARTS (缺件)
- NOT_AS_DESCRIBED (描述不符，虽需后续细化)
- NOT_COMPATIBLE (不兼容，适配问题)

尺寸与合身（服装场景）

- APPAREL_TOO_LARGE
- APPAREL_TOO_SMALL
- POOR_FIT

（配合你自建体系再细分：具体是哪个部位不合身）

供应链/履约类

- DAMAGED_BY_CARRIER
- DAMAGED_BY_FC
- MISSING_PARTS (兼有质检/包装问题)

六、如果用一句话总结

**在没有额外留言的前提下：

真正对“商品和页面优化”价值不大的，是那些：

- “完全看不出问题点 (UNWANTED_ITEM/NO_REASON_GIVEN) ”
- “明确属于用户操作失误 (MISORDERED/ORDERED_WRONG_ITEM/WRONC_SIZE/CR-TOO_SMALL) ”
- “明显是平台/风控标签 (UNAUTHORIZED_PURCHASE/SWITCHEROO/EXTRA_ITEM) ”
- “纯物流未妥投原因 (各种 UNDELIVERABLE_*) **”

这些你可以在内部分析里统一归入：

- 「用户原因/无法行动」
- 「平台/风控原因」
- 「物流未妥投」

真正要做精细分析和优化的，是**产品本身（质量/结构/适配）+ 页面信息 + 价格感知**这几块，再结合你现在搭的那套细颗粒度标签体系。



return_dim_tag_v2_202...
文件

附件是最新一版的标签体系，下面这段留言应该归属到哪里？

Quality is poor

已思考 27s >

就按你这版最新标签体系来说，这句只有 3 个词的留言：

“Quality is poor”

在不做主观猜测的前提下，其实 不应该 打任何一个具体标签，tags 应该是空数组。

return_dim_tag_v2_20251117

原因是：

- 你现在的标签全部是**“具体哪里不好”**：
 - MATERIAL_RUST 要有“生锈/掉皮/涂层问题”；
 - LOOK_COLOR 要有“颜色/做工丑/焊点粗糙”等；
 - CAPACITY_EFF 要有“太小/放不下/浪费空间”；
 - STABILITY_TIP 要有“晃/不稳/会倒”；
 - CLEAN_MAINT 要有“难清理/易积污”等；
 -
- 但 “**Quality is poor**” 只是在说“总体质量不好”，完全没有暴露出是：
 - 材质问题？
 - 做工问题？
 - 稳定性问题？
 - 容量/结构问题？
 - 还是别的？

如果此时硬给它贴上某个标签（比如随便扔到 LOOK_COLOR 或 MATERIAL_RUST），就等于是**在帮用户“脑补”具体问题维度**，会污染后续的分析结论。

实战建议

- **当前体系下：**
 - 这条只打情感 (sentiment = -1) ，
 - tags: [] (无具体问题标签) 。
- **如果你以后很在意这类“只说 overall quality 差”的情况：**

可以考虑在体系里新增一个非常明确的 catch-all，例如：

QUALITY_GENERIC : 整体质量差（用户仅笼统说“质量差/做工差/质量不好”，没有给出具体问题维度时使用）

但前提是**你真有打算专门分析这类“泛质量抱怨”的比例**。

在你现在的设计理念里——**标签只记录“具体问题”而不猜测**——

这句 “Quality is poor” 最正确的做法，就是：**不打标签，只标负面情绪**。

来源



我是通过调用LLM批量进行打标的，如何区分是因为无可
用标签返回空值还是LLM偷懒没返回内容呢

已思考 33s >

这个问题问得很关键 🤝 ——如果不设计好，`tags: []` 既可能是真没标签可用，也可能是模型偷懒，肉眼都分不出来。

我给你一个**实战可落地的方案组合**，从接口设计 + 提示词设计两头抓。

一句总原则

“**无可用标签**”必须变成一个“显式状态”，而不是靠 `tags: []` 这种“顺手就能偷懒”的形式。

只要让「空数组」和「有效决策」混在一起，你就永远区分不了。

方案一（最推荐）：把「无合适标签」做成一个正式标签

1. 在 `tag_library` 里加一个特殊标签

例如：

json

复制代码

```
{  
  "tag_code": "NO_MATCH",  
  "tag_name_cn": "无合适标签（模型明确判断）",  
  "definition": "当 review 内容过于笼统或与标签库定义严重不匹配，无法稳定落到任何一个具体  
  "boundary_note": "仅当认真比对所有标签定义与边界后仍无法找到合适标签时使用；不能作为值  
}
```

这样它和其它标签一样，**也是 `tag_library` 的一部分**，符合你原来的约束（“`tag_code` 只能来自 `tag_library`”）。

2. 修改提示词：不允许 tags 为空数组

在你原来的【要求】上，补两条硬规则。

你现在是：

text

 复制代码

【要求】

...

2. tag_code, tag_name_cn 都只能来自下方的 tag_library, 若找不到匹配标签则返回空数组。

...

我会改成（重点是第 2 条）：

text

 复制代码

【要求】

1. 输出的 JSON 必须只包含以下字段，含义如下：

... (原文保留)

2. tag_code, tag_name_cn 都只能来自下方的 tag_library。

若你认真对照所有标签定义与边界后，认为“确实没有任何一个标签适用”，

则必须返回一个 tags 数组，数组中只有一个元素：

- tag_code="NO_MATCH"
- tag_name_cn="无合适标签（模型明确判断）"

禁止在这种情况下返回空数组。

3. evidence 需为英文原文中触发该标签的简短片段。

若使用 NO_MATCH，则 evidence 需为整句或整段 review（用来回溯确认你确实认真阅读过）。

4. 禁止返回多余字段或解释性文本，严格遵循 JSON Schema。

这样一来：

- “tags: [] 一出现 = 100% 协议违规 / 模型偷懒 / 疏忽”
- ““无可用标签”这个业务语义，被 NO_MATCH 显式表达，而不是用空数组模糊带过。”

你在程序侧可以直接做校验：

- 如果 tags 字段缺失或为空数组 → 直接标为“LLM 输出不合规”，重试/回滚。

- 如果 tags 里只有 NO_MATCH → 视为“模型判断确实没有可用标签”，可以统计 & 质检。

方案二：再加一层「自检」信号（可选增强）

如果你希望再严一点，可以在 schema 里再多加一个字段，比如：

json

 复制代码

```
"tag_decision": "HAS_TAG" | "NO_APPLICABLE_TAG"
```

然后在提示词里写：

- “当你输出的 tags 中包含任何一个具体标签（非 NO_MATCH），必须将 tag_decision 设为 "HAS_TAG"。”
- “当你使用 NO_MATCH 作为唯一标签时，必须将 tag_decision 设为 "NO_APPLICABLE_TAG"。”

这样，你在程序侧可以做交叉校验：

- tag_decision = HAS_TAG 但只有 NO_MATCH → 明显不一致，判为异常
- tag_decision = NO_APPLICABLE_TAG 但 tags 里全是具体标签 → 也不一致

不过这个会改变你原本“只能有这些字段”的约束，要看你愿不愿意扩 schema。如果不只想动 schema，那用**方案一**的 NO_MATCH 标签就够用了。

方案三：规则/二次模型兜底（适配“质量差”这种模糊句）

结合你前面的例子：

“Quality is poor”

这种评论很短、很抽象，我们刚才讨论过：

严格按你现在的标签定义，其实不应该强行归到具体标签，而应该是“无合适标签”。

你可以额外做一个小兜底逻辑：

- 若 sentiment = -1 且文本长度 $\geq N$ （比如 10 个字符），但模型返回了 NO_MATCH
- 把这类样本抽成一个池子做人工抽查，看有没有被“过度保守”的情况

或者反过来：

- 文本里出现了很多强指向词（rust, wobbly, too small, unstable...），但 tags 为空或只有 NO_MATCH → 也可以当可疑情况，送去复查。

这样你可以把两种情况区别开：

- **业务上确实“无合适标签”的**（例如 “Quality is poor”）
- **模型明明有标签可打但没打的**（偷懒/理解错误）

小结

如果你想在技术上严格区别：

- “模型认真看过，发现确实没有合适标签”
- vs “模型没认真标/偷懒导致 tags 空”

最简单、最稳的做法就是：

1. 在 tag_library 里加一个正式标签 NO_MATCH （无合适标签），要求只有这种情况才能用；
2. 在提示词里强制规定：不得返回 tags: []，无匹配时必须用 NO_MATCH；
3. 程序侧把 tags: [] 视作协议错误，而不是业务含义。

这样，“业务上的空”和“模型偷懒导致的空”，在数据上就彻底分开了。



↳ 1\. 在 tag\library 里加一个特殊标签

category code 和对应名称呢

已思考 12s >

很好，这个要单独想一层，否则会把“系统兜底”当成真实用户问题分析掉。

我会建议把 NO_MATCH 放在一个**单独的大类里**，清楚标成“系统状态，不参与问题分析”。

建议的 category_code 和名称

json

复制代码

```
"category_code": "CAT_META",
"category_name_cn": "系统打标状态（非用户问题）"
```

含义：

- **CAT_META**: 所有不是“用户真实问题维度”，而是“打标系统本身的状态/兜底”的标签，都挂在这里；
- **系统打标状态（非用户问题）**：一眼告诉分析同学：
👉 这些不能当作产品/页面/服务问题去解读，做分析时要么单独看，要么直接过滤掉。

配套的 NO_MATCH 示例（你可以直接抄进 tag_library）

json

 复制代码

```
{
  "tag_code": "NO_MATCH",
  "tag_name_cn": "无合适标签（模型明确判断）",
  "category_code": "CAT_META",
  "category_name_cn": "系统打标状态（非用户问题）",
  "level": 2,
  "definition": "当认真比对所有标签定义与边界后，仍无法稳定判断该评论属于任何一个具体问题是，仅在确认没有任何一个具体标签适用时才能使用；不得将其作为偷懒或“不想”",
  "boundary_note": "仅在确认没有任何一个具体标签适用时才能使用；不得将其作为偷懒或“不想”",
  "is_active": 1,
  "version": 1,
  "effective_from": "2025-11-17",
  "effective_to": null,
  "created_at": "2025-11-17 00:00:00",
  "updated_at": "2025-11-17 00:00:00"
}
```

这样你在分析层就可以很简单地写规则：

- WHERE category_code != 'CAT_META' → 只看真实问题标签
- WHERE tag_code = 'NO_MATCH' → 看当前标签体系的覆盖盲区/模型保守区间。



请你评估一下，下面这些留言是否都归属到NO MATCH:

ORDERED_WRONG_ITEM: Mistake

UNWANTED_ITEM: No longer need

ORDERED_WRONG_ITEM: No issues

UNWANTED_ITEM: Ordered three by mistake, opened one and did not like it

ORDERED_WRONG_ITEM: Not the right one

UNWANTED_ITEM: No longer needed

Quality is poor

已思考 18s >

短答案：**按你现在这套“只标具体问题、不脑补”的标签体系，这几条都应该归到 NO_MATCH，没有一个够资格打具体问题标签。**

我逐条帮你拆一下逻辑，方便你以后给标注或模型写指南：

1 ORDERED_WRONG_ITEM: Mistake

用户自己说明：点错了 / 下单失误。

- 没提商品哪里不好
- 也没提页面信息有问题

👉 这是**纯用户操作错误**，在你的问题标签体系里没有对应维度 → NO_MATCH。

2 UNWANTED_ITEM: No longer need

不再需要 / 后悔买了。

- 没说质量、尺寸、功能、外观、价格任何一项
- 👉 典型“**用户需求变了**” → NO_MATCH。
-

3 ORDERED_WRONG_ITEM: No issues

明确写了“**No issues**”。

- 说明商品没问题，就是订错了
 - 👉 也是 **用户原因**，不属于任何产品/页面/服务问题 → NO_MATCH。

4 UNWANTED_ITEM: Ordered three by mistake, opened one and did not like it

订多了，又说“**did not like it**”。

- “**did not like it**”非常笼统：
 - 不知道是外观？材质？尺寸？稳定性？沥水？价格？
- 按你现在的标签定义，每个标签都需要能落到**具体维度**：
 - LOOK_COLOR 要有颜色/做工的指向
 - VALUE_WEAK 要有“贵/不值”的指向
 - 结构类要有晃/小/放不下之类的指向

这里完全没有具体指向，只知道“整体不喜欢”。

👉 在“**不脑补具体问题**”的前提下，仍然应归为 NO_MATCH（或将来如果你加了“整体偏好不明原因”的泛标签再说）。

5 ORDERED_WRONG_ITEM: Not the right one

“不是正确的那个”，但原因是自己点错还是页面误导，不知道。

- 没说“**页面信息不清楚**”
- 也没说“**商品本身有问题**”

👉 在当前体系里，只能当“**下单错误/选错**”，不对应任何产品问题 → NO_MATCH。

6 UNWANTED_ITEM: No longer needed

和第 2 条一样，本质是**需求消失**，商品本身没有信息。

👉 NO_MATCH。

7 Quality is poor

这个我们之前专门讨论过：

- 只说“质量差”，但没说具体是：
 - 会生锈？
 - 做工粗？
 - 容量太小？
 - 不稳？
 - 染色/掉漆？
- 你的标签体系是按“**具体问题维度**”拆的，强行把它塞进某个标签（比如 LOOK_COLOR 或 MATERIAL_RUST）都属于**帮用户脑补**，会污染分析。

👉 在当前版本的设计理念下，这条应该打：

- **情绪：负面**
- **问题标签：** NO_MATCH

小结

在 **没有额外留言细节，且你坚持“只打具体问题，不靠主观猜”的前提下**，这 7 条全部应该归为 NO_MATCH，而不是随便扔进某个具体问题标签。

如果你未来发现类似“Quality is poor”“did not like it”占比很高，可以考虑新增一个通用标签，比如 QUALITY_GENERIC / “**整体质量/体验差（未说明具体问题）**”，但那就是下一阶段的设计了。当前这版体系，标 NO_MATCH 是最干净、最一致的做法。



请你评估一下，下面这条留言应该归属到哪？

UNWANTED_ITEM: This one size didn't fit either. I had no room to put the utensils holder

这条其实非常清晰，该归到你体系里的：

FIT_COMPAT — 尺寸/兼容性不符

原文：

UNWANTED_ITEM: This one size didn't fit either. I had no room to put the utensils holder

关键信息：

- “size didn't fit” → 尺寸不合
- “I had no room to put the utensils holder” → 目标位置（台面/水槽旁）空间不够，**没地方放**这个架子或附带的餐具筒

按照我们给 FIT_COMPAT 的定义：

与目标位置/设备尺寸不匹配，导致 **无法放入、放不开、没地方放** 或间隙过大影响使用

正好完全命中这一条，所以应该标：

- **主标签：FIT_COMPAT 尺寸/兼容性不符**

不需要上 NO_MATCH，也不需要打容量类 (CAPACITY_EFF)，因为这里说的是**厨房空间不够放这套东西**，而不是「架子本身能放的东西太少」。



你觉得deepseek为什么同时打了FIT_COMPAT和CAPACITY_EFF

已思考 15s >

我先说结论：

deepseek 同时打了 **FIT_COMPAT + CAPACITY_EFF**，本质上是被一句“**no room to put the utensils holder**”搞晕了，一半往「环境空间不够」理解，一半往「产品容量不够」理解，再加上你允许多个标签，它就干脆两个都打了。

你这条原句是：

This one size didn't fit either. I had no room to put the utensils holder.

按我们现在的定义，应该是 **只打 FIT_COMPAT** (尺寸/兼容性不符)，但 deepseek 的可能脑回路大概是这样：

1. 语言歧义：模型对“room”的两个理解

这句里有两个关键信号：

- “This one size didn’t fit either”
- “I had **no room** to put the utensils holder”

人类的直觉是一致的：

→ **厨房/水槽周围环境空间不够放这套东西（含餐具筒）**，属于环境兼容性问题。

但模型可能这么拆：

1. 环境维度理解（对 FIT_COMPAT 有利）

- “size didn’t fit” + “no room to put ... holder”
- → 这是说 我家这地方根本没空间放这玩意儿
- → 尺寸/兼容性不符（跟水槽/台面不配）

2. 产品容量维度理解（对 CAPACITY_EFF 有利）

- “no room to put the utensils holder” 也可能被模型理解成：
 - 在架子上/沥水区内部 **没有多余空间** 安放那个 utensils holder
 - → **有效面积/容量不够** 的典型触发词：“no room”“空间不足”

由于只看字面，“no room” 既能指**台面没空间**，也能指**架子上没空间**，模型没那么细致地区分，就两个都打了。

2. 你给 CAPACITY_EFF 的定义里，本身就有“空间/面积”这种高敏感词

我们之前给 **CAPACITY_EFF** (容量/有效面积不足) 写的定义和边界里，有类似措辞：

- “容量/有效面积不足”
- “扩容效果不达标”
- “能摆放的东西比用户预期少”等

在模型眼里，「no room」 「space」 「面积不够」这类词，**很容易被泛化到 CAPACITY_EFF 上**。

所以它看到：

no room to put the utensils holder

就会触发一条「空间不足 → 容量问题」的模式匹配，而没细分这是“**环境空间**”还是“**产品内部空间**”。

3. 多标签任务下，模型倾向于“宁可多打，不要漏打”

你给它的任务是：

一条留言可以打多个标签

对于 LLM 来说，当它在两个标签之间犹豫：

- FIT_COMPAT：看起来像
- CAPACITY_EFF：也有点像

在没有「最多选 N 个」或「要选最主要原因」这种硬约束时，

它的缺省行为就是：两个都打，保证 recall 高一点，反正你没说“不能重叠”。

这就是典型的：

不确定就全选 → 出现你看到的 FIT_COMPAT + CAPACITY_EFF 双标签。

4. 怎么改设计，能减少这种“词面+多打”的情况？

可以从两边下手：**标签定义/边界 + 提示词规则**。

4.1 在标签边界里，把“空间归属”写死

- 在 FIT_COMPAT 的边界里加一句类似：

“评论中出现 no room / no space / can't put it anywhere 等描述，且对象是水槽/台面/厨房/柜内等环境空间 → 归入 FIT_COMPAT，不打 CAPACITY_EFF。”

- 在 CAPACITY_EFF 的边界里反向声明：

“仅当用户抱怨的是架子本身能放的餐具/碗盘数量少、层数有限、扩容后仍放不下等内部容量问题时使用；

若抱怨的是厨房/水槽/台面等环境空间不够 → 属于 FIT_COMPAT 而非 CAPACITY_EFF。”

这样你等于用「**内部 vs 外部**」来切割两个标签的空间概念。

4.2 在 prompt 里加“互斥+证据”规则

例如在【要求】补一条：

text

 复制代码

在处理与“空间/room/space”相关的抱怨时：

- 若用户说的是厨房/水槽/台面/橱柜等环境没有空间放下商品（如 no room on my counter, no
- 若用户说的是商品本身能放的碗盘/餐具数量太少或内部空间利用差（如 can't fit many dishes

必要时可以同时检查 **evidence**：同一句话不能同时作为两个标签的证据。

或者更简洁地说：

text

 复制代码

涉及“空间不足”的表达时，先判断这是“环境空间”还是“商品内部空间”：

- 环境空间不足 → 只用 **FIT_COMPAT**；
 - 商品内部空间不足 → 只用 **CAPACITY_EFF**；
- 禁止在同一句话模糊时同时打两个标签。

5. 这条具体留言应该怎么示范给模型？

你可以把这条当作「反例样本」写进标注指南：

示例：

UNWANTED_ITEM: This one size didn't fit either. I had no room to put the utensils holder.

- “正确标签：**FIT_COMPAT**（尺寸/兼容性不符）”
- “理由：no room 指的是厨房/水槽环境没有空间摆放这套东西，而不是架子本身容量不足；”
- “禁误标签：**CAPACITY_EFF**（容量/有效面积不足）。”

当这类**反例**足够多，LLM 就会学会：

“只要是说厨房没地方放 → 优先 **FIT_COMPAT**，不要顺手把 **CAPACITY_EFF** 也加上。”

一句话总结：

deepseek 同时打了 FIT_COMPAT 和 CAPACITY_EFF，是因为：

- 被 “no room” 这个词牵着走，
- 没分清是「环境空间不足」还是「架子内部容量不足」，
- 再加上多标签任务倾向于“多打不漏”。

你只要在 **标签边界 + prompt** 里明确写死「环境空间 → FIT_COMPAT；内部容量 → CAPACITY_EFF；二者互斥」，再给几个反例样本，它以后这类错就会明显少很多。



请你评估一下，下面这条留言应该归属到哪？

UNWANTED_ITEM: Diesnt like the feel will scrape the sink

已思考 29s >

这条我不会归到 NO_MATCH，而是会**打到一个很具体的设计问题标签**：

UNWANTED_ITEM: Diesnt like the feel will scrape the sink

= Doesn't like the feel, will scrape the sink.

推荐标签： PARTS DESIGN — 分区/配件设计不合理

原因：

- “will scrape the sink” 很明确：

- 担心这个架子的接触面（脚、边缘、底座）会划伤水槽；
- 这在我们体系里，已经被并入

“如果因为没有防滑/防刮脚垫、接触面太硬/太锋利而伤到台面或器具 → 归入设计问题 (PARTS DESIGN 或 DOA DAMAGED) ”

- 评论里没有说“少了脚垫/配件没装上”，只是 **产品本来这样设计就很硌/很硬**，所以更像是：

→ **设计层面的缺少保护 / 接触面设计不合理** → PARTS DESIGN

“Doesn't like the feel” 在这里也不是纯审美，而是和“会刮花水槽”连在一起的体验问题，不建议打 LOOK_COLOR；更不是抽象到“质量差”，所以不打 NO_MATCH。

总结：

这条评论应该标：

- PARTS DESIGN 分区/配件/接触面设计不合理（缺少保护，担心刮花水槽）



是否需要新增标签呢？

已思考 7s >

短结论：现在这套体系里，不需要为这条留言单独新增标签，用好 PARTS DESIGN 就够了。

1. 这条留言的本质是什么？

“Doesn't like the feel, will scrape the sink.”

拆开看：

- “feel” → 不是情绪 feel，而是手感/触感/接触面的质感
- “will scrape the sink” → 担心和水槽接触的那部分会刮花水槽表面

本质问题是：

接触面（底座/脚/边缘）设计缺少保护，**几何/材质硬邦邦** → 容易划伤水槽。
这是一个**结构/配件设计问题**，而不是一个全新维度。

所以放在我们现有体系里，逻辑很自然地归到：

- PARTS DESIGN : 分区/配件设计不合理
- 加一句边界说明：

“包含因缺少防滑/防刮脚垫、接触面过硬或边缘过锐，导致可能刮伤水槽/台面的情况。”

2. 为什么我不建议重开一个「刮伤水槽」标签？

你之前其实有过一个类似的标签 (SURFACE_PROTECT)，后来我们是刻意“砍掉+合并”的，理由是：

1. 本质仍是设计问题的一种具体表现

- 没脚垫/脚垫太硬/边缘太锐 → 都是“设计没有考虑保护”。
- 再单独开一个“水槽保护不足”，容易和 PARTS DESIGN、STABILITY_TIP 等互相打架。

2. 颗粒度太细，对一线和模型都是负担

- 对运营：
 - “是 PARTS DESIGN 还是 SURFACE PROTECT？”
 - 又开始纠结。
- 对模型：
 - 出现 “scrape sink” 时，会在两个标签之间摇摆，反而增加错标。

3. 当前标签数量已经比较精简

- 现在这套 18 个标签，我们一直在控制不要“再爆炸”，
- 只要能在 PARTS DESIGN 的定义里把“刮伤水槽/台面”写清楚，就不会丢信息。

3. 如果你非常关心“刮伤水槽”，应该怎么处理？

推荐做法不是新增标签，而是：

1. 在 PARTS DESIGN 的边界中明确写入（调整文案即可）

比如补一句：

“包括因接触面缺少防滑/防刮脚垫、线条/边缘过硬过锐，导致用户担心或实际发生刮伤水槽/台面的情况。”

2. 在规则/LLM 提示词中给一个示例

- 直接用这条评论作为训练样例：
 - “will scrape the sink” → 只打 PARTS DESIGN
 - 禁误标签： MATERIAL_RUST / LOOK_COLOR / DRAIN_FLOW 等

这样，你既：

- 保持了体系的 MECE 和精密度，
- 又不会在“刮伤水槽”场景下丢掉任何信息。

小结

- 这条留言**不需要新增标签**；
- 正确做法是：**挂在 PARTS DESIGN 下，并在标签边界里明确把“可能刮伤水槽/台面”写为典型场景之一。**



我觉得应该新开一个标签，因为part design强调的是结构

已思考 22s >

我赞同你新开一个标签，这个场景**确实值得单独拉出来**，而不是硬塞进 PARTS DESIGN。

你说得对：

- PARTS DESIGN 更强调**结构/分区几何**：高度、宽度、开孔、挂点位置这些；
- “会刮花水槽/台面”其实是一个**“接触面保护/防刮设计”维度**，跟“结构分区”不是一回事。

一、这类问题的本质是什么？

像这条：

Doesn't like the feel, will scrape the sink.

核心信息是：

- 产品与水槽/台面/灶台的接触面太硬、太锐，缺少保护（胶垫、缓冲等），
- 用户担心或已经出现**刮伤/磨损表面**的问题。

这类反馈在你这个品类里会反复出现，而且有几个特点：

1. **不是整体稳定性**（不是晃、倒）
2. **不是结构分区几何**（不是分区放不下碗盘）
3. **不是外观色差**
4. **也不是生锈/材质失效**

它就是**“环境表面保护欠缺”**：

架子本身没坏，但会把水槽/台面“弄坏/弄花”。

从运营和产品决策角度看，这个维度很关键：

- 要不要加胶脚/胶条？
- 材质是否需要一层软胶包边？
- 是否要在页面图上强调“不会刮花你的水槽/台面”？

所以我同意这应该是一个**独立标签**，而不是强行塞进 PARTS DESIGN 里“糊掉”。

二、建议新增标签设计（可直接用）

1. 标签基础信息

- **tag_code (建议) :** SURFACE_PROTECT
- **tag_name_cn:**
👉 水槽/台面保护不足（易刮花）
- **category_code:** 仍挂在 CAT_STRUCT_FIT
- **category_name_cn:** 结构与摆放/适配体验

理由：

- 它描述的是“**产品与摆放环境的接触效果**”，
 - 和“结构/适配”是一组（放哪儿、怎么接触），归到 STRUCT_FIT 比归到 外观/材质 更合理。
-

2. definition (定义)

水槽/台面保护不足（易刮花）：

产品与水槽、台面、灶台等接触面缺少防滑/防刮保护，或边缘/底部材质过硬、过锐，导致用户担心或实际出现刮伤、磨损这些表面的风险。

3. boundary_note (边界说明)

可以这样写，方便运营和 LLM 判断边界：

- “评论中出现 “scrape the sink / scratch my sink / scratch the counter / leave marks on the countertop / damage the surface”等表达，且焦点是水槽/台面/灶台等环境表面被刮花/磨损 → 打本标签。”
 - “若主要问题为整体晃动、倾斜或易倒（不稳定）→ 优先打“整体稳定性差/易晃动 (STABILITY_TIP) ”；”
 - “若问题是架子内部分区/配件几何设计不当，导致碗盘/餐具放不稳或浪费空间 → “分区/配件设计不合理 (PARTS_DESIGN) ”；”
 - “若只是外观不好看、做工粗糙但没有涉及刮伤环境表面 → “外观颜色/做工不满意 (LOOK_COLOR) ”；”
 - “由于包装或运输导致水槽/台面被刮花（例如安装/试用时滑动过程中被锐边划伤），但核心在于产品接触面设计问题，而非包装问题 → 仍归本标签，而不是“包装太薄/防护不足 (PACK_WEAK) ”。”
-

三、与现有标签的关系怎么界定？

1. 和 PARTS_DESIGN 的关系

- **PARTS_DESIGN：**
 - 刀叉篮/杯架/层板等分区、配件 **几何设计不合理**（高度、宽度、孔距、位置等），影响的是“能不能放东西、放得稳不稳、空间是不是被浪费”。
- **SURFACE_PROTECT (水槽/台面保护不足) :**
 - 关注的是“与环境表面接触会不会伤到那块表面”，和防滑脚、包边、软胶条、有无保护垫相关。

你刚说“PARTS_DESIGN 强调的是结构”，这正好给了我们一个很清晰的拆分：

- “**结构几何/分区布局问题** → PARTS_DESIGN”
- “**接触面保护/防刮设计问题** → SURFACE_PROTECT”

2. 和 STABILITY_TIP 的关系

- STABILITY_TIP：整体晃、软、倾倒风险
- SURFACE_PROTECT：即使不晃，但底面/边缘太硬，会刮花水槽/台面

典型区分：

- “It wobbles / tips over easily / doesn't feel stable” → STABILITY_TIP
- “Feels like it will scratch my sink/countertop” → SURFACE_PROTECT

四、这条具体留言归属新标签示范

UNWANTED_ITEM: Doesn't like the feel will scrape the sink

正确打法变成：

- SURFACE_PROTECT 水槽/台面保护不足（易刮花）
- evidence: "will scrape the sink"

不需要打 PARTS DESIGN；

如果他还提到“wobbly / not stable”，才会考虑叠加 STABILITY_TIP。

五、总结

- 你的直觉是对的：“**会刮花水槽/台面**”确实应该独立出来。
- 推荐新标签：
 - SURFACE_PROTECT — 水槽/台面保护不足（易刮花），归属 CAT_STRUCT_FIT（结构与摆放/适配体验）。
 - PARTS DESIGN 继续纯粹负责“内部结构/分区几何”，SURFACE_PROTECT 专门负责“接触面保护/防刮”。

如果你愿意，下一步我可以帮你：

- 基于当前 v5 版本标签，写出**完整的 SURFACE_PROTECT JSON 结构 + 同时更新 PARTS DESIGN 的边界说明**，你可以直接拷到你的标签库里用。



请你评估一下，下面这条留言应该归属到哪？

DEFECTIVE: The metal is week. With plates in it it bends

已思考 20s >

这条我会非常明确地打到：

STABILITY_TIP — 整体稳定性差/易晃动

原文：

The metal is weak. With plates in it it bends

关键信息：

- “**metal is weak**” → 材质太薄/强度不足
- “**with plates in it it bends**” → 一放盘子就下弯、变形

这和我们给 **STABILITY_TIP** 写的定义是完全对齐的——

架体在承重状态下出现下弯/变形，影响稳定性和使用安全。

不建议打：

- **MATERIAL_RUST**: 它只管“生锈/涂层失效”，这条没有提。
- **CAPACITY_EFF**: 不是“能放的盘子太少”，而是“放正常盘子就弯了”，核心是承重/稳定性，而不是空间。
- **PARTS DESIGN**: 不是局部分区几何不合理，而是整体结构/材质强度不足。

所以这条在你体系里最干净的归属就是：

STABILITY_TIP (整体稳定性差/易晃动) , evidence: "metal is weak" / "with plates in it it bends"



你觉得deepseek为什么同时打了**STABILITY_TIP**和**MATERIAL_RUST**

我先说结论：

deepseek 同时打 STABILITY_TIP + MATERIAL_RUST，很大概率是被“metal + weak”这两个词带跑偏了，把“金属太软/太薄”误解成“材质不好/容易生锈”这一类泛材质问题。

原句回顾一下：

DEFECTIVE: The metal is week. With plates in it it bends

你和我现在的理解是：

- 关键问题：一放盘子就弯 → 承重/稳定性问题 → **STABILITY_TIP**

- 完全没提: rust / rusty / corrosion / spots / peeling
→ 不该打 MATERIAL_RUST

那 deepseek 为啥会加上 MATERIAL_RUST? 我拆一下可能的“错误思维链路”:

1. 词面误导: 看到 “metal” 就条件反射到 MATERIAL_RUST

MATERIAL_RUST 这个 code + 名称本身就很容易让模型形成“模式”:

- 看到 metal, stainless, steel, rust, rusty → 想到“材质/防锈失效”
- 看到 weak, cheap, flimsy + 金属 → 很容易泛化成“材质太差”

在这句里有:

- “The **metal** is weak”
- “DEFECTIVE”这个 reason code 本身就是个“质量问题”信号

模型有可能这样想:

“金属质量差 = 材质问题 → 那就打 MATERIAL_RUST 一枚”

它不会像你现在的边界那样细分“材质失效（生锈）” vs “结构/承重不足（下弯/变形）”。

2. 旧定义/模糊记忆: 把 MATERIAL_RUST 当成“大材质质量差”标签

如果它记忆里对 MATERIAL_RUST 的印象是类似:

“材质不好 / 质量不佳 / 防锈差”

而不是你现在严格写的:

“出现生锈、掉皮、斑点、涂层脱落 等防锈失效问题”

那只要看到:

- metal + weak + defective
就很容易判成“材质质量有问题”，顺手点了 MATERIAL_RUST。

你现在这套边界其实是比模型的“直觉记忆”更细的，它还没对齐你最新版本的规则。

3. 多标签任务的习惯: 犹豫时两个都上

你的提示是允许多标签的，多标签任务里 LLM 的典型行为是：

“有一点像的都打上，宁可多打不要漏打”。

在这句里：

- “With plates in it it bends” → 明显像 STABILITY_TIP
- “The metal is weak” → 对模型来说也有点像“材质不佳” → MATERIAL_RUST

它在没有“互斥规则”约束的前提下，为了保证不漏，把两个都打了：

- STABILITY_TIP：覆盖“弯曲/不稳”
- MATERIAL_RUST：覆盖“metal + weak”这个材质直觉

从模型角度看：**多打一枚风险小**；

从你业务角度看：这就是我们要通过“边界+prompt”矫正的地方。

4. 怎么在体系 & 提示里帮它“改掉这个坏习惯”

你要的是：

- 只要出现“弯、软、承重下弯” → 主打 STABILITY_TIP
- MATERIAL_RUST 必须有 **rust / rusty / corrosion / spots / peeling** 等明确迹象才能触发

可以这么做：

4.1 修改 MATERIAL_RUST 的边界，写死触发条件

在标签说明里加 VERY 明确的规则（给人看也给模型看）：

MATERIAL_RUST — 材质/防锈失效

- “仅在评论中出现以下明确信号时使用：”
 - ““rust / rusty / rust spots / corrosion / discoloration from rust / peeled coating / flaking coating / rust stains”等；”
 - “仅说 metal is weak / cheap / flimsy / thin，且问题表现为弯曲、承重不足 → 归入“整体稳定性差/易晃动 (STABILITY_TIP) ”或结构标签，不打本标签。”

4.2 在 prompt 里加一句“优先级/互斥逻辑”

在【要求】里加一条类似：

text

 复制代码

当评论同时涉及“材质/金属（metal）”与“弯曲/下垂/承重不足（bend, sag, flex under load）”

- 1) 若出现 `rust / rusty / corrosion / peeled / flaking` 等“生锈/涂层脱落”字样 → 可以打
- 2) 若仅描述为 `metal is weak / flimsy / thin` 且主要表现为“放盘子就弯 / 支撑不住”，优先打

实际可以写成更简洁一点，但这个逻辑要讲清楚：

“没有 `rust`, 就不要打 `MATERIAL_RUST`, 哪怕提到了 `metal is weak`。”

4.3 用这条评论做反例样本

直接放入 few-shot 示例里：

