自动化评测需求阐述

背景

wegic是基于LLM构建的应用，那么对于应用功能性能的评测，除了常规传统软件的流程测试外，还直接映射了大模型的功能性能。在所有这种场景下，任何对交互了大模型节点的Prompt或者节点workflow更新的操作，理论上都要进行测试验证，传统的测试方法这时已经不是测试效率问题，而是准确性可能都不足够了，而大模型应用中，这种prompt的更改和workflow的更改却很频繁。所以要构建一套针对LLM应用的评测方法和系统，来系统性提高测试的效率和准确性。

这里的评测不是指传统产品功能测试，也不是指基座大模型的评测，而是从我们具体的产品和业务角度的评测，即端到端的效果评测，聚焦于llm应用这个特点。从重要性看：

- 评测产生的结果，是衡量应用好坏的标准。换句话说，评测能告诉我们大模型在wegic的业务下，能达到一个什么效果，能告诉我们经过我们的core层组合优化改进后这个组合智能基座能对外提供一个什么样的效果。
- 评测是应用优化的基础。产品是不断变更迭代的，新的变动，比如我们改了prompt，这个变动是否应该上线，由评测来决定。
- 评测保证了这个AI智能应用在实际场景中或者线上用户手里的效果。

因此基于上述目标，我们要设计一个新的评测体系以及对应的工具。

预期收益

直接收益

一个自动化执行的评测系统和工具，相对准确的、结果可量化的、高效率执行的评测工具。

它的用处有以下几个方面：

1. **大幅提高测验证效率**

直接举例，现在要执行一次局部修改的验证，20个场景，每个场景执行10遍（这是必要的，因为算法是概率性的，而非传统软件的确定性的输出），大约需要耗掉测试**1000分钟**，换算到有效工作时间，约要耗掉3人天。而使用自动化测试的话，约只耗时**20分钟**，而且执行时不用人工等待，提高的效率不能以倍数来衡量了。

而且它可以完全水平扩展，不管多大规模的测试，只要增加点机器，就可以维持这个总耗时不变。显然串行的人工测试方法完全无法做到。

大幅提高验证效率除了直接的迭代时间和人力成本收益，最最重要的是我们可以快速验证一些想法的有效性。以前的很多迭代可能都限于测试人力原因，而直接放弃掉，现在我们可以直接不限于测试人

力，变更后可以很快的直接的得到验证结果，那么多想法中得出正向优化想法的概率大大增加。

2. **提高验证的全面性、准确性、鲁棒性和可量化性，能给迭代优化予以正确指引**

对于LLM的测试，人工测试理论上来说是最准确的测试。但是限于人力资源和时间成本，现在实行的一般的测试都是挑几个case，执行有限几次后，基于主观感受给出评测的定性结果，而非严格的定量结果。因为LLM的结果是概率性的，这样做的后果可能是失之全面和鲁棒的，这直接导致了准确性可能值得怀疑，甚至间接导致我们失去正确的优化方向。

而基于自动化的评测，可以显著改善这几个方面的问题。因为测试case的多样性类型和标签，分类统计结果可以比较准确的衡量我们应用的欠缺和优势，能相对准确给出前后两个版本的量化评分结果，显示出在哪些方面是进步了，哪些方面退步了等等。至于基于这些评测结果做的下一步动作，这里不再论述。

间接收益

1. 人工标注过的优质数据集

所有基于算法的工作，优质数据集对最终的成功至少起到70%的作用。在这里也一样，所有从算法角度考虑的优化工作，都必须基于数据集才能启动。举例如：RAG、Prompt自动优化工程、Few-shot、模型微调等等。没有这个数据集，那么目前大模型应用领域最流行的解决方案，我们连尝试都没办法尝试，只能人工想规则，然后手工修改Prompt做优化这种单一的迭代方式。领域数据在这里起不到任何作用。无论如何，基于算法的应用里没有数据和自动化这两个因素的话，那么迭代上限、效率和最终的产品竞争壁垒上都需要打个问号。

线上的用户数据，有一定作用，但是质量和数量都很难讲有什么大作用。现在有标注过的足够数量的优质数据集，我们就能启动一些算法和自动化上的优化措施，变成两条腿走路，跟上主流的研发方法。

2. 自动化且量化的工作体系

这里构建了一个自动化量化的工作示例，以后可以泛化到多个场景，这里构建的是一个自动化的场景平台，后续可以插件化的扩展很多场景。大家的工作流程里也许就有了自动化和量化这样一种工作方式，我个人认为这是小团队高效率的必备技能（尤其是对于技术人员）——工欲善其事，必先利其器。工作中能自动化的就尽量自动化，能量化的就尽量量化。复用的威力是无穷的，收益是随着时间不断叠加的。

技术架构

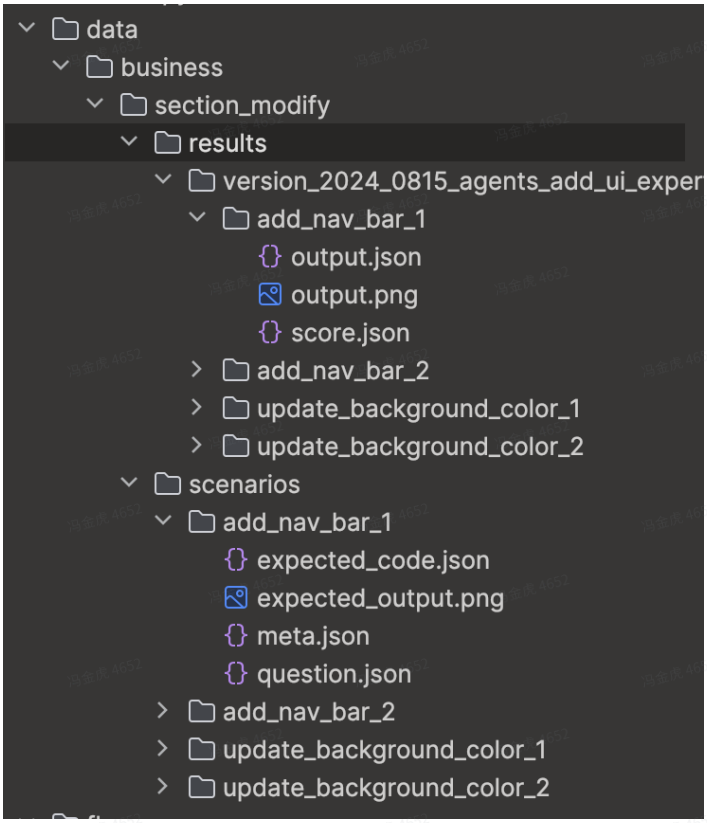
讲完了背景和收益，这里来讲具体的实现。

这里以**局部修改**这个业务来讲述，这也是本期第一个要落地的具体目标。

总体来说，我们的应用是基于LLM构建的，那么我们衡量应用能力的标准就和衡量LLM能力标准很类似，只不过我们有业务场景下的直接表现手段，比如我们这里就是UI/UX对用户修改要求的满足度，以及附带的一些领域评价标准，比如美观度、页面元素排布是否合理等等。

上述的两个特点，直接决定了我们这个评测系统的核心工作，一个是构建类似评测LLM能力的评测系统，第二个是找出并实现这个场景下的量化方法。

LLM评测是基于数据集的，尤其是问答数据集是更通用的做法。它具有结果直观性好、可解释性强、可复现、对不同系统具有相同的客观性等优点。所以这里我们也是使用问答对这种形式来构建数据集。形如：
{q:prompt/request/question a:期望的answer}. 具体形式如下：



其中需要构建的数据集就是business下的scenarios。详细数据不在这里展示了。

关于量化方法，借鉴论文https://arxiv.org/pdf/2406.20098，主要做法是使用LLM的vision能力，从网页UI的相似度方面给出10个维度的评分，然后我们这里再根据情感分析算法计算出局部修改Request在这个十个维度的分布向量，最后给出一个二维向量综合计算出的最终量化得分。

总体是实现了形如 数据集 -> 量化计算-> 数值的工具链。

目前遗留两个自动化的方面需要后续迭代优化，一是交互性的UX评测，二是我们这里特有的Editable的可编辑性测试。这两个方面也是有自动化手段的，后续补充上。

总体来说常规的工作流如下：

第一步： 梳理指令，meta信息里要体现指令的分类、标签、难度等信息，方便后面做更全面合理的统计结论。

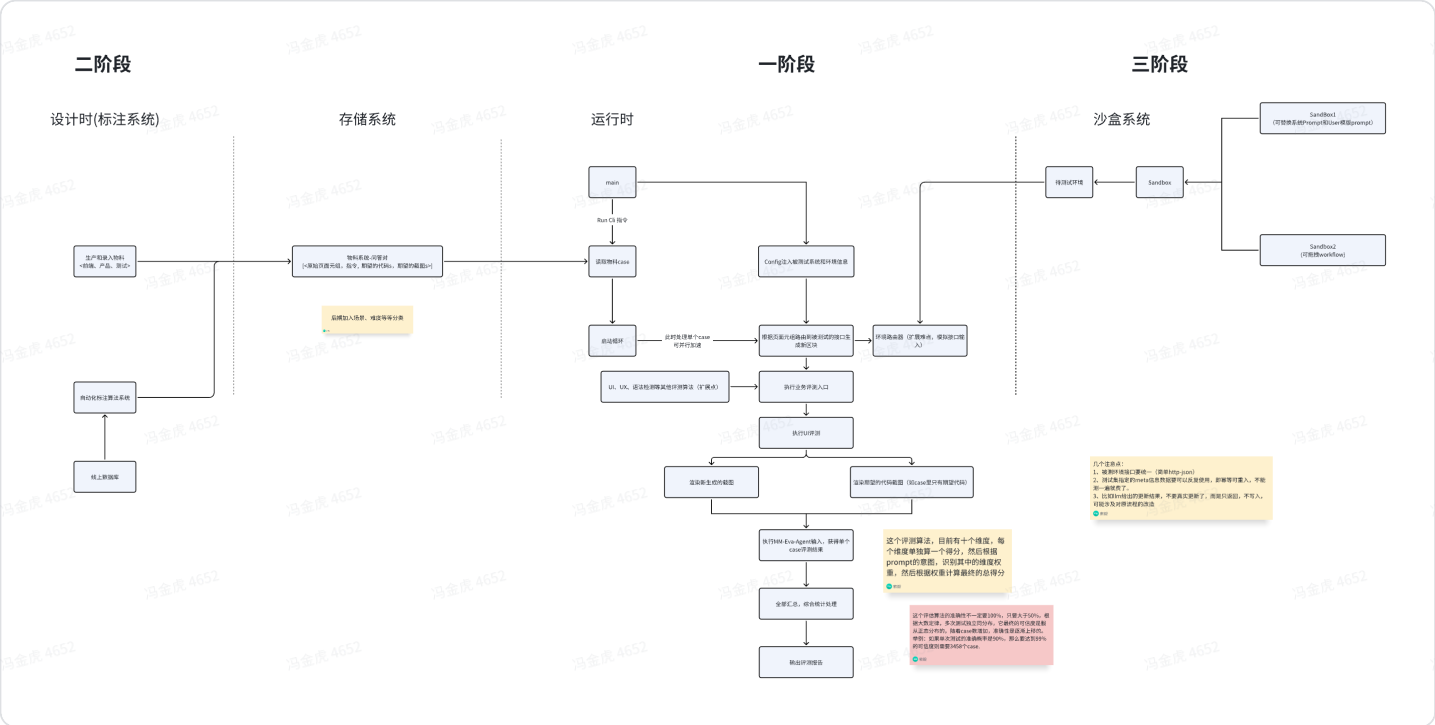
第二步： 构建相应的case

第三步：合作前端来生产对应的answer

第四步：量化计算

第五步：迭代

完整的系统架构



排期

从系统架构看，目前可以推进执行第一阶段的开发，工作量梳理如下：

1. 工具和框架（地址：git@gitlab.js.design:ai-build/llm-app-eval-sys.git）：索毅

工具和框架架构的雏形已经具备，剩余工作量如下:

- 存储从本地迁移到远程对象存储　　1天
- 执行环境的参数实现和切换的路由功能　　1天
- 评测结果维度计算算法的完善　　1天

2. 后端系统可测性改造（@后端）

预计2天

3. 构建数据集（@测试）

- 梳理指令和附加分类、标签等信息　2天
- 构建基础的落地载体　2天
- 之后是逐一生产case，每个case的生产难度有偏差，这里取平均值，单个case需要输入上述两个信息，最难的是生产期望的区块截图，一部分现有系统可以自动产出，一部分需要前端手动生产改写。

可以自动产出且符合标准的一般是难度低的，这样一个case生产平均需要10分钟1个；需要前端手动生产的是难度较高的，平均30分钟1个。

数据集分布量需求需要根据上述的分类和场景等来做更科学的决策。不过这里先总体按照50个case可用来算，约25个简单case，25个复杂case，简单case约1~2天的工作量；25个复杂case需要一个前端一个测试配合，测试耗时约2人天，前端耗时约3人天。

并行最长路径在测试侧，**约8天**可以落地。

后续这些case可以渐进性的生产补充，如每周贡献5个case，数据集数据越多则测试效果越好。