

Chat2Workflow: 一种基于自然语言生成可执行视觉工作流的基准

Yi Zhong[♥], Buqiang Xu[♠], Yijun Wang[♥], Zifei Shan[♥], Shuofei Qiao[♠],
Guozhou Zheng^{♠*}, Ningyu Zhang^{♠*}

[♠]Zhejiang University [♥]Tencent

{zhongyi0212, zhangningyu}@zju.edu.cn

<https://github.com/zjunlp/Chat2Workflow>

摘要

目前, 可执行的可视化工作流已在现实世界的工业部署中成为主流范式, 具备较强的可靠性与可控性。然而, 在当前实践中, 此类工作流几乎完全依赖人工工程构建: 开发者必须仔细设计工作流、为每一步编写提示词, 并随着需求演进而反复修正逻辑——导致开发成本高昂、耗时且易出错。为研究大语言模型是否能够自动化这一多轮交互过程, 我们引入了 **Chat2Workflow**, 这是一个直接从自然语言生成可执行可视化工作流的基准, 同时提出一种鲁棒的 **Agentic 框架** 以缓解重复执行错误。Chat2Workflow 基于大量真实业务工作流构建, 每个实例均设计为生成的工作流可经过变换后直接部署至 Dify、Coze 等实际工作流平台。实验结果表明, 尽管当前最先进语言模型通常能捕捉高层意图, 但在生成正确、稳定且可执行的工作流方面仍面临挑战, 尤其在复杂或动态变化的需求下表现不佳。尽管我们的 Agentic 框架最高可带来 5.34% 的修复率提升, 但现存的实际差距仍使 Chat2Workflow 成为推动工业级自动化发展的基础。

1 引言

大型语言模型如今已成为众多现实应用的核心, 越来越多地通过基于智能体的系统进行部署 (Zhao et al., 2023; Wang et al., 2024; Xi et al., 2025)。现有系统大致可分为 ReAct 智能

体 (Yao et al., 2022) 和 Agentic 工作流 (Zeng et al., 2023; Fan et al., 2024; Qiao et al., 2025)。尽管 ReAct 智能体具有灵活性, 但先前的研究与行业经验表明, Agentic 工作流在可靠性和可控性方面更适用于工业场景 (Shi et al., 2025)。最近的访谈 (Pan et al., 2025) 显示, 简单方法在生产系统中占据主导地位: 超过 **70%** 的现实世界智能体部署依赖于现成的语言模型, 未进行任何权重调优, 而是 **通过显式的流程进行编排**。这一趋势在 Dify 等广泛应用的平台中得到了体现。¹ 和 Coze², 智能体的行为主要通过工作流预先定义。

尽管广泛应用, 智能体工作流仍主要通过人工工程创建。人类开发者必须将自然语言需求转换为结构化工作流, 这一过程耗时且容易出错。这引出了一个自然的问题: **能否从自然语言自动生成智能体工作流?** 实现这一目标具有两大挑战。首先, 现实世界的需求往往复杂且隐含, 仅凭自然语言难以推断出正确的控制流程和工具使用。其次, 用户需求经常变化, 要求工作流在保持正确性和一致性的同时进行修改或重新生成。

为了系统地研究这一问题, 我们引入了 **Chat2Workflow**, 这是首个专注于从自然语言生成可执行视觉工作流的基准 (图 1)。Chat2Workflow 是一个高质量的数据集, 包含六个领域中的 273 个实例 (图 2), 用于评估

* Corresponding Author.

¹<https://dify.ai/>

²<https://www.coze.com/>

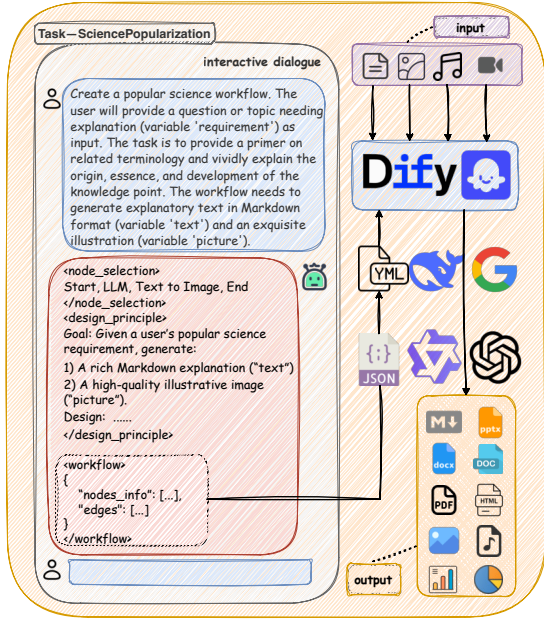


图 1: Chat2Workflow 中的一个示例任务，其特征是具有真实且可变的自然语言指令输入，并生成可直接进行变换和集成到实际工作流平台（例如 Dify 和 Coze）的输出。

生成的工作流是否满足执行预期要求。每轮任务均配备了跨不同领域和复杂度水平的真实用户请求及目标工作流组件。

该基准评估模型是否能够正确推断工作流结构、选择合适的工具，并生成与用户意图一致的可执行工作流。对 Chat2Workflow 的实验表明，当前最先进的语言模型虽然通常能捕捉高层次意图，但在生成正确且稳定的工作流方面仍存在困难，尤其当任务变得更加复杂或需求发生变化时。这些结果表明，可执行工作流生成仍是实现实际系统自动化的主要瓶颈。Chat2Workflow 为这一挑战提供了具体的基准，并指明了未来在结构化推理和自适应工作流合成方面的研究方向，以构建更可靠的系统。

2 构建基准

2.1 工作流编排平台

概述 Dify 和 Coze 是主流的通过连接功能节点构建 Agentic 工作流的平台。这些工作流在前端以有向非循环图的形式可视化，并在后端以结构化的 YAML 文件形式存储。平台

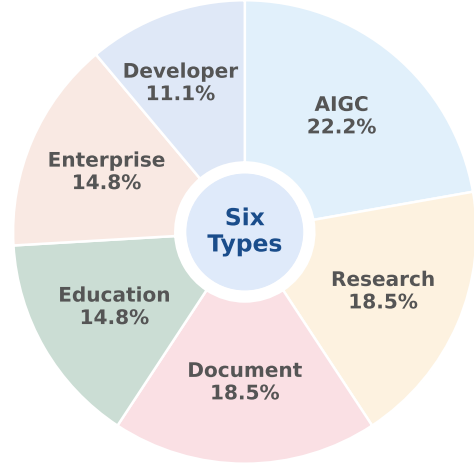


图 2: Chat2Workflow 中任务类型的分布。该基准涵盖六个领域：AIGC、研究、文档、教育、企业及开发者。

按结点逐个执行任务，自动处理工具调用、推理和数据流。以 Dify 为例，它提供了必要的内置单元（如代码、If-Else），以及数百个由社区驱动的扩展（如 Google 搜索），从而能够解决高度多样化且复杂的任务。

2.2 任务表述

我们将工作流生成任务形式化为一个多轮交互过程。给定当前轮次的任务指令和历史交互对话，我们旨在使语言智能体能够生成以 JSON 表示的图结构工作流，以结构化的方式描述结点及其连接关系。形式上，给定指令 q ，历史对话 H 以及语言智能体 \mathcal{M}_θ ，工作流生成可表示为：

$$\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow \mathcal{M}_\theta(q, H), \quad (1)$$

其中 \mathcal{G} 是一个有向非循环图，其结点列表 $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$ 包含具有预定义参数的结点或工具，边 $\mathcal{E} = \{(v_i, p, v_j)\}, 1 \leq i \neq j \leq n, p \in \mathbb{Z}_{\geq 0}$ 表示源结点 v_i 通过输出分支端口 p 连接到目标结点 v_j 。

直接生成可执行的工作流对大模型而言具有挑战性。为解决这一问题，我们采用思维链 (CoT) 方法，输出一个简化的 JSON。模型生成选定的结点 $\hat{\mathcal{V}}$ 、规划推理 p 以及 JSON

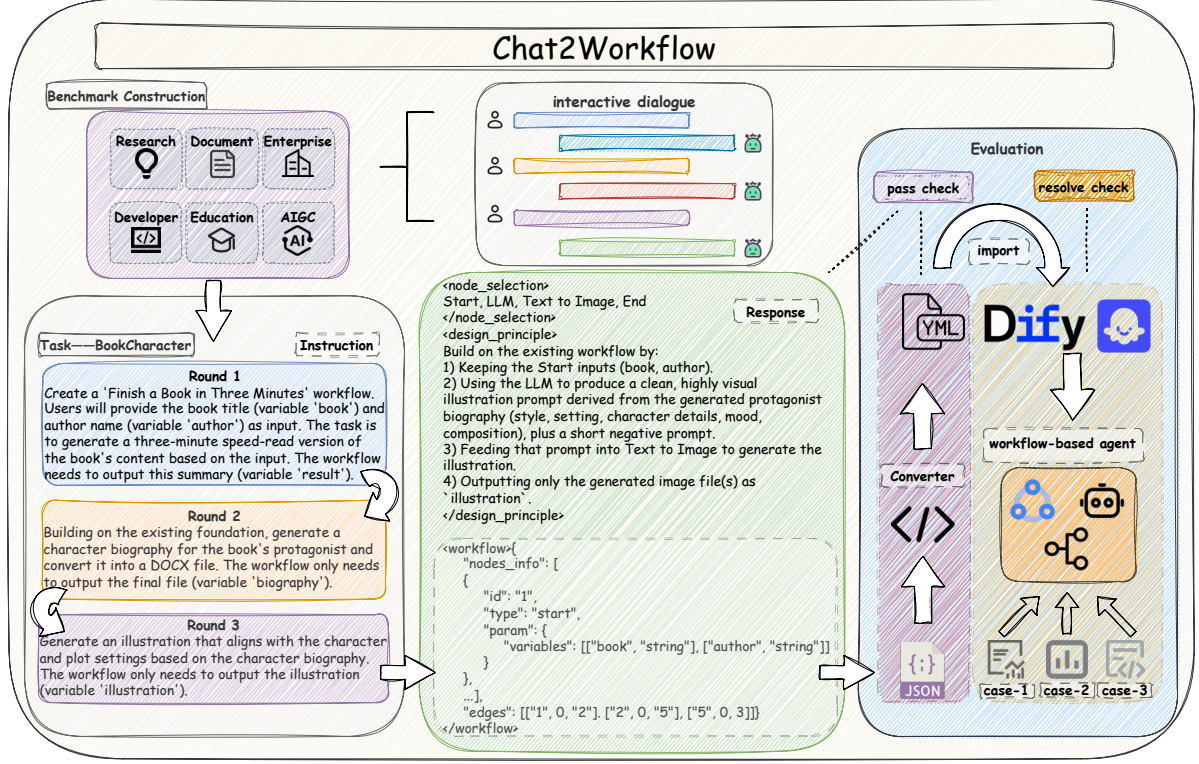


图 3: Chat2Workflow 基准构建与评估框架概述。左: 我们从六个任务领域（研究、文档、企业、开发、教育、AIGC）收集工作流，并逆向工程多轮指令。中心: 用户通过对话与大模型交互，模型生成包含思维链推理（结点选择、设计原则和结构化工作流）的 JSON 格式工作流。右: 将 JSON 工作流转换为 YAML 格式，上传至平台执行，并通过测试用例评估，计算通过率和解决率。

表示 r ，并分别包裹在 $\langle \text{node_selection} \rangle$ 、 $\langle \text{design_principle} \rangle$ 和 $\langle \text{workflow} \rangle$ 标签中。该 JSON 随后被转换为可执行的 YAML 文件。这一过程的形式化表示为：

$$\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow r \leftarrow \{\hat{\mathcal{V}}, p, r\} \leftarrow \mathcal{M}_{\theta}(q, H), \quad (2)$$

其中 \mathcal{V} 和 \mathcal{E} 分别对应 JSON 中的 `nodes_info` 和 `edges` 字段。

2.3 数据集构建

图 3 展示了我们的构建与评估框架。为解决多轮工作流数据集缺失的问题，我们从官方渠道和 GitHub 上收集了生产级别的 Dify 与 Coze 配置。基于这些源自实际生产的流程，我们逆向推导出了工作流创建指令集。为了确保每一轮的指令均基于前一轮，并可能对工作流行为进行添加、修改或优化，我们在同一任务

上下文中对流程进行选择性聚类，然后将每个流程重写为一轮指令，整体作为一个任务。为便于后续评估，所撰写的指令清晰定义了输入与输出变量，并对某些关键结点的使用给出了明确提示。最终，我们获得了包含 27 个任务的数据集，每个任务包含 2 至 4 轮指令。这 27 个任务可进一步根据场景划分为以下六类：研究、文档、企业、开发、教育与 AIGC。更多细节见附录 A。

尽管同一任务目标可通过不同工作流实现，但某些特定结点不可避免会被使用，这些结点将提前记录在对应轮次中。在收集和整理任务指令后，我们为每条指令配备了三个测试用例，以实际开展端到端测试。对于输出相对明确的任务轮次，测试用例将提供参考答案。对于其他一般情况，仅提供任务输入。测试用例主要来源于现有的公开数据集，其次是互联

网上的零散资料，最后是语言模型生成的合成数据。测试用例的设计充分考虑了历史轮次的任务需求，力求全面覆盖多分支工作流的各种情况。

2.4 数据选择与处理

我们选择了 20 个使用频率最高的结点，以全面覆盖我们的数据集。为降低生成复杂度，我们简化了结点的输入/输出变量接口：语言智能体只需提供关键主变量的键值对，而次要变量则采用默认设置。为了赋予语言智能体对每个结点及其对应输入/输出接口的使用能力，我们为 20 种结点类型构建了结点知识库，作为系统提示中理解结点生态系统的关键组成部分。基于上述原则，我们收集了官方结点文档，根据需要选取相关内容进行必要的改写，然后整理成预定义的结构化格式。

2.5 质量控制与数据集统计量

对于已构建的数据集，我们对每一个独立的输入-输出对执行**人工验证流程**，以保障数据集的正确性、完整性、隐私性与整体质量。在此流程中，未达到我们质量标准或违反预定义规则与合规准则的工作流会被系统性地识别并移除。此类情况包含指令模糊、无效中间步骤、随时间变动的数据源，或不一致的工作流行为。经过此严格的过滤与验证流程后，我们整理出 Chat2Workflow 这一高质量数据集：该数据集包含 27 项任务，总计 79 条多轮指令，每条指令附带三个测试实例（总计 237 个实例），用于评估生成的工作流是否符合指定的执行要求。

2.6 评估协议

与文本和图像等通用生成任务不同，工作流的质量不能仅基于其表示形式本身来判断。必须检查其是否能够正常执行以及执行结果如何。因此，我们采用了一种两阶段渐进式评估方法。第一阶段针对工作流表示形式本身，检查其是否与思维链序列一致，以及生成的工作流 JSON 能否转换为有效的 YAML 文件。第二阶段则验证工作流能否成功导入平台、正确

执行，并产生满足指定要求的结果。我们引入了两个指标，*pass rate* 和 *resolve rate*，分别对应于第一阶段和第二阶段。

Algorithm 1: Pass Rate Pipeline

Data: Model response \mathcal{R} , Reference variables \mathcal{V}_{ref} , Key nodes N_{key}

Notation: Selected nodes $\hat{\mathcal{V}}$, Planning rationale p , JSON file r , YAML file F_{yaml} , Workflow \mathcal{G}

```
// Step 1: Format Check
 $\hat{\mathcal{V}}, p, r \leftarrow \text{PARSE}(\mathcal{R});$ 
if  $\mathcal{R}$  lacks required tags or JSON  $r$  not parsable then
  return False;
// Step 2: Conversion & Import
 $F_{yaml} \leftarrow \text{JSONToYAML}(r);$ 
if conversion fails then
  return False;
 $\mathcal{G} \leftarrow \text{IMPORT}(F_{yaml});$ 
// Step 3: Variable Consistency
if  $\text{EXTRACTVARS}(r) \neq \mathcal{V}_{ref}$  then
  return False;
// Step 4: Logical Validity
if  $f_{LLM}(\hat{\mathcal{V}}, p, r)$  indicates inconsistency then
  return False;
if  $f_{LLM}(N_{key}, \hat{\mathcal{V}})$  indicates  $N_{key} \not\subseteq \hat{\mathcal{V}}$  then
  return False;
return True;
```

通过率衡量生成的 JSON 表示与思维链 (CoT) 序列以及预定义参考变量一致的子任务所占比例，并且能够成功转换为有效 YAML 文件，遵循算法 1 中概述的顺序检查步骤。

若上述四个步骤成功完成，工作流将被标记为通过并导入平台，可通过以下公式进行计算：

$$\%Pas. = \frac{N_{\text{success_subtask}}}{N_{\text{total_subtask}}}, \quad (3)$$

其中 %Pas. 表示**通过率**， $N_{\text{success_subtask}}$ 表示成功子任务的数量， $N_{\text{total_subtask}}$ 表示子任务的总数。

解决率衡量的是生成的工作流能够产生满足指令所有指定要求结果的测试用例所占的比例。算法 2 详细描述了两步验证过程。同样，我们将按照该流程依次进行检查。可以通过以

Algorithm 2: Resolve Rate Pipeline

Data: Workflow \mathcal{G} , instructions \mathcal{I} , Test case $\mathcal{T} = \{\text{input}, \text{ref_output}\}$
Notation: Execution output \mathcal{O}
// Step 1: Execution Check
 $\mathcal{O} \leftarrow \text{EXECUTE}(\mathcal{G}, \mathcal{T}.\text{input});$
if *execution fails with errors* or $\mathcal{O} = \emptyset$ then
 return *False*;
// Step 2: Output Validation
 $\mathcal{O}_{\text{file}}, \mathcal{O}_{\text{text}} \leftarrow \text{SPLITOUTPUT}(\mathcal{O});$
if $\mathcal{O}_{\text{file}} \neq \emptyset$ and *file types/extensions mismatch* then
 return *False*;
if $\mathcal{O}_{\text{text}} = \emptyset$ then
 return *True*;
if $f_{\text{LLM}}(\mathcal{I}, \mathcal{O}_{\text{text}}, \mathcal{T}.\text{input}, \mathcal{T}.\text{ref_output})$ confirms all requirements met then
 return *True*;
return *False*;

下公式计算：

$$\% \text{Res.} = \frac{N_{\text{success_case}}}{N_{\text{total_case}}}, \quad (4)$$

其中 %Res. 指 **解析率**, $N_{\text{success_case}}$ 表示执行结果符合需求的测试用例数量, $N_{\text{total_case}}$ 表示测试用例的总数。

我们采用 DeepSeek-V3 进行评估。与通过阶段和解决阶段相关的提示分别详细列于附录中的图 6 和图 7。为评估性能, 我们对两个阶段分别进行了独立采样, 通过阶段采样 500 个样本, 解决阶段采样 1,282 个样本。大语言模型的评估结果与人工评估表现出极高的一致性, 通过阶段达成 100% 的一致性, 解决阶段达成 98.83% 的一致性。

3 实验

3.1 实验设置

为了全面评估 workflow 生成能力, 我们在 Chat2Workflow 基准上对 15 个代表性模型进行验证, 报告三次独立运行的平均结果:

1) 四个**闭源**语言模型: GPT-{5.1, 5.2}、Claude-Sonnet-4.5 和 Gemini-3-Pro-Preview。

2) 十一个**开源**语言模型: Qwen-3-{8B, 14B, 32B}, Qwen3-235B-A22B, Qwen3-Coder-480B-A35B-Instruct, GLM-{4.6, 4.7}, DeepSeek-{V3.1, V3.2}, Kimi-K2-{Instruct, Thinking}。

此外, 为探索性能上限, 我们选取了两个代表性模型 (GPT-5.1 和 GPT-5.2) 进行高级 Agentic 评估, 具体内容详见第 3.5 节。

实验在 Dify 的 1.9.2 版本上进行。更多细节见附录 B。

3.2 主要结果

表 1 展示了我们的实验结果, 我们利用这些结果来分析以下研究问题。

Q1: 工作流通过率与解决率之间的差距是什么? 对于每类任务, 我们同时展示了通过率 %Pas. 和解决率 %Res., 分别代表工作流的格式合法性和实际可用性。可以观察到, 所有模型的解决率均低于通过率, 其中 GLM-4.6 的差距最为显著, 平均差异达到 20.96%。尽管 Kimi-K2-Instruct 的差距最小, 仅为 5.21%, 但其绝对值水平均较低, 相较于最高的可见解决率 71.59% 仍显不足。此外, 在开发者场景中, 其表现也出现了明显下滑。从具体任务场景来看, GLM-4.6 在教育任务场景中的差距更是达到了惊人的 43.44%。我们得出结论: 仅满足格式要求的合法工作流表示距离成功执行仍有很大距离。前者仅是实现后者的基本且必要条件, 而非充分条件, 这往往导致得分被虚高。

Q2: 现有的大模型智能体在多大程度上接近真实 workflow 设计专家的能力? 由于种子数据集是从人们常用典型 workflow 中收集的, 同时结合通过交互对话生成可执行工作流的机制, 以及使用文件作为输入和输出的要求, 我们的基准在很大程度上能够代表现实场景。仅从 workflow 能否解决最终问题的角度来看, Gemini-3-Pro-Preview 的平均表现最佳, 其绝对性能达到 71.59%, 但仍远未达到实际可部署的 workflow 规划器水平。即使表现最好的开源模型 GLM-4.7 在解决率上也仅达到 55.98%, 在开发者

Model	Research		Document		Enterprise		Developer		Education		AIGC		Average	
	%Pas.	%Res.	%Pas.	%Res.	%Pas.	%Res.	%Pas.	%Res.	%Pas.	%Res.	%Pas.	%Res.	%Pas.	%Res.
Closed-Sourced														
GPT-5.1	48.89	45.19	53.33	42.22	50.00	33.33	20.83	12.50	36.36	27.27	57.41	55.56	47.26	39.38
GPT-5.2	68.89	60.74	66.67	55.55	86.11	58.33	70.83	45.83	75.76	62.63	48.15	45.68	67.51	54.71
Claude-Sonnet-4.5	68.89	62.97	75.56	64.44	61.11	33.33	75.00	33.33	90.91	76.77	62.97	49.38	71.31	54.57
Gemini-3-Pro-Preview	77.78	77.78	82.22	77.04	86.11	75.93	70.83	52.78	90.91	72.73	74.08	66.67	80.17	71.59
Open-Sourced														
Qwen-3-8B	40.00	21.48	8.89	4.45	8.33	0.00	12.5	2.78	3.03	0.00	7.41	7.41	13.92	6.89
Qwen-3-14B	26.67	21.48	33.33	17.78	13.89	0.00	20.83	12.50	27.27	5.05	35.18	25.31	27.43	15.19
Qwen-3-32B	31.11	26.67	28.89	14.07	30.56	8.33	50.00	27.78	33.33	28.28	48.15	33.34	36.71	23.35
Qwen-3-235B-A22B	35.55	32.59	51.11	38.52	19.45	2.78	62.50	29.17	39.39	32.32	42.59	27.78	40.93	27.71
Qwen-3-Coder-480B-A35B-Instruct	53.33	39.26	57.78	40.00	25.00	1.85	25.00	0.00	45.45	26.26	37.03	32.71	42.19	26.44
GLM-4.6	62.22	60.00	60.00	45.93	63.89	24.07	45.83	6.94	72.73	29.29	64.82	56.79	62.45	41.49
GLM-4.7	80.00	60.00	62.22	54.81	72.22	47.22	41.67	29.17	78.79	65.66	70.37	65.43	69.20	55.98
Deepseek-V3.1	51.11	51.11	51.11	36.29	61.11	26.85	20.83	4.17	45.45	39.39	62.96	55.56	51.48	39.24
Deepseek-V3.2	51.11	43.70	46.67	36.30	27.78	4.63	16.67	4.17	42.42	29.29	57.41	53.09	43.46	32.49
Kimi-K2-Instruct	51.11	48.89	37.78	33.33	11.11	8.33	29.17	9.72	21.21	15.15	51.85	48.76	36.29	31.08
Kimi-K2-Thinking	55.56	50.37	66.67	52.59	52.78	25.93	62.50	33.33	27.27	19.19	72.22	61.11	57.81	43.46

表 1: 六种任务领域中，4 个闭源模型和 11 个开源模型的实验结果。通过率（%Pas.）衡量格式正确性，而求解率（%Res.）衡量实际问题求解能力。

场景中仍存在显著困难。考虑到所考虑的结点类型数量有限，现实中的 workflow 创建本质上更为复杂；因此，Gemini-3-Pro-Preview 在涉及更多结点的复杂 workflow 上表现不足是理所当然的。

Q3：如何提升大模型智能体的 workflow 生成能力？ 1) 首先，通过分析 Qwen-3 系列模型的性能表现，可以评估模型参数大小对 workflow 生成的影响。从平均指标来看，当模型参数量从 8B 增加到 32B，再到 235B 时，性能提升稳定且显著。由此可见，更大的模型参数规模有助于 workflow 生成。2) 其次，后训练是提升模型在特定领域表现的常用方法。在此领域中，确实能使生成的 workflow 更符合格式规范。然而，较高的通过率并不一定意味着较高的解决率。例如，在企业场景下，GLM-4.6 的通过率高于 GLM-4.7，分别为 45.83% 和 41.67%，但在解决率方面，GLM-4.7 反而更高，达到 29.17% 对比 6.94%。因此，仅依赖后训练来增强格式合规性，可能无法有效提升 workflow 的问题求解能力。3) 最后但同样重要的是，观察发现指令型模型（instruct model）的表现劣于思考型模型（think model）。随着模型规模增

大，Qwen-3-Coder-480B-A35B-Instruct 的解决率低于 Qwen-3-235B-A22B，分别为 26.44% 和 27.71%。同样，作为 Kimi-K2 系列的成员，Kimi-K2-Thinking 在所有任务领域均持续优于其对应的指令型模型。这表明，思考机制能够有效促进 workflow 生成能力。

3.3 从聊天互动中分析用户需求变更

每个任务会经历 2 到 4 轮对话，每轮对话被视为一个子任务。每轮的工作流生成需求将基于历史对话，对现有结构进行增删或修改，这进一步考验了模型在复杂任务理解与长时序指令遵循方面的能力。

为防止表示偏差——因为仅有 4 个任务包含四个对话轮次——我们展示了每个模型在前三个轮次中两个评估指标的变化趋势，其中每个数据点为三次实验运行的平均值（图 4）。从表格中可以直观地看出，对于大多数模型，随着交互轮次的增加，生成工作流的质量呈现稳定下降的趋势，这印证了上述挑战。GPT-5.2 模型在第二轮到第三轮之间出现的轻微提升，是由于部分仅包含两轮交互的任务影响了第二轮但未影响第三轮所致。

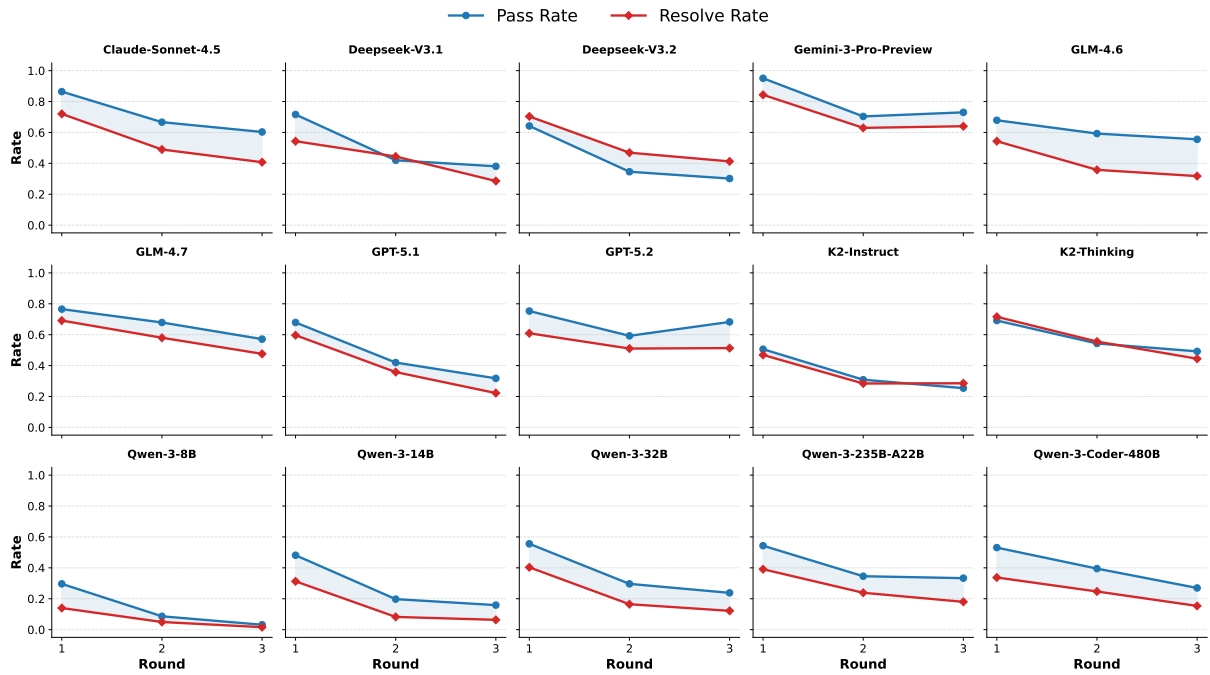


图 4: 对话轮次中的性能下降。我们展示了所有 15 个模型在前三个对话轮次中的通过率和解决率。随着交互轮次的增加, 大多数模型在这两项指标上均呈现稳定下降趋势, 表明在需求不断变化的情况下保持工作流质量具有挑战性。

此外, 可以观察到在大多数模型中, 随着轮次的推进, 折线对应的斜率绝对值逐渐减小, 这一现象可能解释为边际收益递减。在第二轮仍能被合法解析并成功执行的工作流, 已经过一轮筛选, 具备了一定的理解工作流需求变化的能力。随着修改轮次的增加, 性能退化的速率将降低, 这体现了质变与累积性量变之间的区别。

3.4 案例分析

我们以教育场景中的 StudyPlanner 任务为例进行分析。在前两轮中, 给出的指令如下。

第一轮: 创建一个学习路径规划工作流。用户将提供一段描述性指令 (变量 **instruction**) 作为输入。任务是从输入中提取四个核心字段: 感兴趣的学科学习领域、学习目标、学习偏好和平均学习时长。随后, 基于提取的信息, 为用户提供一份详细的自学提升计划, 要求以 Markdown 格式输出。该工作流需要输出自学计划 (变量 **plan**)。

第二轮: 基于现有基础进行修改。工作流

需要能够自动生成一套完整的教程。首先生成课程大纲, 然后按章节迭代生成知识点。内容要求严谨, 包含丰富的示例、优缺点分析以及注意事项。工作流只需输出通过模板转换整合后的最终教程 (变量 **tutorial**)。

在第二轮中, 不同模型给出的响应如图 5 所示。具体而言, GPT-5.2 在 **Dify** 与 **Coze** 平台上生成的可视化工作流分别在附录中的图 8 和图 9 中展示。1) Kimi-K2-Instruct 输出了错误的边连接关系。id 为“5”的结点是一个“迭代”结点, 其内部包含从“iteration-start”开始的子工作流。然而, 二者之间仅存在包含关系, 不存在边连接关系。这一点已在结点文档中明确强调。因此, 生成的工作流实际上无法运行。2) GPT-5.2 生成了合法且可执行的工作流, 并成功解决了给定案例。3) GLM-4.6 输出的内容存在逻辑矛盾。工作流中出现了“code”和“iteration-start”结点, 但这些结点未在 `<node_selection>` 中声明, 表明该模型在工作流创建逻辑上的理解存在缺陷, 此例中表现出了不连贯性, 导致最终生成的工作流实

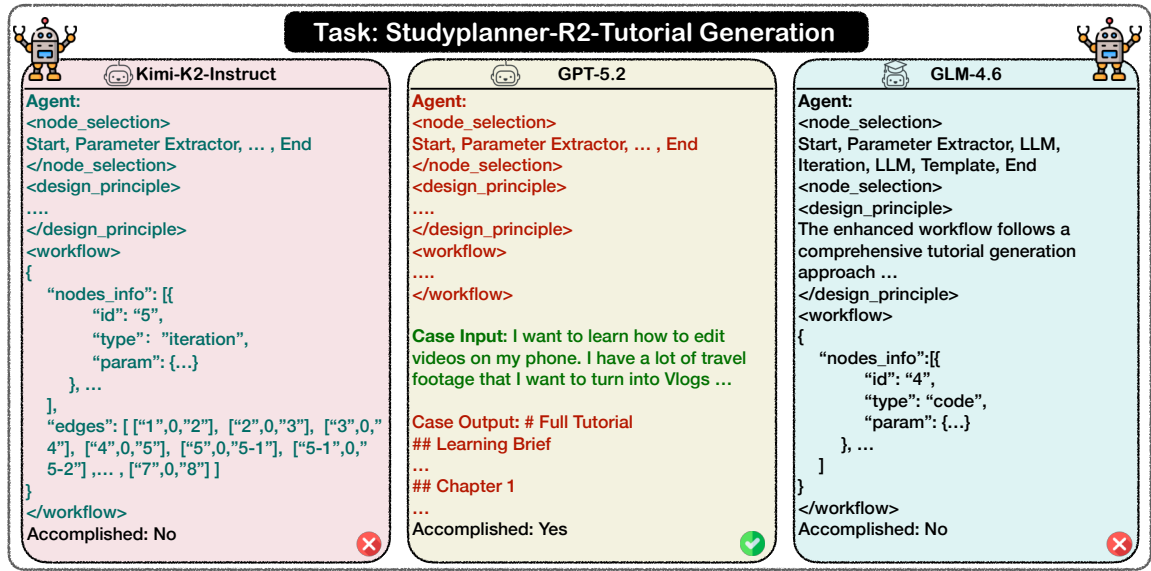


图 5: StudyPlanner 任务的错误案例分析。我们对比了三种代表性模型的输出：Kimi-K2-Instruct 在迭代结点中生成了无效的边连接；GPT-5.2 生成了正确且可执行的工作流；GLM-4.6 输出的结点声明存在逻辑不一致。这反映了工作流生成中的常见失败模式。

际上不可用。

3.5 基于错误驱动的 Agentic 基准

Model	Pass Rate		Resolve Rate	
	Zero-Shot	Agentic	Zero-Shot	Agentic
GPT-5.1	47.26	64.14	39.38	44.31
GPT-5.2	67.51	78.06	54.71	60.05

表 2: GPT-{5.1, 5.2} 在 zero-shot 与所提出的 Agentic 情景下的性能对比。

为解决第 3.4 节及后续章节中识别出的瓶颈问题，我们提出了一种基于错误驱动的 Agentic 基准方法，以探索大模型在该基准测试上的性能上限。该方法通过 OpenCode 框架实现，从原始的 zero-shot 提示逐步过渡到结构化的 SKILL-based 范式，明确整合了任务关键性指南，以及多轮交互和变量引用规则。

为缓解后续交互轮次中的上下文衰减问题，我们动态地从先前的工作流迭代中提取变量摘要，作为补充上下文。至关重要的是，我们构建了一个稳健的执行环，包含五次重试机

制，并辅以全面的结构与语义验证。一旦失败，框架将触发针对性的自动修复模块，纠正四种最常见的错误：代码围栏格式错误、JSON 解码失败、拓扑排序违规以及结点选择不一致。如表2所示，智能体框架带来了显著改进，使 GPT-5.1 和 GPT-5.2 的解决率分别提升了绝对值 4.93% 和 5.34%。

4 相关工作

4.1 基于大型语言模型的智能体

大语言模型 (LLM) 的兴起极大地推动了通用人工智能 (AGI) 的发展，为以大语言模型为中心的人工智能智能体提供了能力基础。由于基于大语言模型的智能体能够充分发挥大语言模型强大的任务理解能力，因此在通用工具 (Schick et al., 2023; Patil et al., 2024)、协作框架 (Hong et al., 2024; Qian et al., 2023) 和广泛的 API 仓库 (Qin et al., 2024; Tang et al., 2023; Zhang et al., 2024a; Yang et al., 2024) 的辅助下，被广泛应用于解决复杂的现实世界问题。许多基于提示的方法 (Wei et al., 2022;

Yao et al., 2022; Zhu et al., 2026) 被证明对提升性能有效。然而, 这些智能体方法或框架主要关注端到端的效果, 对任务求解过程中中间环节的要求与范数 (如规划和推理路径) 关注较少, 这不利于稳定可靠的执行, 也难以实现跨领域结果的可复现性 (Ge et al., 2024; Xie et al., 2024; Liu et al., 2024; Yang et al., 2026; Liang et al., 2026)。

4.2 工作流与语言智能体规划

作为中间状态, 工作流提供了一种连接任务目标与具体可执行步骤的桥梁。一方面, 通过将任务分解为若干可执行的原子步骤, 并基于形式化的逻辑关系 (van der Aalst, 1997; Dijkman et al., 2008) 将其组织成执行计划, 工作流实现了推理路径的完全透明, 有效提升了结果的可解释性与可靠性。另一方面, 引入先验组织结构使得能够利用既有的模式 (van der Aalst et al., 2003; Commoner et al., 1971) 和机器人流程自动化 (Ivančić et al., 2019; Hofmann et al., 2020) 构建复杂任务的逻辑。早期的工作流依赖于细致的手动设计以避免幻觉问题。然而, 手工构建的工作流耗时且费力。因此, 许多研究已转向利用大语言模型实现工作流的自动化生成 (Shen et al., 2023a; Zeng et al., 2023), 包括迭代合成 (Zhang et al., 2024b; Li et al., 2024) 和知识增强型规划框架 (Ye et al., 2023; Zhu et al., 2024; Wornow et al., 2024; Huang et al., 2024)。更多相关工作详见附录 C。

5 结论

我们提出了 Chat2Workflow, 这是一个用于评估大模型从自然语言生成可部署可视化工作流能力的基准。实验表明, 即使在引入智能体框架后, 当前模型在结构约束和需求变更下仍显脆弱。我们希望 Chat2Workflow 能为自动化工作流工程的未来研究提供一个真实的试验平台和具体目标, 并有助于缩小语言模型驱动的工作流设计与工业部署之间的差距。

局限性

尽管做出了贡献, Chat2Workflow 仍存在若干局限性。首先, 虽然高质量的数据集经过人工验证, 但当前规模可能无法涵盖复杂工业业务流程中近乎无限多样的逻辑。其次, 我们简化了结点接口以优先保证可执行性, 这可能无法完全反映某些实际部署中所需的复杂参数配置。最后, 当前系统仅包含 20 种高频结点类型。这些特定结点被选中是因为它们能够有效覆盖大多数标准工作流场景; 然而, 仍有大量有价值的工具有待纳入。

References

- Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and 1 others. 2024. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9510–9529.
- Frederic G. Commoner, Anatol W. Holt, Shimon Even, and Amir Pnueli. 1971. [Marked directed graphs](#). *J. Comput. Syst. Sci.*, 5(5):511–523.
- Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. 2008. [Semantics and analysis of business process models in BPMN](#). *Inf. Softw. Technol.*, 50(12):1281–1294.
- Shengda Fan, Xin Cong, Yuepeng Fu, Zhong Zhang, Shuyan Zhang, Yuanwei Liu, Yesai Wu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Workflowllm: Enhancing workflow orchestration capability of large language models. *arXiv preprint arXiv:2411.05451*.
- Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, and 1 others. 2024. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36.
- Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stable-toolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*.
- Peter Hofmann, Caroline Samp, and Nils Urbach. 2020. Robotic process automation. *Electronic markets*, 30(1):99–106.

- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. [Metagpt: Meta programming for A multi-agent collaborative framework](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Tian Huang, Chun Yu, Weinan Shi, Zijian Peng, David Yang, Weiqi Sun, and Yuanchun Shi. 2024. Promptrpa: Generating robotic process automation on smartphones from textual prompts. *arXiv preprint arXiv:2404.02475*.
- Lucija Ivančić, Dalia Suša Vugec, and Vesna Bosilj Vukšić. 2019. Robotic process automation: systematic literature review. In *Business Process Management: Blockchain and Central and Eastern Europe Forum: BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 280–295. Springer.
- Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Raymond J. Mooney. 2024. [Cat-bench: Benchmarking language model understanding of causal and temporal dependencies in plans](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 16-16, 2024*, pages 19336–19354. Association for Computational Linguistics.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.
- Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. 2024. [Autoflow: Automated workflow generation for large language model agents](#). *CoRR*, abs/2407.12821.
- Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, and 30 others. 2026. [Skillnet: Create, evaluate, and connect AI skills](#). *CoRR*, abs/2603.04448.
- Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qinkai Zheng, Hao Yu, and 11 others. 2024. [Visualagentbench: Towards large multimodal models as visual foundation agents](#). *CoRR*, abs/2408.06327.
- Melissa Z Pan, Negar Arabzadeh, Riccardo Cogo, Yuxuan Zhu, Alexander Xiong, Lakshya A Agrawal, Huanzhi Mao, Emma Shen, Sid Pallerla, Liana Patel, and 1 others. 2025. Measuring agents in production. *arXiv preprint arXiv:2512.04123*.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.
- Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. [Communicative agents for software development](#). *CoRR*, abs/2307.07924.
- Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Benchmarking agentic workflow generation. *arXiv preprint arXiv:2410.07869*.
- Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. Benchmarking agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [Toolllm: Facilitating large language models to master 16000+ real-world apis](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Haiyang Shen, Yue Li, Desong Meng, Dongqi Cai, Sheng Qi, Li Zhang, Mengwei Xu, and Yun Ma. 2024. Shortcutsbench: A large-scale real-world benchmark for api-based agents. *arXiv preprint arXiv:2407.00132*.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023a. Hugginggpt: Solving ai tasks with chatgpt

- and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.
- Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2023b. [Taskbench: Benchmarking large language models for task automation](#). *CoRR*, abs/2311.18760.
- Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. 2025. Flowagent: Achieving compliance and flexibility for workflow agents. *arXiv preprint arXiv:2502.14345*.
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. [Toolalpaca: Generalized tool learning for language models with 3000 simulated cases](#). *CoRR*, abs/2306.05301.
- Wil M. P. van der Aalst. 1997. [Verification of workflow nets](#). In *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer.
- Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. 2003. [Workflow patterns](#). *Distributed Parallel Databases*, 14(1):5–51.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Michael Wornow, Avanika Narayan, Krista Opsahl-Ong, Quinn McIntyre, Nigam H Shah, and Christopher Re. 2024. Automating the enterprise with foundation models. *arXiv preprint arXiv:2405.03710*.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.
- Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622*.
- Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R. Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, Heng Ji, and Chengxiang Zhai. 2024. [If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents](#). *CoRR*, abs/2401.00812.
- Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. 2026. [Autoskill: Experience-driven lifelong learning via skill self-evolution](#). *CoRR*, abs/2603.01145.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.
- Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. [Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios](#). *CoRR*, abs/2401.00741.
- Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, and 1 others. 2023. Proagent: From robotic process automation to agentic process automation. *arXiv preprint arXiv:2311.10751*.
- Zhen Zeng, William Watson, Nicole Cho, Saba Rahimi, Shayleen Reynolds, Tucker Balch, and Manuela Veloso. 2023. Flowmind: automatic workflow generation with llms. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 73–81.
- Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalganekar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, and 3 others. 2024a. [xlam: A family of large action models to empower ai agent systems](#). *Preprint*, arXiv:2409.03215.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2024b. [Aflow: Automating](#)

[agentic workflow generation](#). *Preprint*, arXiv:2410.10762.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2024. [NATURAL PLAN: benchmarking llms on natural language planning](#). *CoRR*, abs/2406.04520.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. [Knowagent: Knowledge-augmented planning for llm-based agents](#). *CoRR*, abs/2403.03101.

Yuqi Zhu, Jintian Zhang, Zhenjie Wan, Yujie Luo, Shuofei Qiao, Zhengke Gui, Da Zheng, Lei Liang, Huajun Chen, and Ningyu Zhang. 2026. Lightthinker++: From reasoning compression to memory management. *arXiv preprint arXiv:2604.03679*.

A 详细数据集描述

任务场景的描述如下：

研究包括 5 个任务，重点在于从非结构化数据中系统地获取、理解和整合知识。它们通常以论文、书籍或研究主题作为输入，强调对内容的深入阅读、逻辑组织以及批判性思维能力。

文档包含 5 个任务，其核心价值在于将文件、图像和表格等原始输入变换为可计算、可重用且可下游处理的结构化结果。它们作为数据处理中“AI 预处理层”的重要代表。

企业包含 4 个任务，针对真实的企业场景，旨在通过人工智能实现复杂业务流程的自动化、标准化以及提供决策支持。其输入大多为企业内部或与业务相关的文档。

开发者包含 3 个任务，服务于软件开发、系统设计和技术理解等场景。核心目标是降低技术理解与开发的阈值，提升工程效率。

学习包含 4 个任务，围绕“教与学”的完整生命周期展开，旨在构建从学习规划、教学内容生成、评估到反馈的闭环系统。

AIGC 包含 6 项任务，位于“从 0 到 1 直接生成可直接使用的内容或材料”的核心位置，强调创造性表达和多模态输出能力。输入通常是高度抽象或简短的指令，而输出则是完整、可发布且可消费的内容成果。

B 实验情景

Dify 平台需要提前准备必要的扩展。一组常用模型已安装，以确保工作流可运行且结果可复现。Dify 中的工作流原生以 YAML 格式表示。我们开发了一个基于规则的变换代码框架，用于将语言智能体生成的 JSON 格式工作流转换为有效的 YAML 格式。这确保了智能体可通过简化的 JSON 表示有效生成工作流，同时与 Dify 执行环境完全兼容。此外，统一采用 CoT 推理序列作为输出方式。知识库中包含的结点类型如下：开始、结束、LLM、问题分类器、代码、文档提取器、HTTP 请求、

如果-否则、列表操作符、参数提取器、模板、变量聚合器、迭代、迭代-开始、文本转语音、文本转图像、Mermaid 转换器、Markdown 导出器、Google 搜索和 Echarts。具体而言，问题分类器和参数提取器结点类型严格指定为 Qwen-3-Max (qwen3-max)，而 LLM 结点类型固定为 Qwen-3-VL-Plus (qwen3-vl-plus)。文本转图像结点类型固定使用 Z-image-Turbo (z-image-turbo)，文本转语音结点类型固定使用 GPT-4o-Mini-TTS (gpt-4o-mini-tts)。对于 Qwen-3-VL-Plus，温度 (temperature) 设置为 0.7，最大 token 数 (max_tokens) 为 32768，解码过程中使用的其他超参数均设为默认值。用于工作流生成的系统提示词和 SKILL.md 将 soon 集成至 GitHub 仓库。至于 Agentic 基准方法，我们使用的开源代码框架版本为 1.3.17。

C 自动化工作流评估的更多相关工作

评估大模型生成的工作流质量至关重要。以往的研究尝试在工具学习场景 (Li et al., 2023; Shen et al., 2023b) 和细粒度规划任务 (Chen et al., 2024; Ye et al., 2024; Zheng et al., 2024) 中自动化工作流评估，使用指标来衡量格式有效性与工具一致性。但这些工作仍存在一些局限性：首先，它们大多停留在工作流的抽象表达阶段 (Qiao et al., 2024)，仅检查关键元素，而未考虑实际执行效果 (Guo et al., 2024)、结构依赖关系 (Lal et al., 2024) 或平台特定约束 (Shen et al., 2024)。它们可能识别出明显的格式错误，但无法保证可用性。其次，这些研究聚焦于单轮评估，忽略了需求随时间变化的场景。

你是一个严格的 Dify 工作流评估者。你的任务是仅根据所提供的信息判断工作流设计是否有效。不要做出假设，不要推断缺失的结点，也不要引入任何外部知识。

Inputs

我将提供四个部分：

1. `node_selection`: 工作流中选定的结点列表或描述。它表示设计者声称将使用的结点。
2. `design_principle`: Dify 工作流的设计原则或约束。用于判断整体逻辑一致性。
3. 工作流: Dify 工作流的 JSON 字符串表示。您必须解析此 JSON 以提取实际使用的结点。
4. `gt_nodes`: 一组必须包含的（真实值）结点。这些结点必须被包含才被视为有效。

Evaluation Rules

所有规则都必须满足，结果才为真。如果任何一条规则不成立，结果必须为假。

规则 1: 真实结点覆盖

检查 `gt_nodes` 中的结点类型是否为 `node_selection` 的子集。只要求结点类型存在，无需考虑出现次数。如果 `gt_nodes` 中有任何结点类型在 `node_selection` 中缺失，立即返回 `false`，不再进行进一步检查。

规则 2: 一致性与确切结点匹配

您必须验证以下所有条件：

- a. 逻辑一致性: 结点选择、设计原则和工作流必须在逻辑上保持一致。工作流结构不得违背所陈述的设计原则。
- b. 确切结点集匹配（双向约束）: 从工作流 JSON 中提取实际的结点集。此结点集必须与 `node_selection` 中声明的结点完全匹配。结点类型只需出现即可，无需考虑其出现频率。具体而言：在 `node_selection` 中未声明的结点不得出现在工作流中。在 `node_selection` 中声明的结点必须出现在工作流中。

Note: 1. The type of the "Template" node is "template-transform". 2. 不区分大小写字母。

Final Decision Logic

仅当规则 1 和规则 2 均完全满足时返回真值；若任一条件不满足，则返回假值。不提供部分得分或“基本正确”的结果。

Output Format

确切的

<reason>

[解释评估推理，说明哪些规则得到满足或违反]

</reason>

<result>

[对或错]

</result>

图 6: 评估通过率的提示。

你是一名工作流执行质量的专家评估员。请根据所有可用信息，判断当前工作流的执行结果是否满足本轮指令的要求。

Inputs You Will Be Given

您将获得以下四个部分：

1. 查询：用于创建或修改工作流的历史指令列表。
 - 每轮的指令均基于前一轮，并可能添加、修改或优化工作流行为。
 - 最新的指令代表当前评估轮次的要求。
 - 忽略查询说明中对文件的输入和输出要求。文件部分已单独评估。
2. 输入：当前轮次工作流执行中使用的非文件输入变量。
 - This field may be empty.
3. 输出：当前轮次工作流执行所产生的非文件输出变量。
4. reference_answer：用于表示预期结果的参考答案。
 - It may contain minor omissions.
 - 其格式可能不完全符合所需输出格式。
 - However, its content is considered correct.
 - This field may be empty.

Evaluation Instructions

请评估工作流执行是否符合当前轮次指令的要求，遵循以下原则：

1. 指令对齐
 - 识别查询中最新指令所隐含或明确陈述的需求。
 - 仅当先前的指令仍然有效且未被覆盖时，才予以考虑。
2. 信息利用
 - 根据所有可用信息（查询、输入、输出、参考答案）进行判断。
 - 输入和参考答案都可能为空；仅此情况并不意味着失败。
3. 输出正确性标准
 - 如果存在输出，请判断其内容和格式是否符合当前指令的要求，但不包括与文件相关的要求。
 - 如果存在 reference_answer，将其用作语义参考，而非严格的模板。
4. 处理缺失或空字段
 - 如果输出明显与指令矛盾，则结果应为错误，但与文件相关的要求除外。
 - 如果 reference_answer 为空，则仅依赖指令遵循性和逻辑一致性。
5. 最终裁决规则
 - 如果工作流执行合理地满足了当前轮次指令的意图和要求（与文件相关的要求除外），则返回 true。
 - Otherwise, return false.

****Note:** This rule takes precedence over all the above ——The file variable part in the instruction requirements, whether as input or output, including images, documents, and audio will not be provided in this evaluation. So it cannot be used as the basis for a judgment of 'False'!!! For example, if the instruction or query requires the output of a PDF document but the output does not include the document, this point should be ignored at this time to check whether other aspects meet the requirements. ******

Output Format

确切的

<reason>

[解释评估推理，说明哪些规则得到满足或违反]

</reason>

<result>

[对或错]

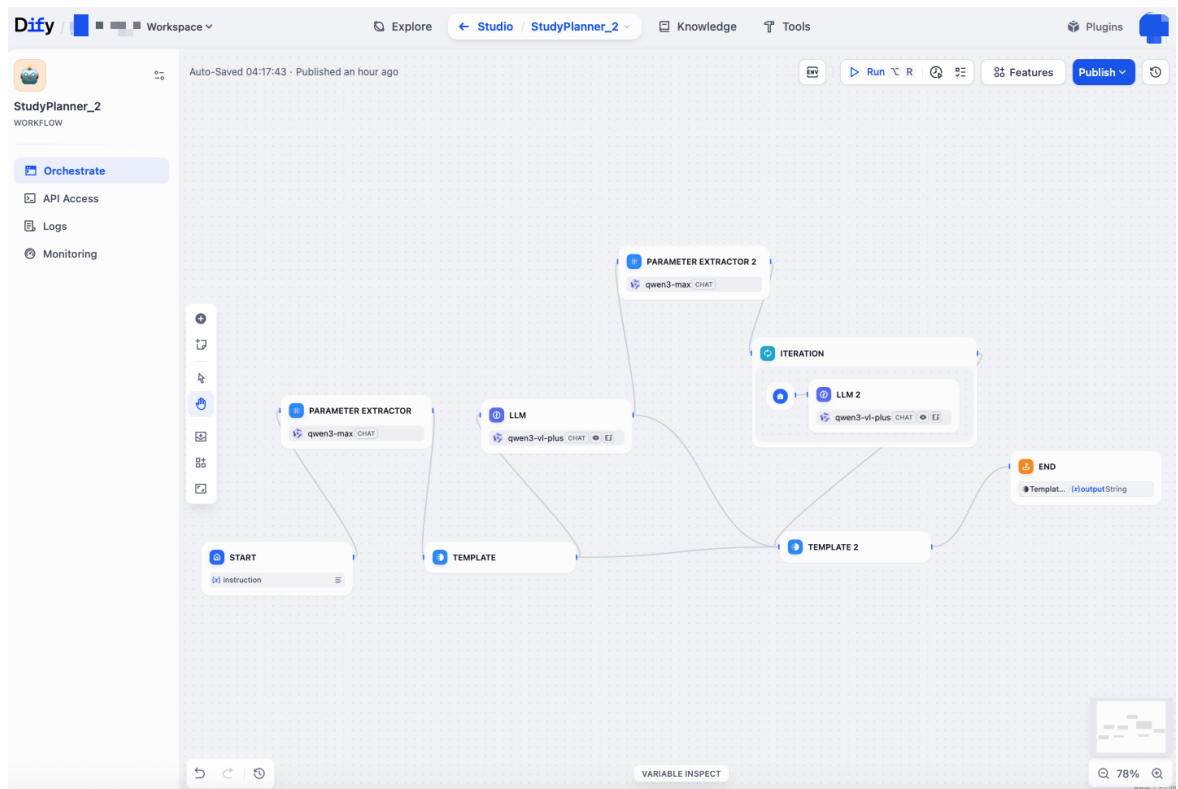


图 8: 由 GPT-5.2 在 Studyplanner 任务第二轮生成的 Dify 工作流。

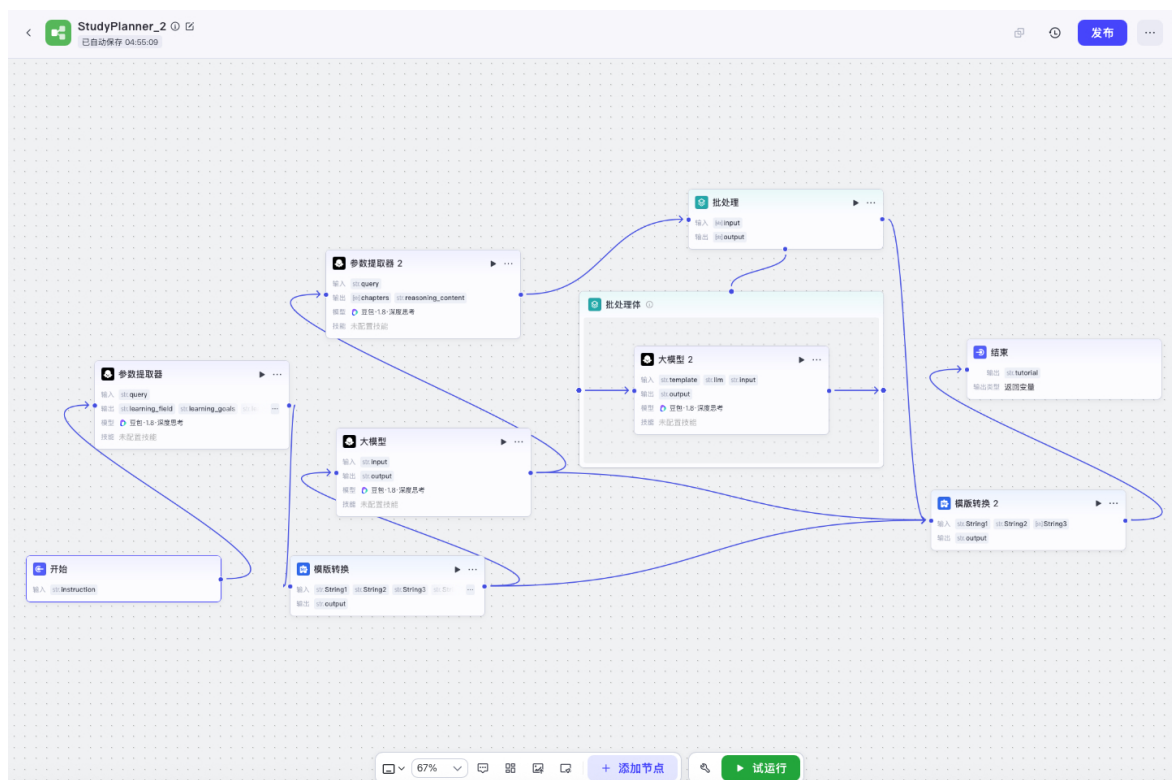


图 9: 由 GPT-5.2 在 Studyplanner 任务第二轮生成的 Coze 工作流。