ChainBuddy: An AI-assisted Agent System for Generating LLM Pipelines

Jingyue Zhang
Montréal HCI
Université de Montréal
Montréal, Quebec, Canada
jingyue.zhang@umontreal.ca

Ian Arawjo
Montréal HCI
Université de Montréal
Montréal, Quebec, Canada
ian.arawjo@umontreal.ca

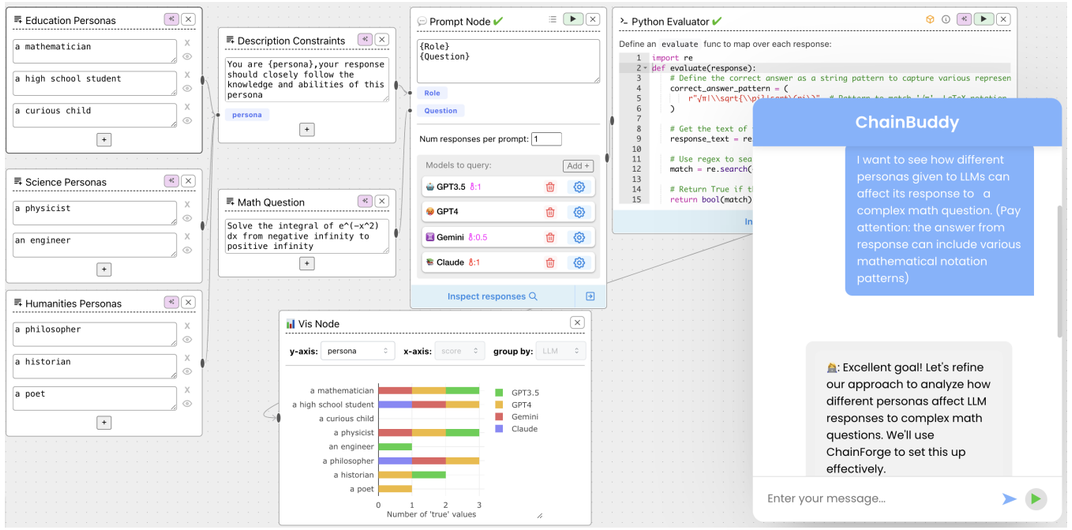

Figure 1: A zero-shot example of a workflow created by ChainBuddy from a single user prompt. The user wishes to investigate how different personas can affect an LLM's response to a complex math question. ChainBuddy generates an LLM pipeline with example personas, a math problem (the Gaussian integral), templated prompts, comparison across four LLMs, and a Python Code Evaluator to check for the solution in the LLM's output ($\sqrt{\pi}$). The user can tweak the output, such as including more patterns in the regex.

Abstract

As large language models (LLMs) advance, their potential applications have grown significantly. However, it remains difficult to evaluate LLM behavior on user-defined tasks and craft effective pipelines to do so. Many users struggle with where to start, often referred to as the "blank page problem." ChainBuddy, an AI workflow generation assistant built into the ChainForge platform, aims to tackle this issue. From a single prompt or chat, ChainBuddy generates a starter evaluative LLM pipeline in ChainForge aligned to the user's requirements. ChainBuddy offers a straightforward and user-friendly way to plan and evaluate LLM behavior and make the process less daunting and more accessible across a wide range of possible tasks and use cases. We report a within-subjects user study comparing ChainBuddy to the baseline interface. We find that when using AI assistance, participants with a variety of technical expertise reported a less demanding workload, felt more confident, and produced higher quality pipelines evaluating LLM behavior. However, we also uncover a mismatch between subjective and objective ratings of performance: participants rated their successfulness similarly across conditions, while independent experts rated participant workflows significantly higher with AI assistance. Drawing connections to the Dunning–Kruger effect, we discuss implications for

CHI '25, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3714085

ChainBuddy：一种用于生成LLM的AI辅助代理系统管道

张靖岳 蒙特利尔 人机交互 蒙特利尔大学 蒙特利尔, 魁北克, 加拿大
ji
ngyue.zhang@umontreal.ca

伊恩·阿拉乔 蒙特利尔 人机交互 蒙特利尔大学 蒙特利尔, 魁北克, 加拿大
ian.ara
wjo@umontreal.ca

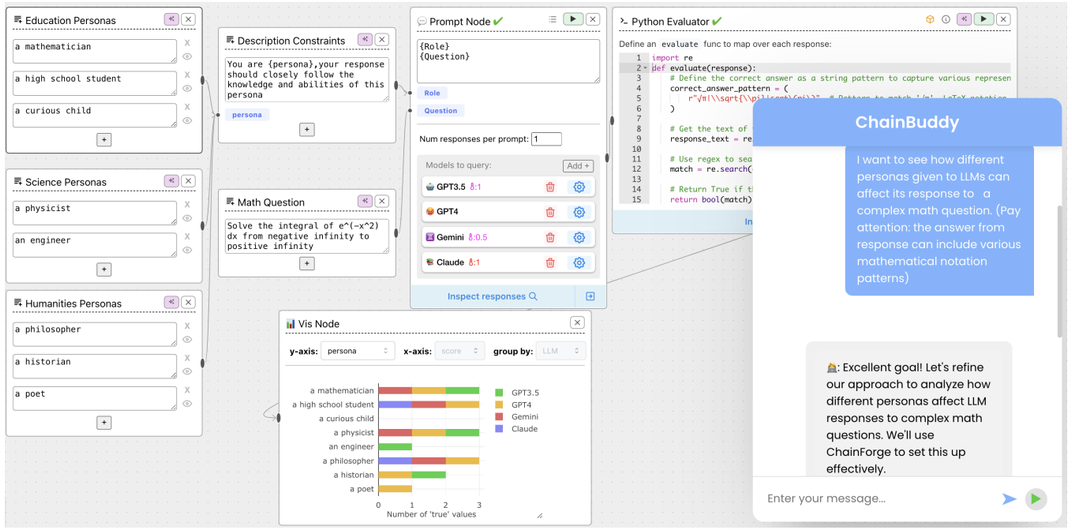

图1：由ChainBuddy根据单一用户提示创建的工作流程零样本示例。用户希望研究不同角色如何影响LLM对复杂数学问题的响应。ChainBuddy生成了一条包含示例角色、一道数学问题（高斯积分）、模板提示、四个LLM的比较，以及一个Python代码评估器用于检查LLM输出中的解决方案（$\sqrt{\pi}$）。用户可调整输出，例如在正则表达式中包含更多模式。

摘要

随着大型语言模型（LLM）的发展，其潜在应用场景已显著扩大。然而，评估LLM在用户定义任务上的行为并构建有效管道仍具挑战性。许多用户面临无从下手的困境，即所谓的“空白页问题”。ChainBuddy作为内置于ChainForge平台的人工智能工作流生成助手，旨在为解决这一问题，ChainBuddy能够根据单一提示或对话，在ChainForge中生成一个符合用户需求的初始评估性LLM管道。ChainBuddy提供了一种直观且用户友好的方式，用于规划和评估LLM行为，使得这一流程在面对广泛任务和用例时不再令人生畏，更易于上手。我们报告了一项受试者内用户研究，将ChainBuddy与基线界面进行对比。研究发现，在使用AI辅助时，不同技术背景的参与者均反馈工作量减轻、信心增强，且生成的用于评估LLM行为的管道质量更高。然而，我们也发现主观与客观性能评价之间存在差异：参与者在不同条件下对自身成功率的评价相近，而独立专家则对AI辅助下的参与者工作流给出了显著更高的评分。结合邓宁-克鲁格效应，我们探讨了这对

CHI 2025，日本横滨

美国计算机协会 ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3714085

the future design of workflow generation assistants regarding the risk of over-reliance.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**; *Empirical studies in HCI*.

Keywords

language models, AI agents, prompt engineering, automation, LLM pipelines, visual programming environments

ACM Reference Format:
Jingyue Zhang and Ian Arawjo. 2025. ChainBuddy: An AI-assisted Agent System for Generating LLM Pipelines. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3706598.3714085

1 Introduction

Over the past two years, the growing interest in AI has spawned a plethora of tools, APIs, and best practices for creating applications based on LLMs. These advancements include a wide range of techniques, from prompt engineering and LLM evaluation platforms to sophisticated AI agent systems equipped with tool use capabilities [1, 4, 38, 46].

Despite these advancements, many users encounter a significant challenge: the "blank page problem." This problem is characterized by the uncertainty and difficulty of knowing where to begin when using platforms like promptfoo [46] or Flowise [13]. Some researchers have proposed social solutions to this problem, such as the development of prompt repositories and community-sharing mechanisms [51]. Another proposed solution is to create an assistant that helps users get started by providing initial guidance and structure [1, 15].

Taking the latter approach, our solution, ChainBuddy, is an AI-powered workflow generation assistant that automatically generates starter LLM pipelines ("flows") given an initial prompt. ChainBuddy helps users get started with evaluating LLM behavior and setting up chains by providing a starter flow, custom-tailored to their use case, that they can edit and extend. We built ChainBuddy on top of an existing open-source visual environment, ChainForge [1], which was designed for open-ended prompt engineering, LLM evaluation, and experimentation tasks (comparing response quality across prompts and models, setting up data processing pipelines, and establish automated evaluation metrics). By building on top of this foundation, ChainBuddy offers users structured assistance to help them overcome the blank page problem, making it easier to explore, experiment and evaluate the behavior of LLMs across a wide range of use cases. We report a mixed methods usability study of ChainBuddy, comparing the assistant to the baseline interface. We find broad support for ChainBuddy, particularly in its reduction of user effort and requirements-gathering intent elicitation feature, with the majority of participants expressing surprise at the quality of the assistant's capabilities. Our contributions are:

(1) A chat-like AI assistant and agent architecture, ChainBuddy, that chats with the user to understand their requirements and goals, and then generates editable and interactive starter LLM pipelines
(2) A within-subjects, mixed methods usability study investigating the relative advantages and trade-offs of the ChainBuddy assistant versus the baseline interface
(3) Insights for future AI support interfaces and reflections on the risk of user over-reliance on AI assistants for LLM pipeline generation

In particular, our user study reveals two complications for workflow generation assistants: that participants did not appear able to adequately assess the quality of their own work compared to the opinions of expert raters (see Sections 5.1.4 and 6); and that while workflow generation assistants can support onboarding, they do so at the potential risk of user over-reliance on AI outputs. In our Discussion, we reflect on these issues and suggest future work explores techniques and study designs that attend to them.

2 Related Work

Since the release of ChatGPT, the LLM landscape has blossomed into a plethora of proprietary and open-source models, infrastructure, and tooling to support LLM operations (sometimes called "LLMOps"). The unique power of LLMs, alongside their stochastic, nondeterministic nature and some high-profile incidents of bias [47], have raised the question of how best to integrate them into larger software systems in a manner that is robust and safe. How to build "LLM-integrated software" is thus emerging as a unique subdiscipline within software engineering, and comprises a number of operations, from prompt engineering, to systematic evaluations, to chaining LLM calls (introduced by Wu et al. in HCI as "AI chains" [50]) as well as more complex network structures [48], and providing LLMs access to *tools*—the ability to call functions that perform actions on the user's machine. The term "AI agents" has come to be synonymous with the latter two architectures.

To support developers in exploring these new practices, a slew of graphical user interfaces and programming libraries have emerged to fill the gap. Coding APIs like LangGraph, CrewAI, and AutoGen [10, 29, 48] support developers in creating flows of AI agents (LLM-integrated submodules) that pass messages to each other in asynchronous-style collaborative architectures. Tools like EvalLM, PromptMaker, BotDesigner, and promptfoo [20, 24, 46, 51] support prompt engineering, while LLM Comparator and ChainForge go further, supporting cross-model comparison, automated code- and LLM-based evaluations, visualizations, and chaining of prompts [1, 21]. Some coding APIs serve to guard against the unpredictable nature of LLM outputs, such as Guardrails, LangChain, and Instructor [11, 16, 19]. Another pattern is the rise of data flow-based interfaces for LLMOps, such as Flowise, LangFlow, ChainForge and PromptChainer, visual programming environments usually (though not exclusively) targeting app development [1, 13, 31, 49].

The many practices of LLMOps have led to recent proposals to *automate* parts of the process—from synthetic input data generation and mining the internet for datasets [5, 14], to prompt optimization [23, 39], to helping users generate automated evaluators that align with their preferences [36–38]. For instance, DSPy and Teola serve as prompt and chain optimization frameworks [23, 42]. However, there remains a problem at a higher level of abstraction: that users, even AI/ML experts, struggle to set up pipelines and automated evaluations of LLM behavior, with Arawjo et al. concluding that

未来工作流生成助手设计的启示
过度依赖风险。

CCS概念

• **人本计算 → 交互系统与**
工具；人机交互实证研究。

关键词

语言模型, 人工智能代理, 提示工程, 自动化, 大型语言模型
管道, 可视化编程环境

ACM参考格式:
张靖岳 和 伊恩·阿拉乔. 2025. ChainBuddy: 一个AI辅助代理
系统用于生成LLM管道. 发表于CHI人因会议
计算系统（*CHI '25*），*2025*年*4*月*26*日至*5*月*1*日，日本横滨。美国计算机协会，
美国纽约州纽约市，21页。https://doi.org/10.1145/3706598.3714085

1 引言

过去两年间，人工智能领域日益增长的兴趣催生了大量基于大型语言模型的应用开发工具、应用程序接口和最佳实践。这些进展涵盖广泛的技术，从提示工程和大型语言模型评估平台，到配备工具使用能力的复杂人工智能代理系统[1, 4, 38, 46]。

尽管取得这些进展，许多用户仍面临重大挑战："空白页问题"。该问题表现为使用promptfoo[46] 或Flowise[13]等平台时，因不确定性和难度而不知从何入手。部分研究者提出了社交解决方案，例如开发提示库和社区共享机制[51]。另一项提议的解决方案是创建辅助工具，通过提供初始引导和结构帮助用户入门[1, 15]。

采用后一种方法，我们的解决方案ChainBuddy是一个人工智能驱动的工作流生成助手，它能根据初始提示自动生成启动型LLM管道（"流程"）。ChainBuddy通过提供针对用户用例定制、可编辑扩展的启动流程，帮助用户开始评估LLM行为并建立任务链。我们在现有开源可视化环境ChainForge [1], 上构建了ChainBuddy，该环境专为开放式提示工程、LLM评估和实验任务设计（比较不同提示和模型间的响应质量，搭建数据处理管道，

并建立自动化评估指标）。基于此基础，ChainBuddy为用户提供结构化辅助，帮助他们克服空白页问题，更轻松地探索、实验和评估各类用例中的LLM行为。我们报告了一项关于ChainBuddy的混合方法可用性研究，将该助手与基线界面进行比较。

我们发现ChainBuddy获得广泛支持，尤其体现在降低用户操作负担及需求收集意图激发功能方面，

大多数参与者对助手能力的质量表示惊讶。我们的贡献包括：

(1) 一个类聊天人工智能助手及代理架构ChainBuddy，
通过与用户对话来理解其需求和目标，随后生成可编辑且交互式的初始LLM管道
(2) 一项受试者内混合方法可用性研究，旨在探究
ChainBuddy助手相较于基线界面的相对优势与权衡
(3) 对未来人工智能支持界面的见解与反思
关于用户过度依赖AI助手进行大型语言模型
流水线生成的风险

特别是，我们的用户研究揭示了工作流程生成助手的两个复杂问题：参与者似乎无法

无法充分评估自身工作质量与
专家评分者意见之间的差距（参见第5.1.4节和第6节）；且

虽然工作流生成助手能辅助入职培训，但
这可能导致用户过度依赖AI输出的风险。在

过论部分，我们反思了这些问题并建议未来工作
探索能应对这些问题的技术及研究设计。

2 相关工作

自ChatGPT发布以来，大型语言模型领域已发展出大量专有及开源模型、基础设施和支持大语言模型运维的工具（有时称为"LLMOps"）。大型语言模型的独特能力，连同其随机性、

非确定性的本质及一些备受关注的偏见事件[47], , 引发了如何以健壮且安全的方式将其集成到更大型软件系统中的问题。因此，如何构建"大语言模型集成软件"正在成为软件工程中一个独特的子学科，涵盖从提示工程到系统评估、

再到链式大语言模型调用（由Wu等人在人机交互领域以"AI链"形式提出）[50]）以及更复杂的网络结构[48], , 并为大型语言模型提供工具访问能力——即调用在用户机器上执行操作的函数。术语"人工智能代理"已成为后两种架构的代名词。

为支持开发者探索这些新兴实践，大量图形用户界面和编程库应运而生以填补空白。诸如LangGraph、CrewAI和Auto-Gen [10, 29, 48] 等编码应用程序接口帮助开发者创建人工智能代理（集成大型语言模型的子模块）流程，使这些代理能在异步协作架构中相互传递消息。而EvalLM、

PromptMaker、BotDesigner和promptfoo [20, 24, 46, 51] 则专注于提示工程支持，LLM比较器与ChainForge更进一步，提供跨模型比较、基于代码和大型语言模型的自动化评估、可视化以及提示链式操作[1, 21]。部分编码应用程序接口如Guardrails、LangChain和In-structor [11, 16, 19]用于防范大型语言模型输出的不确定性。另一种趋势是基于数据流的LLMOps界面兴起，例如Flowise、LangFlow、ChainForge和PromptChainer这类通常（但非绝对）面向应用开发的可视化编程环境[1, 13, 31, 49]。

大语言模型运维（LLMOps）的诸多实践催生了近期多项旨在
实现流程部分自动化的提案——从合成输入数据生成
到互联网数据集挖掘[5, 14], 再到提示优化
[23, 39], 直至帮助用户生成符合其偏好的自动化评估器
[36–38]。例如，DSPy和Teola作为
提示与链式优化框架[23, 42]应运而生。然而，
在更高的抽象层次上仍存在一个问题：即用户，
即便是AI/ML专家，也难以建立管道和自动化
LLM行为评估，其中Arawjo等人得出结论认为

"more work needs to be done on [the] conceptualization and planning aspects" of supporting users in LLM pipeline creation [1, 51]. Part of the issue is certainly the usual difficulty of learning a new interface, but the larger issue is conceptual: what is the *"right way"* to prompt engineer? To set up a pipeline? To evaluate LLM behavior? We and the community are still learning these best practices.

The problem of pipeline generation bears a similarity to AutoML, an area of research in machine learning that focuses on automatically designing machine learning (ML) pipelines to train new ML models, whether in part or in full [3, 12, 18]. Certainly, AutoML faces unique challenges compared to the typical problems in the LLMOps space (such as managing very large training datasets, deciding upon hyperparameters and weighing trade-offs in terms of expected training cost and performance). Inspired by this line of ML research, here we investigate the *end-to-end generation of LLM pipelines*, an emerging research area we call **AutoLLMOps**: from a single user prompt, can we generate an *inspectable*, *interactive*, and *editable* pipeline, complete with input data, prompt(s) and model(s), and even automated evaluations? Could we build such a system for open-ended tasks? What benefits would users derive from it? And what dangers are there, if any, to automation? This problem bears a similarity to previous work on crowdsourcing-based task decomposition [25, 27]. With new LLM agent frameworks, emerging research is showing that this kind of end-to-end generation of a workflow is possible for tasks like descriptive analytics given a user-provided dataset [4], visual programming [53], AI agent workflows [9], or for chatbot creation [35]. Here, we specifically focus on generating LLM pipelines that help users evaluate LLM behavior.

3 ChainBuddy System Design

To automate the creation of LLM pipelines, we focused on creating a flexible, user-friendly interface that could support a wide range of use cases beyond just prompt engineering. We decided to build a chatbot-style assistant interface, ChainBuddy, within the open-source ChainForge platform [1]. This choice was driven by the need for an intuitive and interactive environment that can support various users across a range of different pipelines, such as data processing, prompt optimization, LLM auditing, and more. Our goal was to ensure that ChainBuddy could handle diverse and complex requirements while being accessible to users with varying levels of technical expertise and use cases. Note that our ultimate goal is to extend ChainBuddy to guide users beyond initial generation, i.e., to edit existing flows; however, due to the complexity of the agent system and open-ended nature of the problem, we consider here only the ability of ChainBuddy to generate flows from scratch.

3.1 Interface and Example Usage

The ChainBuddy assistant can be seen in Figure 2. The assistant comprises a standard chat interface in the bottom-left hand corner of the ChainForge platform. The user starts a chat with the assistant to explain their problem (Fig 2a). The assistant then holds a Q&A conversation with the user to disambiguate user intent [33, 43] (Fig 2b). This comprises a pass where the assistant asks a set of up to three questions, and the user can respond individually to each question by filling out a form (Fig. 2c). At any time, the user may end the disambiguation and trigger the AI to generate a flow by clicking the button (Fig. 2d). The user can then inspect the generated flow or request a new generation (Fig. 2e). We kept the assistant interface simple as the majority of our contribution's complexity lies in the agent architecture and flow generation capabilities.

Note that in this paper, we focus on explaining and showing the ChainBuddy interface and workflow, rather than the baseline interface's built-in features like the Response Inspector (table of LLM responses) and different nodes. We point readers unfamiliar with the ChainForge platform to the public documentation or the paper [1].

3.2 System Architecture

ChainBuddy is built on LangGraph [29], a library designed for constructing stateful, multi-actor applications with LLMs. LangGraph's core benefits include its ability to handle cycles, provide fine-grained controllability, and ensure persistence. These features are essential for creating reliable agent-based workflows that can support advanced human-in-the-loop and memory functionalities. We use Anthropic's Claude 3.5 Sonnet for the front-end requirements-gathering agent, and OpenAI's GPT-4o for all agents in the backend.

3.2.1 **Requirement gathering.** The design of the requirement gathering agent for the ChainForge platform draws inspiration from the Chain of Reasoning [40] prompting methodology adapted from the open-source GitHub project Professor Synapse, an "AI guide designed to help users achieve their goals" [41]. The structured interface for intent elicitation was inspired by ExploreLLM [33]. The agent employs a dictionary that updates context about the primary user goal, a list of current requirements the solution should address, and other user preferences. This structure is updated throughout the agent's interactions with the user. Overall, the agent poses three types of targeted questions to refine understanding:

(1) **Goal Clarification Questions**: These questions help to understand the overall objectives the user is aiming to achieve, as well as the context around their problem.
(2) **Requirements Exploration Questions**: Designed to uncover specific needs, constraints, and fine-grained requirements that align with the user's goals.
(3) **Disambiguation Questions**: Ask for any clarifications to address ambiguities or contradictions between the overall user goal and the requirements.

This interactive approach ensures flexibility in accommodating changes as new insights or constraints emerge during the iterative dialogue. The requirement gathering agent also aims to help users better understand and reflect on their needs and goals before precise messages are sent to the back-end workflow-generating agents, which are relatively time-consuming and costly compared to the chat.

3.2.2 Workflow generation. Inspired by concepts from *Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models* [45] and projects like Baby-AGI [34], we designed ChainBuddy's agentic system to generate long-term plans based on user requirements (Figure 3). This approach involves breaking down each task into specific, manageable actions that can be executed by individual agents which return structured data to upstream agents (i.e., JSON). This design allows each agent to

"在[支持用户进行LLM流程创建的] 概念化与规划层面[1, 51]仍需投入更多工作"

部分问题确实源于学习新界面的常见难度，但更大的问题是概念性的：什么是提示工程的"正确方法"？如何搭建管道？如何评估LLM行为？我们和社区仍在学习这些最佳实践。

流水线生成问题与自动机器学习（AutoML）存在相似性，这是机器学习中的一个研究领域，专注于自动设计机器学习（ML）管道以训练新的ML模型，无论是部分还是全部[3, 12, 18]。当然，与LLMOps领域的典型问题（如管理超大规模训练数据集、确定超参数及权衡预期训练成本与性能）相比，AutoML面临着独特挑战。受此机器学习研究方向的启发，我们在此探索LLM管道的端到端生成——这一新兴研究领域我们称之为AutoLLMOps：仅凭单一用户提示，能否生成可检查、可交互且可编辑的完整管道，包含输入数据、提示及模型，

甚至包括自动化评估？我们能否为开放式任务构建这样的系统？用户能从中获得哪些益处？自动化又存在哪些潜在风险？这一问题与先前基于众包的任务分解研究[25, 27]有相似之处。随着新型LLM代理框架的出现，新兴研究表明，针对用户提供的数据集[4], 进行描述性分析等任务，这种端到端工作流程生成已成为可能，包括可视化编程[53], 、AI代理工作流[9], 或聊天机器人创建[35]。在此，我们特别关注生成帮助用户评估LLM行为的LLM管道。

3 链友系统设计

为实现LLM管道的自动化创建，我们专注于开发一个灵活、用户友好的界面，该界面不仅能支持提示工程，还可覆盖更广泛的用例。我们决定在开源ChainForge平台[1]内构建一个聊天机器人风格的助手界面——ChainBuddy。这一选择源于对直观交互环境的需求，该环境需支持不同用户处理各类管道，如数据处理、提示优化、LLM审计等。我们的目标是确保ChainBuddy既能应对多样复杂的任务需求，又能适应不同技术专长水平和用例的用户。需注意的是，我们的最终目标是扩展ChainBuddy的功能，使其不仅能引导用户完成初始生成阶段，还能——

用于编辑现有流程；然而，由于代理系统的复杂性和问题的开放式特性，我们在此仅考虑ChainBuddy从零开始生成流程的能力。

3.1 界面与示例用法

ChainBuddy助手如图2所示。该助手
包含一个位于ChainForge平台左下角的标准聊天界面
用户通过与助手开启聊天
来阐述他们的问题（图2a）。随后助手会进行问答
与用户对话以澄清用户意图[33, 43]
(图2b)。这一过程包含一个环节，助手会提出最多
三个问题，用户可以分别回答每个

问题，通过填写表单(图2c)。用户可随时终止并
通过点击消除歧义并触发人工智能生成流程
按钮（图2d）。用户随后可检查生成的流程或
请求重新生成内容(图2e)。我们将助手界面保持
简洁，因为我们贡献的主要复杂性体现在
代理架构和流程生成能力。

请注意，在本文中，我们重点在于解释和展示ChainBuddy界面和工作流程，而非基线

界面内置的功能，如响应检查器（LLM响应
表格）和不同的节点。我们建议不熟悉
ChainForge平台的读者参考公开文档或
论文 [1]。

3.2 系统架构

ChainBuddy构建于LangGraph [29], 之上，这是一个专为构建有状态多参与者应用与大型语言模型设计的库。LangGraph的核心优势包括处理循环的能力、提供细粒度可控性以及确保持久性。这些特性对于创建可靠的基于代理的工作流程至关重要，能够支持先进的人在回路和记忆功能。我们使用Anthropic的Claude 3.5 Sonnet作为前端需求收集代理，后端所有代理则采用OpenAI的GPT-4o。

3.2.1 需求收集。该需求收集代理的

设计灵感源自ChainForge平台，
其采用改编自推理链 [40] 的提示方法。
开源GitHub项目Professor Synapse，一个"AI指南"
旨在帮助用户实现目标" [41]。该结构化
界面用于意图引导的设计灵感来自ExploreLLM [33]。该
代理采用了一个字典，用于更新关于主要
用户目标的上下文、解决方案应满足的当前需求列表，
以及其他用户偏好。这一结构会在整个过程中持续更新
代理与用户的交互过程中，总体而言会提出三类
针对性问题以完善理解：

(1) 目标澄清问题：这类问题用于理解用户希望达成的总体目标，
以及围绕其问题的上下文背景。
(2) 需求探索问题：旨在挖掘与用户目标相符的具体需求、
约束条件和细粒度需求。
(3) 消歧问题：用于请求用户对任何需要澄清之处
解决整体之间的歧义或矛盾
用户目标与需求之间的歧义或矛盾。

这种交互式方法确保了在适应
迭代对话过程中出现新见解或约束条件时的灵活性。
需求收集代理还旨在帮助用户
在精确表达前更好地理解和反思其需求与目标。
消息被发送至后端工作流生成代理，
与…相比，这些方法相对耗时且成本较高
聊天。

3.2.2 工作流程生成。灵感源自《计划与解决提示：通过大型语言模型[45] 改进零样本思维链推理》的理论框架及*Baby-AGI*[34],等项目

我们设计的ChainBuddy代理系统能根据用户需求生成长期计划（图3）。该方法将每个任务拆解为可由独立代理执行的特定可管理动作，这些代理会向上游代理返回结构化数据（即JSON）。此设计使得每个代理能

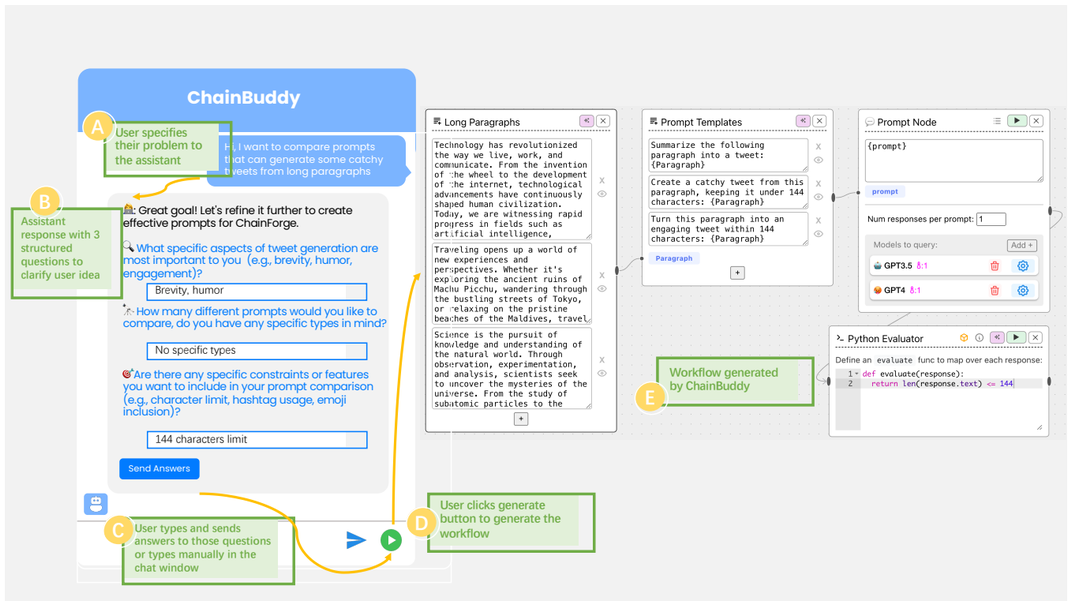

Figure 2: ChainBuddy interface and example usage. Users specify requirements (A), ChainBuddy replies with a requirements-gathering form (B) that users can either fill out and send, or follow up with an open-ended chat (C). User presses green button (D) to indicate that they are ready to generate a flow. After a delay of 10-20sec, ChainBuddy produces a starter pipeline (E). Here, the starter pipeline includes example inputs, multiple prompts to try (prompt templates), two queried models, and a Python-based code evaluator.

focus on a single task, improving efficiency and accuracy. Key architectural features include:

- **Requirement Gathering Chat Assistant**: A chat-focused agent interacts with the user to disambiguate user intent and gain context for their problem, before proceeding to the generation step.
- **Planner Agent**: Takes the specification from the front-end of the user goal, and develops a comprehensive plan for implementation. The Planner is passed contextual information on all nodes in ChainForge that it has access to, their names and descriptions, and how they are allowed to connect. For full details, please see Appendix B.
- **Task-Specific Agents**: Each task in the plan is assigned to a specific agent, allowing for focused execution. Here, a "task" largely maps to different nodes in the ChainForge interface that need to be generated.[1] This specialization can allow utilizing smaller, less powerful models for execution tasks while reserving larger, more capable models for planning.
- **Connection Agents**: These agents take the task-specific output (as JSON data representing ChainForge nodes to add), create edge specifications to connect them and fill in starter x-y positions for nodes.

A detailed walkthrough of how our system goes from initial user requirements to the final workflow, alongside example prompt templates and outputs from individual agents, is presented in Appendix B. Note that the system we present here is limited to a few select nodes in ChainForge: TextFields Node, for defining input data; Prompt Node, for prompting one or more models and prompt templating; Python Code Evaluator and LLM Scorer Nodes for evaluating LLM outputs; and the Vis Node. The system also supports template chaining [1] to compare across prompts; i.e., putting prompt templates inside TextField inputs and chaining them together. Template chaining is primarily useful for comparing across prompts in a structured way. In Appendix B, we cover how consistency in variable names is accomplished via variable traces, which enables template chaining.

3.3 Early Feedback

We designed ChainBuddy through an iterative process of testing internally on new tasks, from prompt comparison, to model comparison, to evaluating LLMs for identity-based bias. In particular, we found that the AI had a tendency to overfit to few-shot examples (e.g., always choosing to evaluate two models, or use specific input

[1]For our usability study, we limited the nodes to: TextFields Node, Prompt Node, LLM Scorer Node, and Python Code Evaluator Node.

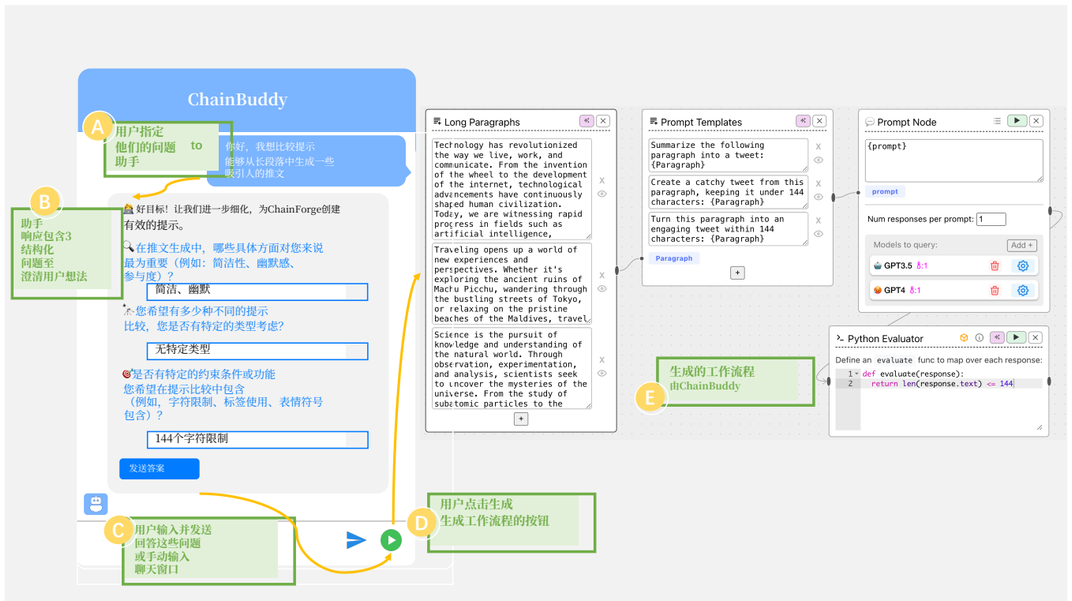

图2：ChainBuddy界面及使用示例。用户指定需求（A），ChainBuddy回复一个需求收集表单（B），用户可填写后提交或继续开放式对话（C）。用户点击绿色按钮（D）表示准备生成流程。经过10-20秒延迟后，ChainBuddy生成初始管道（E）。此处初始管道包含示例输入、多个可尝试的提示（提示模板）、两个查询模型及一个基于Python的代码评估器。

专注于单一任务，从而提升效率与准确性。关键架构特性包括：

- **需求收集聊天助手：一个专注于对话的**代理与用户交互以澄清用户意图并获取问题的上下文，然后继续进入生成步骤。
- 规划代理：从用户目标的前端获取规范，并制定全面的实施计划。规划器会接收关于ChainForge中所有可访问节点的上下文信息，包括它们的名称和描述，以及允许的连接方式。完整细节请参阅附录B。
- 任务特定代理：计划中的每个任务都会分配给一个特定的代理，以实现专注执行。这里的“任务”主要映射到ChainForge界面中需要生成的不同节点。这种专业化可以允许在执行任务时使用较小、功能较弱的模型，同时保留较大、能力更强的模型用于规划。
- 连接代理：这些代理接收任务特定的输出（作为表示要添加的ChainForge节点的JSON数据），创建边规范以连接它们并填充起始内容节点的x-y坐标位置。

附录B详细展示了我们的系统如何从初始用户需求到最终工作流程的全过程，包括示例提示模板和各个代理的输出。需要注意的是，本文展示的系统仅限ChainForge中的部分精选节点：文本字段节点（用于定义输入数据）、提示节点（用于向一个或多个模型发起提示及提示模板化）、Python代码评估器和LLM评分器节点（用于评估LLM输出）以及可视化节点。该系统还支持模板链式调用[1] 以比较不同提示，例如将提示模板放入文本字段输入中并链式串联。模板链式调用主要用于以结构化方式比较不同提示。附录B还介绍了如何通过变量追踪实现变量名称的一致性，从而支持模板链式调用。

3.3 早期反馈

我们通过迭代测试流程设计了ChainBuddy
在内部新任务上，从提示比较、模型对比到评估大型语言模型的基于身份的偏见，特别是，

我们发现该人工智能存在对少样本示例过度拟合的倾向（例如，总是选择评估两个模型，或使用特定的输入

1在我们的可用性研究中，我们将节点限制为：文本字段节点、提示节点、大型语言模型评分器节点，以及Python代码评估器节点。

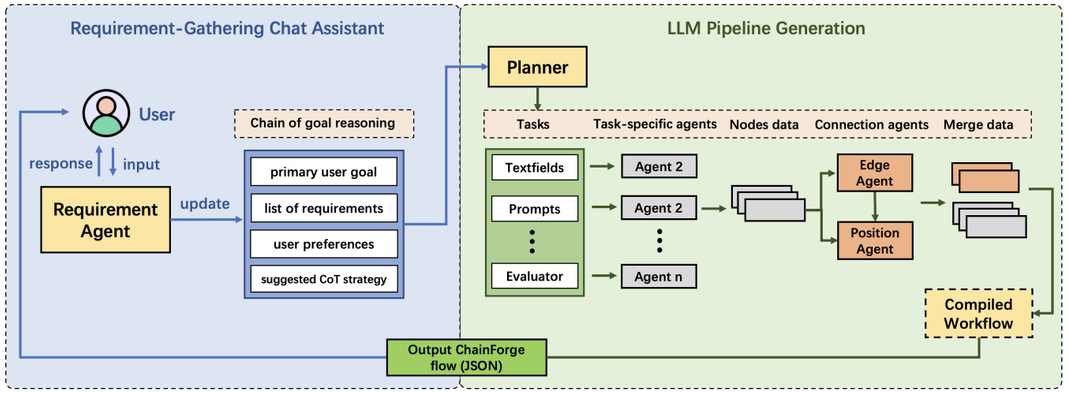

Figure 3: ChainBuddy system architecture. A front-end requirement agent elicits user intent and context (left). When the user presses the generate button (Fig 2d), a specification of user intent is sent to the Planner in the back-end. The Planner agent breaks down the problem into tasks and sends each to a dedicated agent; the outputs are combined through layout-providing connection agents and merged. The final output is passed to the front-end as a complete ChainForge flow (JSON).

data); based on this, we removed few-shot examples in specific places such as the Planner agent prompt. We also conducted informal pilot studies to gather early feedback and improve our system. Some early insights we discovered:

- **Intent disambiguation**: One prompt entered by users did not usually contain enough information to generate a detailed workflow that addresses user's actual needs. Based on this, we opted for a more interactive chat.
- **Structured elicitation**: We implemented ChatGPT-like chat where the system asks users questions. We first tried in 'infinite questioner' approach where the agent keeps asking the user disambiguating questions, akin to Cheng et al. [9]; however, we found this approach slow and stressful to the user, as the user feels pressured to reply to every question, or struggled to reply in a natural way. The LLM output was also often long or listed several questions at once. Based on this feedback, we opted for a structured form-filling approach, presenting three questions at once and letting the user address only the questions they wanted to. We also limited the number and length of questions.
- **Feedback**: Users suggested incorporating features like visualizing the loading progress, as well as providing explainable AI elements to help users understand how the system arrives at certain visualizations or results.
- **Desire to edit existing flows**: Some users wanted to continue the assistant chat, asking ChainBuddy to revise or extend the flow. We too wanted this feature, but felt it was too complex to address within the limits of a single paper.

3.4 Technical Evaluation

Following our iterative design process, we conducted a small-scale technical evaluation to estimate the quality of ChainBuddy's pipeline generation algorithm. Evaluating workflows is an active area of research and is not straightforward: in many situations, as we reflect on in Discussion, there is no one ground truth "best" workflow; what is more, there is no benchmark of evaluative LLM pipelines, and our prototype was deliberately limited its featureset (Section 3.2.2). We thus prepared a set of representative prompts to the system. We used the same three broad categories of use cases found in a prior study of the base interface [1]—Prompt Engineering, LLM Auditing, and Data Processing—with corresponding subtasks (e.g., "comparing across prompts") and three prompts per subtask, for a total of 27 prompts (Appendix A). These prompts were devised such that with its current features, the system should reasonably be able to address them (e.g., we did not include ground truth evaluation with a spreadsheet, as the Tabular Data node was not part of our prototype), but without foreknowledge of how the system would perform. Additionally, the advising author, and not the author who programmed the system, wrote the prompts. Each prompt was provided to ChainBuddy verbatim (no requirements gathering step). The generated workflow was run and exported, with only the layout adjusted for ease of analysis (e.g., moving nodes around, as our layout system is not perfect).

To grade workflow quality, we recruited 6 expert raters who are AI/ML experts with experience prompt engineering,[2] and paid them $75 CAD each for 75 minutes of time. Each rater was assigned one prompt from each subtask, for a total of 9 workflows, with the exact prompts assigned at random in a balanced manner such that two expert raters independently graded each workflow. Raters were asked to grade how well each workflow aligned with the input

[2]In the screening survey, all participants indicated they have "Advanced (extensive experience)" or "Intermediate (regularly work with LLMs)" levels of experience with both LLMs and prompt engineering. In addition, when asked their level of Python programming skill, four wrote a 5 and two a 4 (out of 5), indicating advanced knowledge; this was important for evaluating Code Evaluator nodes. Participants were recruited from an institute for AI research.

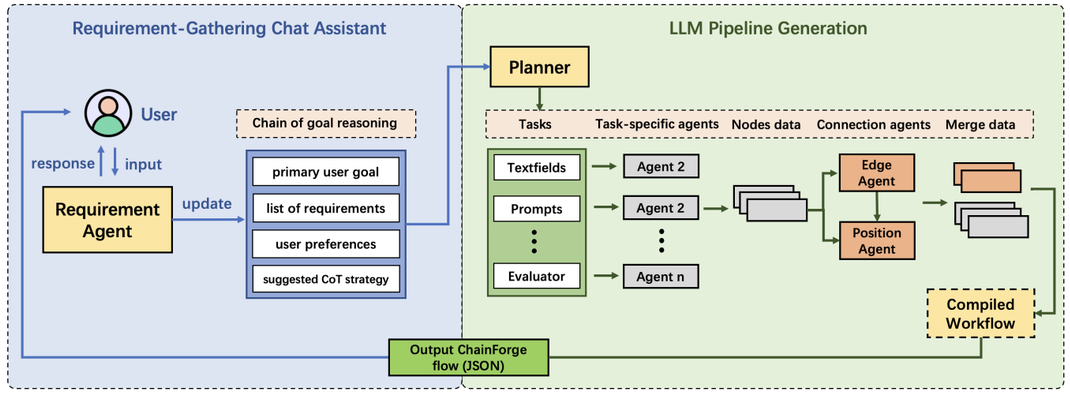

图3：链友系统架构。前端需求代理获取用户意图和上下文（左）。当用户按下生成按钮（图2d），用户意图的规范会被发送至后端的规划器。规划代理将问题分解为任务并分发给专用代理；输出通过布局提供连接代理进行组合与合并。最终输出以完整的链锻造流程（JSON）形式传递至前端。

数据）；基于此，我们在特定位置移除了少样本示例，如规划器代理提示。我们还进行了非正式试点研究，以收集早期反馈并改进我们的系统。

我们早期发现的一些见解：

- 意图消歧：用户输入的单个提示通常不足以生成满足用户实际需求的详细工作流程。基于此，我们选择了更具交互性的聊天方式。
- 结构化引导：我们实现了类ChatGPT聊天，系统会向用户提问。最初尝试了'无限提问者'模式，代理会持续向用户提出消歧问题，类似Cheng等人[9]的方法；但发现这种方式对用户而言速度慢且压力大，用户被迫回答每个问题或难以自然回应。大型语言模型的输出也常冗长或一次性列出多个问题。根据此反馈，我们改用结构化表单填写方法，

  一次性呈现三个问题，让用户仅回答他们想处理的问题。同时限制了问题的数量和长度。

- 反馈：用户建议加入可视化加载进度等功能，并提供可解释AI元素，帮助用户理解系统如何生成特定可视化或结果。
- 编辑现有流程的愿望：部分用户希望继续助手聊天，要求ChainBuddy修订或扩展流程。我们也希望实现此功能，但认为在单篇论文的篇幅限制内解决此问题过于复杂。

3.4 技术评估

遵循我们的迭代设计流程，我们进行了小规模技术评估，以估算ChainBuddy流水线生成算法的质量。

评估工作流是一个活跃的研究领域且并不简单：正如我们在讨论中反思的那样，在许多情况下并不存在唯一的基准真相"最佳"工作流；更重要的是，目前尚无评估性LLM流水线的基准，且我们的原型在功能集上进行了刻意限制（第3.2.2节）。

为此，我们准备了一组具有代表性的系统提示。
我们采用了与基础界面[1]前期研究相同的三大类用例——提示工程、LLM审计和数据处理——及其对应子任务（例如

"跨提示比较"），每个子任务包含三条提示，总计27条提示（附录A）。这些提示的设计原则是：系统当前功能应能合理应对（例如未包含需要电子表格的基准真相评估，因表格数据节点未纳入原型），但事先并不知晓系统实际表现。此外，提示由顾问作者（而非系统编程作者）撰写。每条提示均原样提供给ChainBuddy（无需求收集步骤）。生成的工作流程经运行导出后，仅调整布局以便分析（例如移动节点位置）。

（因为我们的布局系统并不完美）。
为了评估工作流质量，我们招募了6位AI/ML专家作为评分者，他们均具备提示工程经验，并向每人支付

75加元作为75分钟工作的报酬。每位评分者被分配
来自每个子任务的一个提示，总计9个工作流，
具体提示的分配采用随机平衡方式进行，以确保
由两位专家评分者独立对每个工作流程进行评分。评分者
被要求评估每个工作流程与输入需求的匹配程度

2在筛选调查中，所有参与者均表示具备"高级（丰富经验）"或"中级（经常使用大型语言模型）"级别的大型语言模型及提示工程相关经验。此外，当问及Python编程技能水平时，四人自评为5分，两人为4分（满分5分），表明具备高级知识；这对评估代码评估节点至关重要。参与者通过来自人工智能研究所。

requirements on a scale of 1-5 (Appendix A.1), and to provide qualitative feedback and reasoning for their score. The total 27 prompts are listed in Appendix A.2, alongside quality scores averaged across two independent raters, and a summary of qualitative feedback.

Across all categories, the average rating is 4.09 with a standard deviation of 0.7, indicating the vast majority of ratings are within the 3-5 range. A score of 4 indicates that the average workflow generated by ChainBuddy was aligned to the user requirements, with only 1-2 minor issues that were easily fixable (Appendix A.1). Notably, only one workflow obtained an average score less than 3 (prompt Q3.3.1). Running a one-way ANOVA on the average scores, comparing across categories, is not significant (p=0.42); thus, ChainBuddy had similar performance across categories considered here. 22 out of 27 ratings were within one point of each other, with raters expressing slight agreement (weighted Cohen's kappa = 0.17).

Overall, this indicated that ChainBuddy was strong enough to proceed to a user study; yet also indicated areas for improvement. From the qualitative feedback, the most consistent issue across 14 workflows concerned the quality of the evaluator(s) implemented and chosen: LLM Scorers could have prompts with criteria that were slightly too vague; and Code Evaluators could be "too stringent" and struggle to generalize (e.g., to different formatting of LLM outputs). Beyond this, 6 issues were related to the Vis Node plot, specifically not plotting by a certain expected variable like prompt or prompt variable. This was caused by an oversight on our part: our ChainBuddy prototype only added a default Vis Node which by default plots by LLM; while the broader ChainForge interface allows users to change the variable. Although we also saw minor problems with input data and phrasing of prompts, this feedback indicates that most challenges stem from choices ChainBuddy makes in the evaluation part of LLM pipelines. Note that aligning evaluators with user expectations is non-trivial, with one recent solution arguing the necessity of a human-in-the-loop approach to align generated evaluators through user grading of LLM outputs [38].

4 Usability Study

To evaluate ChainBuddy, we ran a within-subjects, mixed-methods user study against the baseline interface (ChainForge without ChainBuddy), since it was the most direct comparison to a "manual" open-ended system for setting up LLM pipelines. Our goals were broadly focused on how people would want to use an AI assistant for generating evaluations of LLM behavior; specifically, for our qualitative evaluation:

(1) What aspects of the assistant do they appreciate the most, compared to the baseline?
(2) Do users find the requirements-gathering interaction helpful or necessary?
(3) What kinds of problems do participants want to use ChainBuddy for? (free exploration task)
(4) Do people feel that their ideas or hypotheses changed after interacting with the assistant?
(5) How do people edit the generated flows? What kinds of edits do they tend to make?
(6) Do people learn anything from interacting with the assistant (and if so, what)?

We also sought quantitative, subjective metrics for the following hypotheses for the structured tasks measured via self-reported completion times,[3] NASA TLX cognitive load scale [17], and a subset of five system usability Likert questions[4] derived from the System Usability Scale [6]:

H1. Users feel that they complete tasks more quickly when using ChainBuddy, versus without.
H2. Users perceive their workload with ChainBuddy as less demanding than the manual baseline interface.
H3. Users report greater self-satisfaction with ChainBuddy, versus without (i.e., ease of use, confidence, learnability).
H4. Users are able to complete tasks more successfully with ChainBuddy, versus without.

4.1 Recruitment and Participants

We recruited 12 in-lab participants around our North America-based university through listservs and Slack channels, mainly in computer science and engineering contexts. Participants were between ages 18-34 (seven between 23-27, three from 28-34 and one 18-22) and balanced across gender (7 female; 6 male). Ten out of 12 were from computer science or engineering backgrounds (the other two were from Neuroscience and Life Sciences, respectively). They reported a relatively high past experience with LLMs (μ=3.83, σ=0.71 on a scale 1-5, with 5 highest) as well as Python programming knowledge (μ=3.83, σ=1.11; only P6, who had a background in Life Sciences, indicated no knowledge). Half of the participants self-reported as having "worked on a university study, paper, or project involving the evaluation of [LLMs]" and three participants had heard of or used ChainForge prior to the study. Each study took 75 minutes, and participants were compensated $30 in cash (CAD).

4.2 Methodology, Procedure and Tasks

We designed a within-subjects study with mixed methods. There were two Conditions: Assistant (with ChainBuddy) and Control (the baseline ChainForge interface without ChainBuddy). We also devised two tasks of roughly the same style and difficulty, and randomly assigned them in a counterbalanced manner to the conditions. The tasks were $T_{\texttt{email}}$: "You are a software engineer tasked with designing an automated tool to help people professionalize their emails for work contexts", and $T_{\texttt{tweet}}$: "You are working on a tool to help summarize long text paragraphs into concise, catchy tweets limited to 144 characters." In both cases, we told users their goal was "to set up a workflow in ChainForge that can help you find the 'best' prompt" and emphasized that their goal "is not to find the best prompt, but rather to set up a flow that can help you find the best prompt and compare between prompts." There were $2 \times 2 = 4$ unique orderings of Condition × Task. With 12 participants, this means that three (3) participants experienced each unique ordering, and six (6) experienced the Control first.

After obtaining informed consent, the study procedure was:

(1) A 5 minute video overview of the ChainForge interface

[3]Specifically, the time from when participants first clicked or typed in the interface, to the time when they said they were "done" or ran out of time. Participants were given 12 minutes for all tasks; if they ran out of time, the time was recorded as the maximum.
[4]The five questions, "I found the system easy to use," "I would like to use this system frequently," "I found the system unnecessarily complex," "I felt confident using the system", and "I needed to learn a lot of things before I could get going with this system".

采用1-5等级进行评分（附录A.1），并提供定性反馈及评分理由。完整的27条提示列于附录A.2中，同时附有两名独立评分者的平均质量评分，以及定性反馈的总结。

在所有类别中，平均评分为4.09，标准差为0.7，表明绝大多数评分集中在3-5分范围内。4分的评分意味着由ChainBuddy生成的平均工作流程与用户需求相符，仅有1-2个易于修复的小问题（附录A.1）。值得注意的是，仅有一个工作流程的平均得分低于3分（提示Q3.3.1）。对平均分进行单向方差分析，跨类别比较的结果不显著（p=0.42）；因此，ChainBuddy在本文所考虑的各类别中表现相近。27个评分中有22个彼此相差不超过1分，评分者之间表现出轻微的一致性（加权科恩卡帕= 0.17）。

总体而言，这表明ChainBuddy已足够强大，能够进入用户研究阶段；同时也指出了需要改进的领域。从定性反馈来看，14个工作流中最普遍的问题涉及所实现和选择的评估器质量：LLM评分器的提示可能包含稍显模糊的标准；而代码评估器则可能“过于严格”且难以泛化（例如，针对LLM输出的不同格式化）。除此之外，有6个问题与可视化节点图相关，特别是未能按某些预期变量（如提示或提示变量）进行绘制。这是由于我们的疏忽造成的：我们的ChainBuddy原型仅添加了一个默认的可视化节点，该节点默认按LLM进行绘制，而更广泛的ChainForge界面则允许用户更改变量。尽管我们也发现了小问题与输入数据及提示措辞相关，这一反馈表明大多数挑战源于ChainBuddy在LLM管道的评估部分所做的选择。需注意，将评估器与用户期望对齐并非易事，近期一项解决方案提出需要采用人在回路的方法来对齐生成内容评估器通过用户对LLM输出[38]的评分。

4 可用性研究

为评估ChainBuddy，我们针对基线界面（不含ChainBuddy的ChainForge）开展了受试者内混合方法用户研究，因其最接近“手动”开放式LLM管道搭建系统的直接对比。我们的目标广泛聚焦于人们如何期望使用AI助手生成LLM行为评估；具体而言，定性评估关注：

(1) 用户最欣赏该助手的哪些方面，与基线相比?
(2) 用户是否认为需求收集交互有帮助或必要?
(3) 参与者希望将Chain-Buddy用于解决哪些类型的问题？（自由探索任务）
(4) Do 人们 feel that 他们的 想法 or 假设 改变 之后 与助手互动时的情况?
(5) 人们如何编辑生成的流程？通常会进行哪些类型的修改 他们倾向于做出什么?
(6) 人们在与助手互动过程中是否有所收获（若有，具体是什么）？

我们还针对以下方面寻求了定量的主观指标
针对通过自评方式测量的结构化任务的假设
完成时间、3项NASA TLX认知负荷量表[17], 以及
从系统可用性量表中提取的5个李克特问题4
系统可用性量表[6]:

假设1. 用户认为在使用ChainBuddy时完成任务的速度比不使用更快。
ChainBuddy时更快。
假设2. 用户认为使用ChainBuddy的工作量比手动基线界面更轻松。

H3. 用户报告称使用ChainBuddy时自我满意度更高，相较于不使用的情况（即易用性、信心、可学习性）。

H4. 用户能够更成功地完成任务
使用ChainBuddy时，相较于不使用的情况。

4.1 招募与参与者

我们通过邮件列表和Slack频道招募了12名实验室参与者，主要来自我们所在的北美大学计算机相关领域。参与者年龄介于18-34岁之间（7人23-27岁，3人28-34岁，1人18-22岁），且性别分布均衡（7名女性；6名男性）。12人中有10人具备计算机科学或工程背景（另两人分别来自神经科学与生命科学领域）。他们报告称对大型语言模型具有较高的既往使用经验（μ=3.83, σ=0.71，基于1-5分制（5分为最高）以及Python编程知识（μ=3.83, σ=1.11；仅参与者6具有生命科学背景表示不具备相关知识）。半数参与者自称曾“参与过涉及{v1}大型语言模型{v2}评估的大学研究、论文或项目”，另有三位参与者在研究前曾听闻或使用过ChainForge。每次研究时长为75分钟，参与者获得了30加元（CAD）的现金补偿。

4.2 方法论、程序与任务

我们设计了一项采用混合方法的受试者内研究。设有两个条件组：助手组（使用ChainBuddy）和对照组（基线ChainForge界面，不含ChainBuddy）。我们还设计了两项风格与难度大致相同的任务，以平衡抵消的方式随机分配至各实验条件。任务包括Temail：“你是一名软件工程师，负责设计一个自动化工具来帮助人们将工作邮件专业化处理”，以及Ttweet：“你正在开发一个工具，用于将长文本段落总结为简洁、吸引人且限制在144个字符内的推文。”两种情境下，我们都告知用户其目标是“在ChainForge中设置一个能帮助你构建“最佳”提示，并强调他们的目标“并非寻找最佳提示，而是建立一个能帮助你找到答案的流程最佳提示并比较提示之间的差异。”共有$2 \times 2 = 4$条件× 任务的独特顺序。在12名参与者的情况下，这意味着每三个（3）参与者经历了每个独特的顺序，且有六（6）位参与者先经历了对照组。
[illegible] 获取 知情 同意, the 研究 程序 was:
(1) 一段5分钟的ChainForge界面视频概述

具体而言，时间从参与者首次在界面中点击或键入开始计算，直至他们表示“已完成”或用尽规定时间为止。参与者被给予12分钟完成所有任务；若超时，则记录为最长时限。
4五个问题分别是：“我发现该系统易于使用”，“我希望频繁地使用该系统”，“我认为该系统不必要的复杂”，“我对使用该系统有信心”，以及“我需要学习很多内容才能开始使用该系统”。

(2) A tutorial that guides users through the process of writing a prompt template that takes an ingredient and returns the dishes it can make, comparing prompts using template chaining, querying models, and inspecting results. The tutorial gave participants everything they needed to succeed, as the solutions to tasks were structurally similar to the tutorial flow. (The Tutorial did not introduce the Assistant; see below.)
(3) The first condition and task, in an order randomly assigned and counter-balanced across participants. After the task, the post-task questionnaire.
(4) The second condition and task, and post-task questionnaire.
(5) Unstructured exploration, where participants examined an idea or hypothesis they had within the Assistant condition. They were not required to use the assistant.
(6) Post-interview and compensation

There is also one mini-tutorial for the Assistant that occurs directly before the Assistant condition, wherever the condition appears. We deliberately avoided showing participants the Assistant during the pre-tasks tutorial, because we wanted to mitigate the potential for bias from participants "guessing the hypothesis" of the study (demand characteristics) when experiencing the Control interface first.

4.3 Data Analysis

Participant interactions were screen and audio recorded. We also captured log data for chats with the Assistant and saved flows (files) for all tasks for later analysis. We analyzed qualitative data through inductive thematic analysis; specifically, the first author iteratively affinity diagrammed all participant remarks during the study (including post-interview) to arrive at clusters (codes). These clusters and codes were then discussed by both authors until we reached consensus.

We analyzed quantitative data using a repeated measures linear mixed effects model in R [32], examining fixed effects of Condition, Task, and Order and all interaction effects between these factors, and controlling for random effect of Participant. P-values for main and interaction effects are calculated using Satterthwaite's method for degrees of freedom and reported from ANOVA tables with `lmerTest` [28]. Post-hoc tests were done via estimated marginal means (`emmeans`) with Bonferroni correction; when including the estimate (β) and t-statistic (t), reported p-values are from `emmeans`. We visually inspected the normality of model residuals (via Q-Q plots) and assumption of homoscedasticity. Our quantitative results are meant to complement our qualitative findings.

5 Findings

5.1 Quantitative results

We summarize major quantitative findings here before delving into specifics. Overall, participants perceived the Assistant as significantly **less mentally demanding** and **less physically demanding** than the Control. Participants were also **more confident** and **more performant** when using the Assistant. Finally, participants **created at least one more type of node in the Assistant condition,** on average, with a portion of the effect explained by the Assistant generating Evaluator nodes (where only one participant created a Python Code Evaluator in the Control condition).

There were also two ordering effects by task for perceived Successfulness and Ease of Use, both when experiencing the Control condition after the Assistant. Though these results are specific, interact with Task (see below) and could be considered random fluctuations or demand characteristics given the small study size; however, considered together, they might suggest a small "missing the assistant" effect—using the Assistant first helped some participants apply their knowledge to the next task, but it also may prime them to perceive the baseline interface as harder to use.

5.1.1 **Time to completion (H1).** We did not find a significant main effect at $p<0.05$. We report a borderline significant interaction effect for Condition given Order ($\beta = -239.5$, $t = -2.22$, $p=0.04$): when people experienced the Assistant after the Control condition, they perceived they were finished with the task faster (μ=357 secs versus μ=610; 95% CI [195, 518] versus [445, 771]).

5.1.2 **Workload Demand (NASA TLX) (H2).** We find main effects of Condition on mental demand and physical demand, with participants finding tasks **less mentally demanding** ($\beta = -0.91$, $t = -2.66$, $p=0.01$) and **less physically demanding** ($\beta = -1.08$, $t = -3.15$, $p=0.01$) when using the Assistant (Figure 4). For perceived successfulness, there is also a significant effect of Task (β=0.66, t=2.53, p=0.02) and two interaction effects of between Condition and Task (p=0.02), and Task and Order (p=0.006). Post-hoc `emmeans` tests suggest two sources for the effect: participants report they are more successful at $T_{\texttt{email}}$ than $T_{\texttt{tweet}}$, but only when $T_{\texttt{email}}$ appeared second, possibly indicating a learning effect (β=1.5, t=4.02, p=0.005); and participants felt they were less successful when solving $T_{\texttt{tweet}}$ in the Control condition (β=1.33, t=3.57, p=0.01). No other questions reach significance at $p<0.05$.

5.1.3 **Perceived System Usability (H3).** We find a significant main effect of Condition on Confidence (β=0.5, t=2.44, p=0.04). Participants felt moderately **more confident** using the Assistant (Fig 4).

For Ease of Use, we found a three-way interaction between Condition, Task, and Order (p=0.003). Visual inspection of data reveals two outliers as the culprit: two participants who received $T_{\texttt{tweet}}$ with the Control condition in the second position, who both rated ease of use 1 and 2 (P2 and P7; only one other participant, P9, experienced this order). All other participants rated the Control condition 4 or above for Ease of Use, with a median of 4.5. Eleven (out of 12) participants gave the Assistant a 5 for ease of use (one 4). One possible explanation for the outliers is that these participants "missed" the AI assistant's reduction to their workload during $T_{\texttt{tweet}}$, which asks participants to summarize "long paragraphs" into tweets—requiring them to create test data. Possibly, knowing that the Assistant could generate long example paragraphs for them, these participants were frustrated by having to manually type in input data. (Indeed, there is evidence for this: P2 used a lone Prompt Node to generate test paragraphs, as a workaround for not having assistant access.)

No other questions reached significance, suggesting that participants liked the interface equally across conditions, and found it of similar difficulty to learn.

（2）一份教程，引导用户完成编写提示模板的流程，该模板接收一种食材并返回可制作的菜肴，通过模板链式调用比较提示、查询模型及检查结果。该教程为参与者提供了成功所需的一切，因为任务解决方案在结构上与教程流程相似。（该教程未引入助手；详见下文。）

(3) 第一个条件和任务，顺序随机分配并在参与者间进行平衡。任务完成后填写任务后问卷。

(4) 第二个条件和任务，以及任务后问卷。
(5) 非结构化探索阶段，参与者可在助手条件下验证其想法或假设。

不强制要求使用该助手。
(6) 后访谈 and 补偿

这里还有一个针对助手的迷你教程
在助手条件之前直接出现，无论条件位于何处
出现时，我们刻意避免向参与者展示助手。
在预任务教程期间，因为我们希望减轻
参与者"猜测假设"可能带来的偏见
（需求特性）当首先体验对照组
界面时。

4.3 数据分析

参与者互动进行了屏幕和音频记录。我们还
捕获了与助手聊天的日志数据并保存了流程
（文件）用于后续分析所有任务。我们通过分析定性数据
采用归纳主题分析法；具体而言，第一作者
对研究中所有参与者言论（包括后访谈）进行迭代式亲和图分析，
最终形成聚类（代码）。随后
两位作者对这些聚类和代码展开讨论，直至
达成共识。
我们使用R语言中的重复测量线性混合效应模型[32], 分析了定量数据，考察了条件、

任务和顺序的固定效应以及这些因素之间的所有交互效应、
并控制了参与者的随机效应。主效应和交互效应的P值采用萨特思韦特方法计算自由度，并通过lmerTest包[28]的方差分析表报告。事后检验通过估计边际均值（emmeans包）进行邦费罗尼校正；当包含估计值（β）和t统计量（t）时，报告的p值来自emmeans包。

我们通过Q-Q图目视检查了模型残差的正态性和同方差性假设。我们的定量结果旨在补充定性研究发现。

5 研究发现

5.1 定量结果

在深入细节之前，我们在此总结主要的量化发现。总体而言，参与者认为助手在心理和生理上的负担显著低于对照组。使用助手时，参与者还表现出更高的信心和更好的表现。最后，在助手条件下，参与者平均至少多创建了一种类型的节点，部分效果可归因于
助手生成评估节点（其中仅有一位参与者
在控制条件下创建了一个Python代码评估器)。
在任务顺序上还发现了两个影响：感知成功性和易用性，均出现在先使用助手后经历控制条件的情况下。尽管这些结果具有特定性，

与任务存在交互作用（见下文），且可能因研究规模较小而被视为随机波动或需求特性；但综合来看，它们可能暗示了一种微妙的"怀念助手"效应——先使用助手帮助部分参与者将知识应用到后续任务中，但也可能使他们更倾向于认为基线界面更难使用。

5.1.1 完成时间（假设1）。我们未在$p<0.05$处发现显著的主效应。但报告了条件与顺序($\beta = -239.5$, $t = -2.22$, $p=0.04$)之间边界显著的交互效应：

当参与者在控制条件后体验助手时，
他们感知自己更快完成了任务（μ=357 秒对比μ=610；95%置信区间 [195, 518] 对比[445, 771]）。

5.1.2 工作量需求（NASA任务负荷指数）（H2）。我们发现条件对心理需求和生理需求存在主效应，参与者认为任务的心理需求较低($\beta = -0.91$,

$t = -2.66$ $p=0.01$)，**且生理需求较低**($\beta = -1.08$,
$t=-3.15$ $p=0.01$)，当使用助手时（图4）。对于感知成功率，任务也存在显著效应（β=0.66,

t=2.53, p=0.02）以及条件与任务之间的两个交互效应(p=0.02)，以及任务与顺序的交互效应（p=0.006）。事后emmeans包测试表明效应有两个来源：参与者报告他们在T电子邮件任务上比T推文任务更成功，但仅当T电子邮件出现在第二位时，可能表明存在学习效应（β=1.5, t=4.02,

p=0.005）；且参与者在控制条件(β=1.33, t=3.57, p=0.01)下解决T tweet时感觉成功率较低。其他问题在$p<0.05$水平上均未达到显著性。

5.1.3 感知系统易用性（H3）。我们发现条件对信心有显著主效应（β=0.5, t=2.44, p=0.04）。
参与者使用AI助手时信心程度中等偏高（图4）。

在易用性方面，我们发现条件、任务和顺序之间存在三重交互作用（p=0.003）。数据可视化显示两个异常值：两名在第二顺位接受控制条件Ttweet的参与者，其易用性评分分别为1和2（P2和P7；仅另一名参与者P9
经历过此顺序）。其余参与者对控制条件的易用性评分均在4分及以上，中位数为4.5分。12名参与者中有11人给AI助手的易用性打了5分（1人打4分）。异常值的可能解释是：这些参与者未能注意到人工智能助手在Ttweet任务（要求总结"长段落"）中对其工作量的减轻作用。
转化为推文——要求他们创建测试数据。或许，知道助手能为他们生成长篇示例段落，

这些参与者因不得不手动输入输入数据而感到沮丧。（确实有证据表明这一点：P2使用了一个单独的提示节点来生成测试段落，作为无法访问助手的变通方法。）

其他问题均未达到显著性差异，表明参与者在不同条件下对界面的喜爱程度相同，且认为学习难度相近。

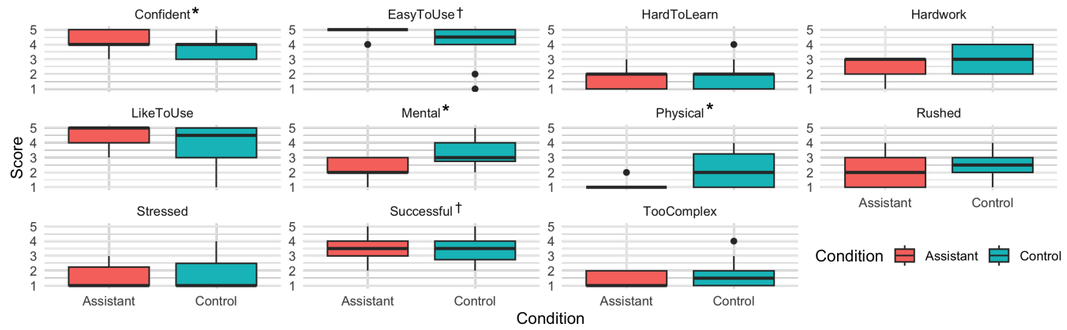

Figure 4: Participant responses to Likert Questions for six NASA TLX measures and five system usability measures, grouped by Condition. Asterisk (*) indicates a significant main effect of Condition at $p < 0.05$; † indicates another main effect, interaction effect or outliers.

5.1.4 **Performance (H4)**. To examine performance, we asked the same expert raters from Section 3.4 to rate the workflows of participants on a scale from 1-5 (Appendix A.1). Experts were blinded to condition, given the same task description as participants, and randomly assigned 8 workflows, with each expert receiving an equal number of workflows from each condition (4 each). For reliability, two separate experts rated each user workflow.

A Mann-Whitney U test shows significant gains in performance for the AI assisted condition compared to the Control at p<0.001 (U=108; see Figure 5). Qualitative examination of the workflows suggests that two-thirds of the workflows in the human condition failed to compare two prompts, compared to all but one workflow in the AI condition; and two-thirds of workflows did not use template chaining, which was an expected part of the user's solution (but not the only valid method). A weighted Cohen's kappa provides an estimate of inter-rater agreement[5] at k=0.388, indicating fair agreement (z=1.96, p=0.0499) among raters.

5.1.5 **Number and Types of Created Nodes**. Finally, we examined the Number of Nodes in participants' final flows, as well as the Number of Node Types (e.g., a flow with two TextFields and one Prompt Node has 3 nodes and 2 types of nodes). Participants created a similar number of nodes regardless of condition or task—suggesting a similar complexity and size of their pipelines. However, for types of nodes, we find main effects of Condition (β=0.583, t=4.04, p=0.003) and Order (β=−0.417, t=−2.887, p=0.02). On average, people created significantly **more node types in the Assistant condition than the Control**; specifically, they created about one more node type (median of 2 node types, versus median of 3 for Control). People also **created more node types during their second task,** indicating a learning effect irrespective of Condition and Task. Post-hoc analyses show significantly more Evaluator Nodes (either Code- or LLM-based) in the Assistant condition (β=0.58, t=3.5, p=0.008), with no other node type reaching

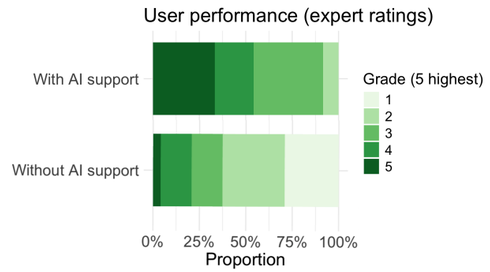

Figure 5: Performance ratings of participant workflows by expert raters, with AI support (Assistant) and without (Control). Expert raters were blind to the condition and used the same scoring scale from our technical evaluation (Appendix A.1).

significance. Inspecting the data further, only one participant added a Python Code Evaluator when in the Control condition.[6] Overall, this suggests **the Assistant helped participants make more Evaluator-type nodes,** especially Python Code Evaluators, and that participants tried out more interface features over time.

5.2 Qualitative results

We examined interactions and interview data across the structured tasks and unstructured exploration. Overall we found several key insights into participants' experiences with ChainBuddy:

- ChainBuddy helps users in overcoming the "blank page problem," converting vague ideas into concrete workflow drafts.
- ChainBuddy's requirement gathering process helped participants elicit user goals and criteria, provided a structured way to present requirements, and expanded user thinking.

[5]Note that because of the particular setup of random assignment to raters, this is only an estimate; we provide it to help readers parse our results. For our MWU test reported here, we included all datapoints; but note that averaging the two rater scores for each workflow and conducting MWU also reaches significance at $p = 0.001$.

[6]One explanation is that writing Python code is difficult and time-consuming, and since we did not screen participants for programming skill, some may have been unable to write said code.

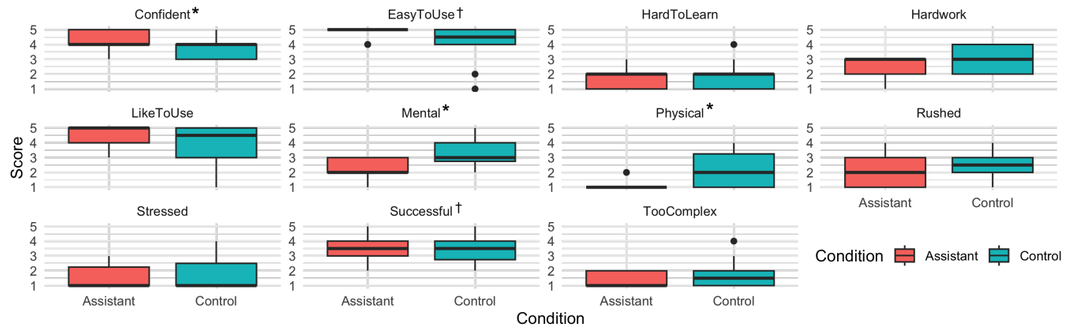

图4：参与者对六项NASA任务负荷指数测量和五项系统可用性测量的李克特问题响应，按条件分组。星号（*）表示在$p < 0.05$时条件的主效应显著；†表示另一主效应、交互效应或异常值。

5.1.4 性能（H4）。为评估性能，我们邀请第3.4节中的同批专家评分者对参与者工作流进行1-5分评级（附录A.1）。专家们对实验条件不知情，接收与参与者相同的任务描述，并随机分配8个工作流，每位专家从每个条件中均获得等量工作流样本（各4个）。为确保可靠性，

两位独立专家对每位用户的工作流程进行了评分。

曼-惠特尼U检验显示性能有显著提升
在p<0.001水平上，AI辅助条件相较于对照组
（U=108；参见图5）。对工作流的定性分析
表明人工条件下三分之二的工作流
未能比较两个提示，而AI条件下除一个工作流外
均完成比较；且三分之二工作流未使用模板
链式操作，这是用户解决方案中的预期部分（但
并非唯一有效方法）。加权科恩卡帕提供了
评分者间一致性5在k=0.388的估计值，表明
评分者之间存在一般一致性（z=1.96, p=0.0499）。

5.1.5 创建节点的数量与类型。最后，我们检查了参与者最终流程中的节点数量以及节点类型数量（例如，一个包含两个文本字段和一个提示节点的流程有3个节点和2种节点类型）。无论条件或任务如何，参与者创建的节点数量相似——表明其管道的复杂性和规模相近。

然而，在节点类型方面，我们发现条件（β=0.583, t=4.04, p=0.003）和顺序（β=−0.417, t=−2.887, p=0.02）存在主效应。

平均而言，参与者在助手条件下创建的节点类型数量显著多于对照组；具体而言，他们多创建了约一种节点类型（对照组中位数为2种，助手条件下中位数为3种）。参与者在执行第二个任务时也创建了更多节点类型，这表明存在学习效应，且不受条件与任务的影响。事后分析显示，助手条件下评估节点（基于代码或大语言模型）数量显著增多（β=0.58, t=3.5, p=0.008），其他节点类型均未达到

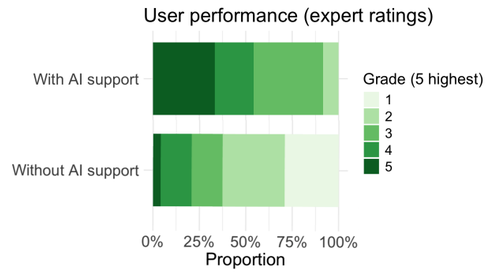

图5：专家评分者对参与者工作流程的性能评分，含AI支持（助手）与不含支持（对照组）。专家评分者对实验条件不知情，并采用技术评估（附录A.1）中的相同评分量表。

显著性水平。进一步检查数据发现，仅有一位参与者添加了
在控制条件下使用Python代码评估器。6总体而言，
这表明助手帮助参与者实现了更多
评估器类型节点，尤其是Python代码评估器，以及
参与者随时间尝试了更多界面功能。

5.2 定性结果

我们分析了结构化任务与非结构化探索中的交互和访谈数据。整体而言，我们发现了几个关键
对参与者使用ChainBuddy体验的见解：

- ChainBuddy帮助用户克服‘空白页问题’，将模糊的想法转化为具体的工作流草案。
- ChainBuddy的需求收集过程帮助参与者明确用户目标和标准，提供了结构化方式来呈现需求，并拓展了用户思维。

5由于随机分配给评分者的特殊设置，这仅是一个估计值；我们提供此信息以帮助读者理解我们的结果。

一个估计值；我们提供它是为了帮助读者解析我们的结果。对于我们所报告的曼-惠特尼U检验，
在此，我们纳入了所有数据点；但需注意，对每位评分者的两个评分者分数取平均值时
工作流程和进行MWU也在$p = 0.001$处达到显著性。

6一种解释是编写Python代码既困难又耗时，且
由于我们未对参与者的编程技能进行筛选，部分人可能
无法编写上述代码。

- Participants perceived that the assistant reduced effort, sped up their workflow and alleviated the learning curve of using the baseline interface.
- ChainBuddy surpassed participants' expectations, with its capabilities surprising users.
- Participants with substantial prior experience in prompt engineering perceived that ChainBuddy aided their prompt engineering process.
- Concerns about potential bias or influence on their problem-solving approach after interacting with the assistant.

We first expand on these findings in the sections below and then discuss the interaction patterns identified by our user study.

5.2.1 **Assistant as a starting point.** Eight participants found that ChainBuddy provided a good starting point for their tasks, helping them overcome the initial challenge of starting from scratch. P5 said, *"maybe it's hard to start with an empty page. It's better to start with a little bit of question and answer here, so you have a base structure."* The assistant helped users quickly generate initial workflow drafts, reducing the effort required to start from scratch. P1 appreciated that *"the assistant can give me, like a first draft. That is pretty good."* Similarly, P9 highlighted the importance of having an example to *"put things into context much more easily,"* which helped them decide whether to proceed with the assistant's suggestions or refine their own approach.

5.2.2 **Requirement gathering process helped refine ideas and elicit requirements.** Six participants reflected that the assistant's structured and iterative approach to intent disambiguation made the process more organized and easier to follow compared to traditional prompting. Moreover, the iterative questioning made users think more deeply about their tasks and specify details they might otherwise overlook, which they felt led to more detailed pipelines.

For instance, P12 remarked that *"sometimes... I know exactly what I want, but I'm not necessarily putting it all in the prompt."* P9 found that *"the follow-up questions made you create a more comprehensive workflow"* and P8 was *"really surprised when it asked questions based on what I need."* The form-filling interface helped users to provide information in a more manageable and less overwhelming way than writing everything at once (e.g. P9: *"I was pleasantly surprised that you can type the answers under each question, instead of having just to blurt it all out in a long piece of text"*). By asking users relevant questions, the assistant could also help broaden their thinking and refine their ideas. P1 found that the assistant *"asked me other questions, and I think of like, 'Oh, it's true, I can do this.'"* P7 said *"it was impressive, because it's started to ask me about the design itself, it was clear and precise,"* and commented that the assistant encouraged him to *"think more on the abstract side of the idea itself."*

However, participants could also worry that Q&A intent elicitation could become excessive, with four suggesting rounds limits or guidance on when sufficient information has been provided. P9 worried about getting *"too caught up in the details, which is me answering the supplemental questions too much. Like I keep on digging... and you end up creating a model that's just too rigid."* P12 also wondered, *"if you keep going and questions keep being asked, that's where you start contradicting yourself."* One suggestion to improve the design was to have the assistant deduce when there is enough information to proceed with generation (e.g., P7 said *"I do think, if it understand the question, it should know where to stop."*, and P2 wanted *"an indicator of... [the] level of understanding that the system has achieved at that point of time"*).

5.2.3 **Exceeding user expectations.** Ten participants remarked the assistant not just met, but exceeded their initial expectations. Participants consistently expressed surprise by the assistant's capabilities, particularly in its abilities to ask relevant follow-up requirement-gathering questions and generate detailed workflows. For instance, P2 remarked, *"I was not expecting it to generate even remotely close workflow. But it gave me two additional nodes, understood that these are my requirements, so I might use it."* P1 was surprised at how well the assistant performed: *"It's asking really good like, follow up questions that made me rethink the problem and generated a good flow for it."* Similarly, P8 initially expected a basic chatbot but was impressed when the assistant *"asked questions based on what I need,"* which helped expand their ideas and improve their prompts. P6 was particularly impressed by how ChainBuddy was able to generate a workflow for a complex goal, stating, *"I asked it to compare detailed incomes for 10 medical specialties, and it did really well, and it also handled a more complex task without misunderstanding."*

When asked to clarify, participants explained that they expected the assistant to be a basic Q&A chatbot, but found it more interactive and insightful. For example, P10's *"expectation was something like ChatGPT that actually gives you the final output, not asking too many questions, back and forth,"* but they appreciated how the assistant *"tries to understand what's your goals, what's your criteria. So I love it better than ChatGPT."*

5.2.4 **Reducing effort.** Participants felt the assistant reduced their workload and streamlined task execution, complementing our quantitative results. P10 said *"the mental demand was much less when I used the assistant,"* which was particularly beneficial because the assistant helped *"translating my inquiries to this platform,"* making the experience accessible for beginners. Similarly, P8 observed that without the assistant, it was *"pretty hard to think on my own, like making flows and like arranging it properly,"* whereas the assistant streamlined the process. Without the assistant, participants could find manually setting up flows confusing or inefficient. P6 reflected that work without the assistant was *"a lot harder than I expected. It seems that the task is simple at first, but then when I do it manually, some of the things does not fall into the category that I expected."* P4 found the assistant interface to be *"very intuitive, it generated and you just drag and drop everything [after]. So I did not find any frustrations."*

5.2.5 **Accelerating workflow.** Participants felt that the assistant significantly accelerated their workflow. P11 remarked that *"with the assistant, it's really quick. It gives you the ideas, and it makes all the necessary connections on the platform, so you save up that manual time of creating the whole workflow. And on top of that, it also saves mental efforts to start with the prompt."* Similarly, P9 felt that while there wasn't a significant difference in their success rate and solution, *"it was just faster with ChainBuddy."* Figure 6 shows interaction traces for user actions across both conditions, plotted by time. Most notable is how much effort users in the Control

- 参与者认为该助手减少了工作量，加快了优化了他们的工作流程并减轻了使用基线界面的学习曲线。
- ChainBuddy超出了参与者的预期，其功能令用户感到惊喜。
- 具有丰富提示工程经验的参与者认为ChainBuddy对他们的提示工程流程有所帮助。
- 对与助手交互后可能产生的偏见或影响其问题解决方法的担忧。

我们首先在以下章节中详细阐述这些发现，然后讨论用户研究识别出的交互模式。

5.2.1 助手作为起点。八位参与者发现 ChainBuddy为他们的任务提供了良好的起点，能够协助帮助他们克服从零开始的初始挑战。P5表示，

“也许从一张空白页面开始很难。最好是先有一些问答环节，这样你就有了一个基础结构。”

该助手帮助用户快速生成初步的工作流草案，减少了从零开始所需的努力。P1很欣赏

“助手能给我一个初稿。这相当不错。”

同样地，P9强调了拥有一个示例的重要性，以便"更容易地将事物置于上下文中"，这帮助他们决定是继续采用助手的建议还是完善他们自己的方法。

5.2.2 需求收集过程有助于完善想法并引出需求。六位参与者反映，助手结构化且迭代式的意图消歧方法使该过程比传统提示更有条理且更易于遵循。此外，迭代提问促使用户更深入地思考他们的任务，并明确他们可能忽略的细节，他们认为这导致了更详细的流程。

例如，*P12*评论道："有时...我确切地知道自己想要什么，但我不一定把所有内容都放入提示中。" *P9*发现"后续问题让你创建了一个更全面的工作流程"，而*P8*则"真的惊讶于它会根据我的需求提问"。表单填写界面帮助用户以一种更易管理且不那么压倒性的方式提供信息，而不是一次性写出所有内容（例如*P9*："我惊喜地发现你可以在每个问题下键入答案，而不是必须在一大段文字中全部说出来"）。通过向用户提出相关问题，助手还能帮助拓宽他们的思维并完善想法。*P1*发现助手"问了我其他问题，然后我想，'哦，确实，我可以这样做。'" *P7*表示"这令人印象深刻，因为它开始询问我关于设计本身的问题，既清晰又精确"，并评论说助手鼓励他"更多地思考想法本身的抽象层面"。

然而，参与者也可能担心问答式意图 elici- tation 会变得过度，其中有四人建议设置轮次限制或提供关于何时信息已足够的指引。P9 担心会"过于纠结细节，也就是我回答补充问题太多。就像我不断深挖……最终会创建一个过于僵化的模型。" P12 也疑惑，"如果你继续下去，问题不断被提出，这时你就会开始自相矛盾。" 改进设计的一个建议是让助手推断何时有足够的信息来继续生成（例如，P7 说"我确实认为，如果它理解了问题，就应该知道在哪里停止。", 而*P2*希望"一个能体现 *[系统]* 在该时间点已达到理解水平的指标在那一刻已经实现了。

5.2.3 超越用户期望。十位参与者表示，该助手不仅满足还超出了他们的初始预期。

参与者们对助手的能力——尤其是提出相关后续需求收集问题和生成详细工作流程的能力——持续表现出惊讶。

例如，*P2*评论道："我原本不指望它能生成哪怕勉强接近的工作流程。但它给了我两个额外节点，理解这些是我的需求，所以我可能会使用它。" *P1*对助手表现之佳感到惊讶："它提出的后续问题非常好，让我重新思考问题并生成了一个优秀的流程。" 类似地，*P8*最初期待一个基础聊天机器人，但当助手"根据我的需求提问"时深受触动，这帮助他们拓展了想法并改进了提示。参与者*6*尤其对*ChainBuddy*能为复杂目标生成工作流程印象深刻，表示："我要求它比较*10*个医学专业的详细收入，它做得非常好，还处理了更复杂的任务且没有误解。"

当被要求澄清时，参与者解释说，他们原本期望助手是一个基础问答聊天机器人，但发现它更具互动性和洞察力。例如，*P10*表示“原本期待的是类似*ChatGPT*那样直接给出最终输出，不会问太多来回问题”，但他们赞赏助手“试图理解你的目标和标准。所以我更喜欢它胜过*ChatGPT*。”

5.2.4 减轻工作量。参与者认为助手减少了他们的工作量并简化了任务执行，这与我们的定量结果相呼应。P10表示“使用助手时心理需求大幅降低”，这一点尤其有益，因为助手帮助“将我的查询翻译到这个平台”，使得初学者也能轻松上手。同样，P8观察到没有助手时，“靠自己思考相当困难”，

比如创建流程和合理排列”，而助手则简化了这一过程。没有助手的情况下，参与者反思道“工作比预期困难得多。起初任务看起来简单，但手动操作时，有些内容并不符合我预期的分类。”*P4*认为助手界面“非常直观，它生成内容后你只需拖放所有内容*[之后]*。所以我没有任何挫败感。”

认为手动设置流程令人困惑或低效。参与者6

5.2.5 加速工作流程。参与者普遍认为该助手显著加快了他们的工作流程。参与者11表示：“有了这个助手，速度真的很快。它会提供想法，并在平台上完成所有必要的连接，这样你就节省了手动创建整个工作流程的时间。更重要的是，它还省去了从零开始构思提示的脑力消耗。”类似地，P9认为虽然在使用ChainBuddy时成功率与解决方案质量没有显著差异，“但速度就是更快了。”图6展示了两种条件下用户操作的交互轨迹随时间变化的绘制结果。最值得注意的是对照组用户需要付出多少额外努力

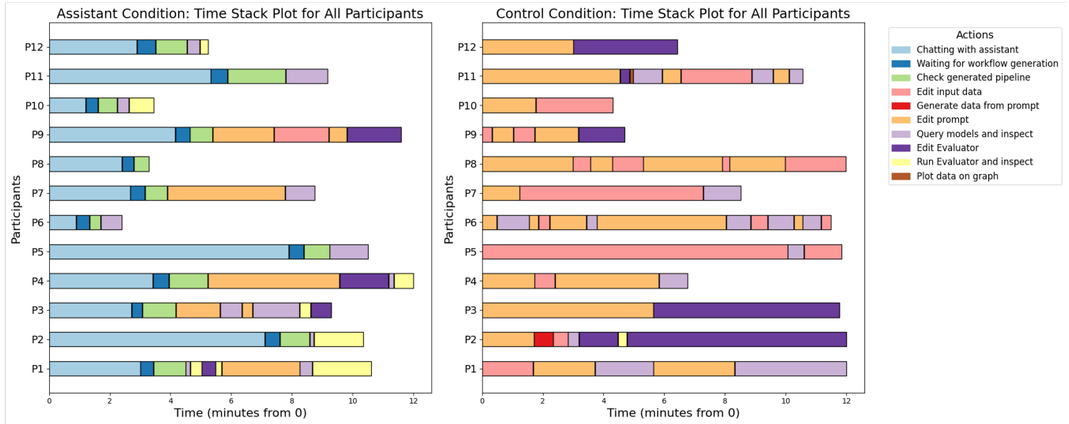

Figure 6: Interaction traces of participant actions when completing tasks, comparing conditions. Notice the relative dearth of time spent editing input data in the Assistant condition, compared to the Control.

condition spent on preparing and editing input data (TextFields nodes), compared to the Assistant, where only one participant (P9) edited the input data generated by ChainBuddy. Notable too is that half the participants ran evaluators and inspected results when using the Assistant, compared to only one in the Control. Yet, we also see the time users spent chatting with the assistant could sometimes take significant time—in four instances here, it neared or exceeded five minutes.

5.2.6 **Helping users learn the platform.** Some participants felt that the assistant helped them learn and adapt to the platform, reducing the learning curve and making the tool more accessible for first-time users. P7 highlighted that *"usually when you don't have an assistant and have nothing to work with you need, similar to learning new coding languages,"* but with the assistant, the process was *"very fast."* P3 noted, *"for the first exercise, I really didn't have much ideas of what I should do, after using the assistance, it's just much more clear."* P5 also found the assistant helpful in guiding them through tasks, saying that *"it gives a sample that shows how these [nodes] could be connected. I just saw that, oh, I can name these [nodes] with relevant things [titles of the nodes], it's good to name relevant [nodes] especially for someone using the tool for the first time, it's very helpful."* P3, who had limited experience with AI tools, was impressed by the assistant's capability, saying, *"The assistants just enlightened me,"* and found it surprisingly effective compared to their prior experiences with AI chatbot from other interfaces.

The assistant was especially beneficial for helping participants overcome initial obstacles to understanding the interface. P6 shared that without the assistant, *"I don't really know what to do sometimes,"* but with the assistant, they *"did not need to think [about] anything."* P7 also said that*"without the assistant, there would be obstacles for first-time users, a lot of people will struggle,"* like the experience to the challenges of using something unfamiliar. P11 felt that the assistant *"changed [my approach] to be better"*.

5.2.7 **Streamlining prompt engineering.** In our study, there were three participants who often did prompt engineering in their daily work. P7 explained that in their prompt engineering tasks, they often need to *"write a program for each LLM and collect responses,"* which is time-consuming. The assistant's ability to handle these tasks with built-in tools meant that *"it will do all of that for you, because it already have the model, it has its evaluation tool, and you don't need to do much time in order to experiment [with] multiple things."* P4 reflected, *"I think the most difficult part of doing this prompt engineering is writing the initial set of prompts. And I think that is where the assistant is very helpful, that it gives you an initial set of prompts, then that you can start working on... It's a very, very good product, I would use it a lot."* P12, who often needed to duel with variations in data formats in their real work, found the assistant helpful in creating a workflow that could compare prompts to standardize differently structured information.

5.2.8 **Interaction patterns with the assistant.** We identified three patterns in the process of participants' usage of the assistant, derived from qualitative analysis of the screen-capture recordings of the structured task and free exploration time. Many participants (8) **made only minimal edits to assistant's solution,** tweaking input values but keeping its overall structure. This approach was observed mainly during the structured task, with seven participants reliant during the task, versus only three in free exploration. Second, we saw participants **heavily revising the assistant's solution without changing its structure**, entirely changing prompts in addition to input data. This approach allowed participants to tailor the assistant's suggestions to better fit their needs while still trusting the structure of the assistant's initial solution. Third, participants **amended or extended the structure of the assistant's solution**, resulting in more complex and customized chains (e.g., P1, P2, P9, and P10). This last approach demonstrates a collaborative use of the assistant, where its suggestions are seen as a foundation upon

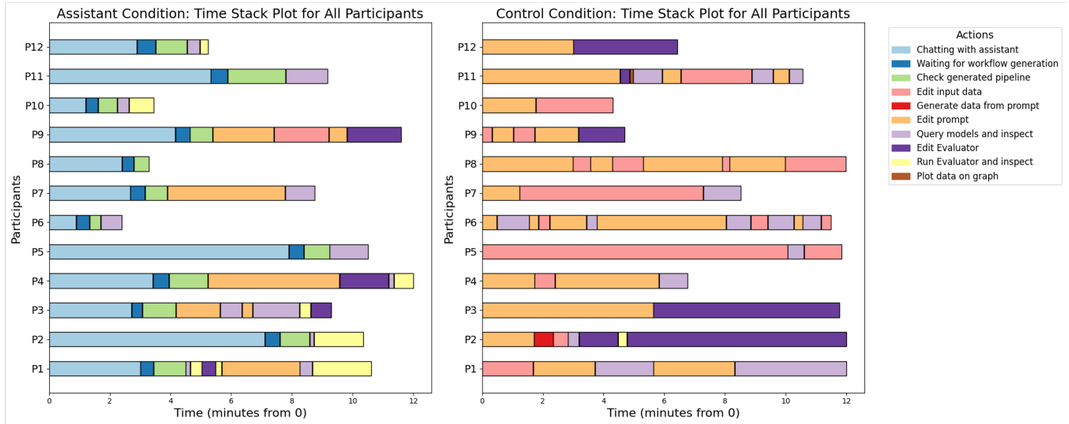

图6：参与者完成任务时的交互轨迹对比（不同条件）。注意相较于对照组，助手条件下编辑输入数据的时间明显偏少

条件花费在准备和编辑输入数据上（文本字段节点），而助手条件下仅一名参与者（P9）修改了由ChainBuddy生成的输入数据。同样值得注意的是半数的参与者在运行评估器并检查结果时使用了助手，而对照组中仅有一人如此。然而，我们也发现用户与助手聊天所花费的时间有时会相当长——在此处的四个案例中，时间接近甚至超过了五分钟。

5.2.6 帮助用户学习平台。部分参与者认为助手帮助他们学习并适应了平台，

降低了学习曲线，使该工具对首次使用者更易上手。*P7*强调说："通常当你没有助手且缺乏所需资源时，就像学习新编程语言一样困难"，但有了助手后，这个过程变得"非常快速"。*P3*指出："在第一个练习中，我完全不知道该做什么，使用助手后思路就清晰多了。"*P5*也认为助手在引导完成任务时很有帮助，表示"它提供了一个展示这些*[节点]*如何连接的示例。我看到后恍然大悟——原来可以给这些*[节点]*赋予相关*[节点标题]*，，尤其对初次使用工具的人来说，给*[节点]*命名相关名称非常实用。"

这非常有帮助。"P3对人工智能工具经验有限，但对助手的能力印象深刻："助手简直让我茅塞顿开"，并发现与其他界面的人工智能聊天机器人相比，其效果出奇地好。

该助手在帮助参与者克服理解界面的初始障碍方面尤为有效。参与者6表示，若没有助手，"有时我真的不知道该做什么"，但有了助手后，他们"无需思考*[关于]*任何事情"。参与者7也提到"没有助手的话，首次使用者会遇到障碍，很多人会感到困惑"，就像面对使用陌生事物时的挑战体验。参与者11认为助手"改变*[了我的方法]*使其更优"。

5.2.7 优化提示工程。在我们的研究中，有三名参与者经常在日常工作中进行提示工程。参与者7解释道，在他们的提示工程任务中，

他们经常需要"为每个大型语言模型编写程序并收集响应"，这非常耗时。助手内置工具处理这些任务的能力意味着"它会为你完成所有这些工作，因为它已经拥有模型和评估工具，你不需要花费太多时间就能实验*[多种]*可能性。"参与者*4*反思道："我认为提示工程最困难的部分是编写初始提示集。而助手在这方面非常有用，它能提供初始提示集让你开始工作...这是个非常非常棒的产品，我会经常使用它。"经常需要处理数据格式变化的*P12*发现，助手在创建工作流程以比较提示来标准化不同结构信息方面很有帮助。

5.2.8 与助手的交互模式。我们识别出参与者使用助手过程中的三种模式，

源自对结构化任务和自由探索时间的屏幕录制进行的定性分析。许多参与者（8人）仅对助手解决方案进行了最小程度的编辑，调整输入值但保留其整体结构。这种方法主要在结构化任务期间观察到，有七名参与者在任务期间依赖此方法，而在自由探索中仅有三人。其次，

我们看到参与者在未改变结构的情况下大幅修改助手解决方案，完全更改提示以及输入数据。这种方法使参与者能够调整助手的建议以更好地满足其需求，同时仍信任助手初始解决方案的结构。第三，参与者修改或扩展了助手解决方案的结构，

从而形成更复杂和定制化的链（例如P1、P2、P9，以及P10）。这最后一种方法展示了助手的协作使用，其建议被视为一个基础，

which users can add their own creativity and complexity. Note that during free exploration, participants P3 and P12 did not use the assistant directly, but were inspired by its solution to the second task (e.g., *"because the structure that is in my mind is kind of similar to the last one done by the assistant"* and *"I already had something in my mind, I started building a chain from an existing one"*).

5.2.9 **Concerns about assistant bias and influence.** Some participants expressed concerns that using the assistant influences their problem-solving approach, potentially biasing their subsequent tasks. P9 reflected on this impact: *"The first task I did was with the ChainBuddy. The second one was without, [but] I think because I started off with [it], it kind of taught me how to do it."* Two participants noticed that their approach to their ideas in the free exploration time mirrored the structure introduced by the assistant. P3 said, *"I didn't use the assistant because the structure that is in my mind is kind of similar to the last one done by the assistant,"* indicating that the assistant's influence persisted even when not directly used. P12 also described how she *"started building a chain from an existing one, and that one is created from the second task, which is done by the assistant."* This continuity suggests that initial interactions with the assistant could set a pattern that users continue to follow, even when trying to work independently. We reflect on this risk further in Discussion.

5.2.10 **Use cases in free exploration time.** During the free exploration time, participants explored a diverse range of ideas, showcasing creative and varied uses in collaborating with the assistant to address specific needs and interests. The diverse range of topics reflected a similar diverse range in the study of ChainForge [1], the baseline interface. Participant usage fell into three major categories: comparing and exploring different prompts, evaluating behavior across different LLMs, and testing LLMs for bias and handling of sensitive topics. The full list of participant ideas is listed in Appendix C.

5.3 Limitations

Our within-subjects usability study had a small sample size and compared to a single baseline. It could be that for a different LLM pipeline interface, we might have seen different results. We did not restrict users by prior knowledge of the baseline interface—it could be that expert ChainForge users would feel differently about the assistant's capabilities. For various constraints common to usability studies, we also only examined two tasks that both involved comparing across prompts, even though ChainBuddy is capable of more types of tasks and LLM pipelines, such as comparing responses across models or setting up data processing workflows. An alternate study design might explicitly try three or more very different tasks with a larger sample size, to better understand when ChainBuddy is and is not useful. Finally, our quantitative findings should also be taken with a grain of salt: the complex interaction effects we observed for Ease of Use and Successfulness in particular may be due to random noise. Nevertheless, we believe that the body of evidence is clear, in that all tests of significance leaned in favor of ChainBuddy, and qualitative feedback was overwhelmingly positive. We intend to run a follow-up study post-deployment, investigating a broader range of tasks and real-world use cases.

6 Discussion

Our findings provide evidence in favor of AI-assisted interfaces for generating LLM pipelines. Participants found their workload less demanding when using an AI assistant's help in an interface for setting up evaluative LLM pipelines, compared to without. And, they were more confident, performant, and able to create automated evaluations with greater regularity. In post-interviews, participants appreciated how the requirements gathering interaction supported refining their ideas, and consistently expressed that they were impressed by the assistant's capabilities. We also observed interaction effects that suggest participants can experience lower perceived successfulness and interface ease-of-use when AI assistance is taken away from them (mediated by task); a generous interpretation is that people miss the assistant's help in reducing their workload.

One interesting null result is time. We did not find significant *quantitative* evidence that ChainBuddy helped people solve the task faster, as might be expected (although participants did express this feeling in post-interviews). However, four things to note. First, ChainBuddy takes time to generate flows, on the order of 10 seconds or greater. Second, our times are subjective, with a participant-reported indication of being "done." Third, post-interview data indicates that some participants felt they could go further in their analysis with the assistant, compared to without—for instance, whereas 6 participants ran evaluators and spent time inspecting evaluation results with the assistant, only a single participant without the assistant ran evaluators. Finally, though participants spent around the same time per condition, *what* they spent that time on differed. Participants using the assistant spent time clarifying their intent with the assistant, revising, running or inspecting the generated flows. Interaction traces (Fig. 6) of user actions indicated that considerably less time was spent on preparing and editing input data for the pipeline, in particular.

More broadly than LLM pipelines, our work contributes to a growing body of literature on AI agents integrated into software platforms to assist users in implementing ideas (e.g., an agent that creates Powerpoint slides, visualizations from datasets, or dynamic widgets [4, 9, 44, 52, 53]). Our study provides evidence for the intuitive hypothesis that users perceive that AI agents improve workload by reducing the mental and physical burdens placed on them when interacting with software—clicking and dragging, planning what to do. However, our quantitative findings indicate that for the most part, users perceived they were just as successful *without* the assistant—even when performance analysis suggests performance plummeted without the assistants' help. This finding resembles the Dunning–Kruger effect [26]—that people who lack the skill to evaluate the quality of their work in a domain may overestimate its quality—and, combined with worries about being biased by the assistant (discussed below), contributes to growing concerns about non-experts' over-reliance on AI systems [2, 7]. Note that prior studies on LLM-assisted workflow generation only report user self-perceptions of performance (e.g., [9, 53]); our findings suggest future studies on workflow generation assistants need to complement subjective measures with more objective evaluations of performance (e.g., by expert raters).

用户可以在其上添加自己的创造力和复杂性。请注意
在自由探索期间，参与者P3和P12并未直接使用
该助手，但受到了其对第二个

任务解决方案的启发（例如："因为我脑海中的结构与
助手上次完成的那个有些相似"以及"我已经有了类似
在我脑海中，我开始从现有的链条上构建新的链条。"

5.2.9 对助手偏见与影响的担忧。部分参与者担心使用助手会影响他们的问题解决方法，可能对后续任务产生偏见。P9反思了这种影响："我完成的第一个任务是与ChainBuddy合作的。第二个没有使用助手，*[但]*我认为因为一开始用了*[它]*，，它某种程度上教会了我如何操作。"两位参与者注意到，他们在自由探索时间内的想法处理方式与助手引入的结构相似。

参与者*P3*表示："我没有使用助手，因为我脑海中的结构与上次助手完成的结构有些相似。"这表明即使未直接使用助手，其影响仍然持续存在。

P12也描述了她如何"从现有链条开始构建新链条，而那条链条源自助手完成的第二个任务。"这种连续性表明，与助手的初始互动可能设定了一种模式，即使用户尝试独立工作时仍会遵循。我们将在讨论中进一步反思这一风险。

5.2.10 自由探索时间中的用例。在自由探索时间内，参与者探索了多样化的想法，展示了与助手协作解决具体需求和兴趣时富有创意且多变的用途。主题的多样性反映了ChainForge [1]，研究中类似的广泛性。

基线界面。参与者的使用情况可分为三大类：比较和探索不同提示、评估不同大型语言模型的行为，以及测试大型语言模型对偏见和敏感话题的处理。完整的参与者想法列表见附录C。

5.3 局限性

我们的受试者内可用性研究样本量较小，且仅与单一基线进行比较。对于不同的LLM流程界面，我们可能会得到不同的结果。我们没有限制用户对基线界面的先验知识——ChainForge专家用户可能对助手能力的感受会有所不同。由于可用性研究中常见的各种约束条件，我们仅考察了两个涉及跨提示比较的任务，尽管ChainBuddy能够处理更多类型的任务和LLM管道，例如跨模型比较响应或设置数据处理工作流。

另一种研究设计可能会明确尝试三个或更多截然不同的任务，并采用更大的样本量，以更好地理解ChainBuddy在何时有用、何时无用。最后，我们的定量研究结果也应持保留态度：我们观察到的易用性和成功率之间复杂的交互效应，尤其可能是随机噪声所致。尽管如此，我们相信证据主体是清晰的，因为所有显著性检验都倾向于支持ChainBuddy，且定性反馈压倒性地积极。我们计划在部署后进行一项后续研究，

调查更广泛的任务和实际使用案例。

6 讨论

我们的研究结果为AI辅助界面在生成LLM管道方面的优势提供了证据。参与者发现，在使用AI助手帮助设置评估性LLM流水线的界面时，他们的工作量比不使用时要轻松。而且，

他们更有信心、表现更佳，并能更规律地创建自动化评估。在事后访谈中，参与者赞赏需求收集交互如何支持他们完善想法，并一致表达了对助手能力的印象深刻。我们还观察到交互效应表明，当AI辅助被撤走时（受任务调节），参与者可能会经历较低的感知成功率和界面易用性；一个宽泛的解释是，人们怀念助手在减轻工作量方面提供的帮助。

一个有趣的零结果是时间。我们并未发现ChainBuddy能显著帮助人们更快完成任务的数量证据，尽管参与者在后访谈中表达了这种感受。但有四点需要注意。首先，

ChainBuddy生成流程需要时间，通常在10秒或更久。其次，我们的时间数据是主观的，基于参与者自我报告的"完成"状态。第三，事后访谈数据表明，部分参与者认为在助手协助下能比单独工作时进行更深入的分析——例如，

有6名参与者在助手协助下运行评估器并花时间检查评估结果，而仅有1名参与者在无助手时运行了评估器。最后，虽然参与者在每种条件下花费的时间相近，但其时间分配方式不同。使用助手的参与者会将时间用于向助手澄清意图、修改、运行或检查生成的流程。用户行为的交互轨迹（图6）特别显示，用于准备和编辑管道输入数据的时间明显减少。

更广泛地说，我们的工作不仅限于大型语言模型管道，还对整个领域有所贡献。
越来越多关于人工智能代理集成到软件中的文献
平台协助用户实现想法（例如，一个代理可以
创建Powerpoint幻灯片、从数据集生成可视化或动态内容
控件 [4, 9, 44, 52, 53]）。我们的研究为
用户直观地假设认为人工智能代理有所改进
通过减轻用户在交互过程中承受的精神和身体负担来减少工作量
如点击、拖拽等操作时
以及规划下一步行动时的负担。然而我们的量化数据显示
大多数用户认为即使没有助手也能取得同样的成功
尽管性能分析表明
在没有助手协助的情况下性能会急剧下降。这一发现
类似于邓宁-克鲁格效应[26]——即缺乏
评估其工作质量所需技能的领域从业者可能高估其质量——再加上对存在偏见的担忧

（由助手引发，下文将讨论），这些因素共同加剧了人们
对非专业人士对AI系统过度依赖[2, 7]的日益关注。值得注意的是，
先前关于LLM辅助的工作流生成研究仅报告了
用户对性能的自我感知（例如[9, 53]）；而我们的发现
建议未来研究需要针对工作流生成助手
以更客观的评估补充主观测量
性能（例如由专家评分者评估）。

6.1 Risk of over-reliance on AI assistance for LLM pipeline creation

The reader might ask whether and to what extent the user over-relies on, or is biased by, the assistant's solutions. Even if the assistant helps the user learn the platform (which some participants indicated), is there a risk of it proposing solutions that are subpar or misleading in some way, outside of more straightforward usability errors? For instance, imagine a user asks ChainBuddy to set up a flow to evaluate LLMs for gender bias. In this task, ChainBuddy usually sets up a flow that compares whether the presence of a social identity marker (male, female, non-binary) changes the LLM success rate on solving a math problem—a common "persona" approach used in benchmarking papers [8]. While one way to evaluate gender bias, it certainly does not exhaust all possibilities, nor does the current assistant help the user "scale up" their analysis, choose one solution from alternatives and weigh trade-offs (a learning technique from variation theory [30]), or help them hedge against misinterpretation of random noise. In our study, we did see one error case where ChainBuddy constructed a wrong solution that did not compare two prompts. The participant was evidently not able to identify and fix the AI's mistakes. This raises the question of how future assistants can help the user learn the interface, while mitigating the potential for bias.

6.2 What is the right interaction design for gathering user requirements?

AI-assisted workflow generation is a budding area of research that requires eliciting user requirements. However, it remains unclear what the relative trade-offs are for different interface designs that elicit user intent. Past approaches allow only for a single prompt [27, 53], adopt an "infinite questioner" that ask one question at a time until the user presses generate [9], or stop generation in stages to ask for user feedback before continuing [22]. Here, we explored a structured form-filling approach that mixed chat with a three-question form, where users could choose which questions to answer; in part, this was because our early pilot studies suggested that an unstructured infinite questioner was time-consuming and pressuring to users. Users in our study appreciated our approach, however it still remains unclear what the "right" approach is to eliciting user requirements. In particular, we felt that while functional, the form-filling approach feels awkward inside a chat window. Future work could investigate the advantages and drawbacks of different interfaces for eliciting user requirements for generation tasks with high complexity, and perhaps compare findings with interactive "steering"-based approaches such as Kazemitabaar et al. [22].

6.3 Future Work

Our work presents only the starting point for the emerging research area of AutoLLMOps: AI assistance for designing and implementing LLM pipelines. We summarize potential directions for future work (beyond the obvious one of connecting the chatbot to interface documentation).

Facilitate importing external data within the chat. The process of intent elicitation may involve the user referencing data, such as spreadsheets, internal documents or online resources. A future system of ChainBuddy could allow users to import their own data or other contextual information directly into the system, say using a drag-and-drop interface. By supporting data imports, users can leverage existing datasets or relevant context that the agent can, for instance, transform and add to a Tabular Data Node.

Support Editing of Existing Chains. Another important enhancement is enabling users to edit existing chains. Currently, ChainBuddy supports the creation of new chains but lacks the functionality to modify them once created. A few participants in our study expressed a desire for editing.

Provide User Control Over AI Decision-Making. We may consider providing users with more fine-grained control over the decision-making process of the AI agents. This involves developing features that allow users to influence and guide the agents' decisions more directly, ensuring that the outcomes align closely with their expectations and requirements. For instance, if the user's intent remains vague, a downstream agent handling input data generation may wish to ask the user a question to clarify the ambiguity. However, given that some participants felt the agent should know when to "stop" eliciting their intent, this may trade-off some reduction of user effort for more accurate generation.

Offer a Variety of Suggested Chains. Finally, we plan to increase the variety of suggested chains and nodes provided by ChainBuddy. Here, we limited ChainBuddy to only a few of the most common nodes in ChainForge, missing nodes like Join and Split. By offering a broader range of pre-configured chains and templates, users will have more options to choose from, catering to diverse use cases and preferences. This variety will help users find more relevant and effective starting points for their specific needs, further reducing the "blank page problem."

References

[1] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 304, 18 pages. https://doi.org/10.1145/3613904.3642016

[2] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 81, 16 pages. https://doi.org/10.1145/3411764.3445717

[3] Rafael Barbudo, Sebastián Ventura, and José Raúl Romero. 2023. Eight years of AutoML: categorisation, review and trends. *Knowledge and Information Systems* 65, 12 (2023), 5097–5149.

[4] Cole Beasley and Azza Abouzied. 2024. Pipe(line) Dreams: Fully Automated End-to-End Analysis and Visualization. In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics* (Santiago, AA, Chile) *(HILDA 24)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3665939.3665962

[5] Pierre Boyeau, Anastasios N. Angelopoulos, Nir Yosef, Jitendra Malik, and Michael I. Jordan. 2024. AutoEval Done Right: Using Synthetic Data for Model Evaluation. *CoRR* abs/2403.07008 (2024). https://doi.org/10.48550/ARXIV.2403.07008 arXiv:2403.07008

[6] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* 189 (11 1995).

[7] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. 2021. To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–21.

[8] Myra Cheng, Esin Durmus, and Dan Jurafsky. 2023. Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers,

6.1 过度依赖人工智能辅助的风险 LLM流程创建

读者可能会问，用户是否以及在多大程度上过度依赖或受到助手解决方案的偏见影响。即使助手帮助用户学习平台（部分参与者

是否存在其提出的解决方案质量不佳或
在更直接的可用性之外存在某种误导性的风险
错误？例如，想象用户要求ChainBuddy设置一个
流程来评估大型语言模型的性别偏见。在此任务中，ChainBuddy
通常会建立一个流程，比较是否存在
社会身份标识（男性、女性、非二元性别）会改变大型语言模型
解决数学问题的成功率——这是基准测试论文[8]中常用的“角色”方法。虽然这是评估

性别偏见的一种方式，但它显然没有穷尽所有可能性，也未能
当前助手帮助用户“扩大”其分析范围，选择
从备选方案中挑选一个解决方案并权衡取舍（一种源自
变异理论[30]的学习技巧），或帮助他们规避
对随机噪声的误解。在我们的研究中，我们确实遇到一个
错误案例，其中ChainBuddy构建了一个错误的解决方案，
该方案未比较两个提示。参与者显然没有
能够识别并修正人工智能的错误。这引发了一个问题
即未来的助手如何在帮助用户学习界面的同时，
又能减少潜在的偏见。

6.2 收集用户需求的正确交互设计是什么?

人工智能辅助工作流生成是一个新兴的研究领域，需要获取用户需求。然而，目前尚不清楚不同界面设计在获取用户意图时的相对权衡是什么。过去的方法仅允许单一提示[27, 53]，采用“无限提问者”模式，每次只问一个问题，直到用户按下生成[9]，或分阶段停止生成以在继续前征求用户反馈[22]。在此，我们探索了一种结构化表单填写方法，将聊天与三问题表单相结合，用户可选择回答哪些问题；部分原因是我们的早期试点研究表明，非结构化的无限提问者既耗时又给用户带来压力。我们研究中的用户对我们的方法表示赞赏，

然而，目前仍不清楚什么是获取用户需求的“正确”方法。特别是，我们认为虽然功能性很重要，

但在聊天窗口中使用表单填写的方式显得格格不入。
未来工作可以研究不同界面在获取高复杂性生成任务的用户需求时的优缺点，或许还能将研究结果与Kazemitabaar等人提出的基于交互式“引导”的方法进行比较。

[22]。

6.3 未来工作

我们的工作仅为这一新兴研究领域提供了一个起点
AutoLLMOps领域：人工智能辅助设计与实现
LLM管道。我们总结了未来工作的潜在方向
（除了将聊天机器人连接到接口文档这一显而易见的方案外）

。

促进在聊天中导入外部数据。意图引导过程可能涉及用户直接引用数据、

例如电子表格、内部文档或在线资源。

未来的ChainBuddy系统可能允许用户导入他们自己的
数据或其他上下文信息到系统中，例如
通过拖放界面。支持数据导入后，用户
可利用现有数据集或代理相关的上下文，
例如将其转换并添加到表格数据节点中。

支持编辑现有链。另一项重要改进是允许用户编辑现有链。目前，

ChainBuddy支持创建新链，但缺乏创建后的修改功能。我们研究中的部分参与者表达了编辑需求。

提供用户对人工智能决策的控制权。我们可考虑让用户对人工智能代理的决策过程拥有更细粒度的控制权。这需要开发允许用户影响和引导代理

决策更加直接，确保结果与他们的期望和需求紧密契合。例如，若用户意图仍不明确，负责输入数据生成的下游代理可能需要向用户提问以澄清模糊之处。然而，鉴于部分参与者认为代理应懂得何时“停止”追问其意图，这可能会以略微增加用户操作为代价，换取更精准的生成结果。

提供多样化的建议链。最后，我们计划增加ChainBuddy提供的建议链和节点种类。当前版本仅包含ChainForge中最常见的少数节点，缺失了如连接和拆分等节点类型。

通过提供更广泛的预配置链和模板，
用户将拥有更多选择空间，从而满足不同用例和偏好需求。这种多样性将帮助用户找到更贴合其具体需求的起始方案，进一步缓解“空白页问题”。

参考文献

[1] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 304, 18 pages. https://doi.org/10.1145/3613904.3642016

[2] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the Whole Exceed its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 81, 16 pages. https://doi.org/10.1145/3411764.3445717

[3] Rafael Barbudo, Sebastián Ventura, and José Raúl Romero. 2023. Eight years of AutoML: categorisation, review and trends. *Knowledge and Information Systems* 65, 12 (2023), 5097–5149.

[4] Cole Beasley and Azza Abouzied. 2024. Pipe(line) Dreams: Fully Automated End-to-End Analysis and Visualization. In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics* (Santiago, AA, Chile) *(HILDA 24)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3665939.3665962

[5] Pierre Boyeau, Anastasios N. Angelopoulos, Nir Yosef, Jitendra Malik, and Michael I. Jordan. 2024. AutoEval Done Right: Using Synthetic Data for Model Evaluation. *CoRR* abs/2403.07008 (2024). https://doi.org/10.48550/ARXIV.2403.07008 arXiv:2403.07008

[6] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* 189 (11 1995).

[7] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. 2021. To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–21.

[8] Myra Cheng, Esin Durmus, and Dan Jurafsky. 2023. Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers,

Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1504–1532. https://doi.org/10.18653/V1/2023.ACL-LONG.84
[9] Yu Cheng, Jieshan Chen, Qing Huang, Zhenchang Xing, Xiwei Xu, and Qinghua Lu. 2024. Prompt sapper: a LLM-empowered production tool for building AI chains. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–24.
[10] CrewAI. 2024. CrewAI: AI Agents for real use cases. https://www.crewai.com Accessed: 2024-09-08.
[11] Harrison Chase et al. 2023. LangChain. https://pypi.org/project/langchain/.
[12] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research* 23, 261 (2022), 1–61.
[13] Flowise AI. 2024. Flowise AI: Visual Toolkit for Prompt Engineering. https://flowiseai.com/
[14] Saumya Gandhi, Ritu Gala, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. 2024. Better Synthetic Data by Retrieving and Transforming Existing Datasets. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 6453–6466. https://doi.org/10.18653/V1/2024.FINDINGS-ACL.385
[15] Andrew D Gordon, Carina Negreanu, José Cambronero, Rasika Chakravarthy, Ian Drosos, Hao Fang, Bhaskar Mitra, Hannah Richardson, Advait Sarkar, Stephanie Simmons, et al. 2023. Co-audit: tools to help humans double-check AI-generated content. *arXiv preprint arXiv:2310.01297* (2023).
[16] Guardrails 2023. Guardrails AI. https://github.com/guardrails-ai/guardrails.
[17] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9
[18] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. 2020. DeepLine: AutoML Tool for Pipelines Generation using Deep Reinforcement Learning and Hierarchical Actions Filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2103–2113. https://doi.org/10.1145/3394486.3403261
[19] Instructor 2023. Instructor, Generating Structure from LLMs. https://github.com/instructor-ai/instructor.
[20] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-based Prototyping with Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. https://doi.org/10.1145/3491101.3503564
[21] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. LLM Comparator: Visual Analytics for Side-by-Side Evaluation of Large Language Models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 216, 7 pages. https://doi.org/10.1145/3613905.3650755
[22] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Zachary Henley, Carina Negreanu, and Advait Sarkar. 2024. Improving Steering and Verification in AI-Assisted Data Analysis with Interactive Task Decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 92, 19 pages. https://doi.org/10.1145/3654777.3676345
[23] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *CoRR* abs/2310.03714 (2023). https://doi.org/10.48550/ARXIV.2310.03714 arXiv:2310.03714
[24] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2024. EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 306, 21 pages. https://doi.org/10.1145/3613904.3642216
[25] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 43–52. https://doi.org/10.1145/2047196.2047202
[26] Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology* 77, 6 (1999), 1121.
[27] Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '12)*. Association for Computing Machinery, New York, NY, USA, 1003–1012. https://doi.org/10.1145/2145204.2145354
[28] Alexandra Kuznetsova, Per B. Brockhoff, and Rune H. B. Christensen. 2017. lmerTest Package: Tests in Linear Mixed Effects Models. *Journal of Statistical Software* 82, 13 (2017), 1–26. https://doi.org/10.18637/jss.v082.i13
[29] LangChain AI. 2024. LangGraph: Documentation. https://langchain-ai.github.io/langgraph/ Accessed: 2024-07-19.
[30] Mun Ling Lo. 2012. *Variation theory and the improvement of teaching and learning*. Göteborg: Acta Universitatis Gothoburgensis.
[31] Logspace. 2023. LangFlow. https://www.langflow.org/.
[32] Steven G Luke. 2017. Evaluating significance in linear mixed-effects models in R. *Behavior research methods* 49 (2017), 1494–1502.
[33] Xiao Ma, Swaroop Mishra, Ariel Liu, Sophie Ying Su, Jilin Chen, Chinmay Kulkarni, Heng-Tze Cheng, Quoc Le, and Ed Chi. 2024. Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 56, 12 pages. https://doi.org/10.1145/3613905.3651093
[34] Yohei Nakajima. 2024. BabyAGI: An AI Agent That Can Achieve Goals and Execute Tasks. https://github.com/yoheinakajima/babyagi
[35] Jesús Sánchez Cuadrado, Sara Pérez-Soler, Esther Guerra, and Juan De Lara. 2024. Automating the Development of Task-oriented LLM-based Chatbots. In *Proceedings of the 6th ACM Conference on Conversational User Interfaces* (Luxembourg, Luxembourg) *(CUI '24)*. Association for Computing Machinery, New York, NY, USA, Article 11, 10 pages. https://doi.org/10.1145/3640794.3665538
[36] Omar Shaikh, Michelle S. Lam, Joey Hejna, Yijia Shao, Michael S. Bernstein, and Diyi Yang. 2024. Show, Don't Tell: Aligning Language Models with Demonstrated Feedback. *CoRR* abs/2406.00888 (2024). https://doi.org/10.48550/ARXIV.2406.00888 arXiv:2406.00888
[37] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, J. D. Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G. Parameswaran, and Eugene Wu. 2024. spade: Synthesizing Data Quality Assertions for Large Language Model Pipelines. *Proc. VLDB Endow.* 17, 12 (Aug. 2024), 4173–4186. https://doi.org/10.14778/3685800.3685835
[38] Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 131, 14 pages. https://doi.org/10.1145/3654777.3676450
[39] Arnav Singhvi, Manish Shetty, Shangyin Tan, Christopher Potts, Koushik Sen, Matei Zaharia, and Omar Khattab. 2023. DSPy Assertions: Computational Constraints for Self-Refining Language Model Pipelines. *CoRR* abs/2312.13382 (2023). https://doi.org/10.48550/ARXIV.2312.13382 arXiv:2312.13382
[40] Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954* (2024).
[41] Synaptic Labs. 2024. Professor Synapse. https://github.com/ProfSynapse/Synapse_CoR Accessed: 2024-06-20.
[42] Xin Tan, Yimin Jiang, Yitao Yang, and Hong Xu. 2024. Teola: Towards End-to-End Optimization of LLM-based Applications. *CoRR* abs/2407.00326 (2024). https://doi.org/10.48550/ARXIV.2407.00326 arXiv:2407.00326
[43] Priyan Vaithilingam, Ian Arawjo, and Elena L. Glassman. 2024. Imagining a Future of Designing with AI: Dynamic Grounding, Constructive Negotiation, and Sustainable Motivation. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) *(DIS '24)*. Association for Computing Machinery, New York, NY, USA, 289–300. https://doi.org/10.1145/3643834.3661525
[44] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. https://doi.org/10.1145/3613904.3642639
[45] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 2609–2634. https://doi.org/10.18653/V1/2023.ACL-LONG.147
[46] Ian Webster. 2023. promptfoo: Test your prompts. https://www.promptfoo.dev/.
[47] Marty J Wolf, K Miller, and Frances S Grodzinsky. 2017. Why we should have seen that coming: comments on Microsoft's tay" experiment," and wider implications. *Acm Sigcas Computers and Society* 47, 3 (2017), 54–64.

Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1504–1532. https://doi.org/10.18653/V1/2023.ACL-LONG.84
[9] Yu Cheng, Jieshan Chen, Qing Huang, Zhenchang Xing, Xiwei Xu, and Qinghua Lu. 2024. Prompt sapper: a LLM-empowered production tool for building AI chains. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–24.
[10] CrewAI. 2024. CrewAI: AI Agents for real use cases. https://www.crewai.com Accessed: 2024-09-08.
[11] Harrison Chase et al. 2023. LangChain. https://pypi.org/project/langchain/.
[12] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research* 23, 261 (2022), 1–61.
[13] Flowise AI. 2024. Flowise AI: Visual Toolkit for Prompt Engineering. https://flowiseai.com/
[14] Saumya Gandhi, Ritu Gala, Vijay Viswanathan, Tongshuang Wu, and Graham Neubig. 2024. Better Synthetic Data by Retrieving and Transforming Existing Datasets. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 6453–6466. https://doi.org/10.18653/V1/2024.FINDINGS-ACL.385
[15] Andrew D Gordon, Carina Negreanu, José Cambronero, Rasika Chakravarthy, Ian Drosos, Hao Fang, Bhaskar Mitra, Hannah Richardson, Advait Sarkar, Stephanie Simmons, et al. 2023. Co-audit: tools to help humans double-check AI-generated content. *arXiv preprint arXiv:2310.01297* (2023).
[16] Guardrails 2023. Guardrails AI. https://github.com/guardrails-ai/guardrails.
[17] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9
[18] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. 2020. DeepLine: AutoML Tool for Pipelines Generation using Deep Reinforcement Learning and Hierarchical Actions Filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2103–2113. https://doi.org/10.1145/3394486.3403261
[19] Instructor 2023. Instructor, Generating Structure from LLMs. https://github.com/instructor-ai/instructor.
[20] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. PromptMaker: Prompt-based Prototyping with Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 35, 8 pages. https://doi.org/10.1145/3491101.3503564
[21] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. LLM Comparator: Visual Analytics for Side-by-Side Evaluation of Large Language Models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 216, 7 pages. https://doi.org/10.1145/3613905.3650755
[22] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Zachary Henley, Carina Negreanu, and Advait Sarkar. 2024. Improving Steering and Verification in AI-Assisted Data Analysis with Interactive Task Decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 92, 19 pages. https://doi.org/10.1145/3654777.3676345
[23] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. *CoRR* abs/2310.03714 (2023). https://doi.org/10.48550/ARXIV.2310.03714 arXiv:2310.03714
[24] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2024. EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 306, 21 pages. https://doi.org/10.1145/3613904.3642216
[25] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 43–52. https://doi.org/10.1145/2047196.2047202
[26] Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology* 77, 6 (1999), 1121.
[27] Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '12)*. Association for Computing Machinery, New York, NY, USA, 1003–1012. https://doi.org/10.1145/2145204.2145354
[28] Alexandra Kuznetsova, Per B. Brockhoff, and Rune H. B. Christensen. 2017. lmerTest Package: Tests in Linear Mixed Effects Models. *Journal of Statistical Software* 82, 13 (2017), 1–26. https://doi.org/10.18637/jss.v082.i13
[29] LangChain AI. 2024. LangGraph: Documentation. https://langchain-ai.github.io/langgraph/ Accessed: 2024-07-19.
[30] Mun Ling Lo. 2012. *Variation theory and the improvement of teaching and learning*. Göteborg: Acta Universitatis Gothoburgensis.
[31] Logspace. 2023. LangFlow. https://www.langflow.org/.
[32] Steven G Luke. 2017. Evaluating significance in linear mixed-effects models in R. *Behavior research methods* 49 (2017), 1494–1502.
[33] Xiao Ma, Swaroop Mishra, Ariel Liu, Sophie Ying Su, Jilin Chen, Chinmay Kulkarni, Heng-Tze Cheng, Quoc Le, and Ed Chi. 2024. Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 56, 12 pages. https://doi.org/10.1145/3613905.3651093
[34] Yohei Nakajima. 2024. BabyAGI: An AI Agent That Can Achieve Goals and Execute Tasks. https://github.com/yoheinakajima/babyagi
[35] Jesús Sánchez Cuadrado, Sara Pérez-Soler, Esther Guerra, and Juan De Lara. 2024. Automating the Development of Task-oriented LLM-based Chatbots. In *Proceedings of the 6th ACM Conference on Conversational User Interfaces* (Luxembourg, Luxembourg) *(CUI '24)*. Association for Computing Machinery, New York, NY, USA, Article 11, 10 pages. https://doi.org/10.1145/3640794.3665538
[36] Omar Shaikh, Michelle S. Lam, Joey Hejna, Yijia Shao, Michael S. Bernstein, and Diyi Yang. 2024. Show, Don't Tell: Aligning Language Models with Demonstrated Feedback. *CoRR* abs/2406.00888 (2024). https://doi.org/10.48550/ARXIV.2406.00888 arXiv:2406.00888
[37] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, J. D. Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G. Parameswaran, and Eugene Wu. 2024. spade: Synthesizing Data Quality Assertions for Large Language Model Pipelines. *Proc. VLDB Endow.* 17, 12 (Aug. 2024), 4173–4186. https://doi.org/10.14778/3685800.3685835
[38] Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 131, 14 pages. https://doi.org/10.1145/3654777.3676450
[39] Arnav Singhvi, Manish Shetty, Shangyin Tan, Christopher Potts, Koushik Sen, Matei Zaharia, and Omar Khattab. 2023. DSPy Assertions: Computational Constraints for Self-Refining Language Model Pipelines. *CoRR* abs/2312.13382 (2023). https://doi.org/10.48550/ARXIV.2312.13382 arXiv:2312.13382
[40] Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954* (2024).
[41] Synaptic Labs. 2024. Professor Synapse. https://github.com/ProfSynapse/Synapse_CoR Accessed: 2024-06-20.
[42] Xin Tan, Yimin Jiang, Yitao Yang, and Hong Xu. 2024. Teola: Towards End-to-End Optimization of LLM-based Applications. *CoRR* abs/2407.00326 (2024). https://doi.org/10.48550/ARXIV.2407.00326 arXiv:2407.00326
[43] Priyan Vaithilingam, Ian Arawjo, and Elena L. Glassman. 2024. Imagining a Future of Designing with AI: Dynamic Grounding, Constructive Negotiation, and Sustainable Motivation. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference* (Copenhagen, Denmark) *(DIS '24)*. Association for Computing Machinery, New York, NY, USA, 289–300. https://doi.org/10.1145/3643834.3661525
[44] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. https://doi.org/10.1145/3613904.3642639
[45] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 2609–2634. https://doi.org/10.18653/V1/2023.ACL-LONG.147
[46] Ian Webster. 2023. promptfoo: Test your prompts. https://www.promptfoo.dev/.
[47] Marty J Wolf, K Miller, and Frances S Grodzinsky. 2017. Why we should have seen that coming: comments on Microsoft's tay" experiment," and wider implications. *Acm Sigcas Computers and Society* 47, 3 (2017), 54–64.

[48] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *CoRR* abs/2308.08155 (2023). https://doi.org/10.48550/ARXIV.2308.08155 arXiv:2308.08155

[49] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 359, 10 pages. https://doi.org/10.1145/3491101.3519729

[50] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 385, 22 pages. https://doi.org/10.1145/3491102.3517582

[51] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

[52] Chengbo Zheng, Dakuo Wang, April Yi Wang, and Xiaojuan Ma. 2022. Telling Stories from Computational Notebooks: AI-Assisted Presentation Slides Creation for Presenting Data Science Work. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 53, 20 pages. https://doi.org/10.1145/3491102.3517615

[53] Zhongyi Zhou, Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, et al. 2023. InstructPipe: Building Visual Programming Pipelines with Human Instructions. *arXiv preprint arXiv:2312.09672* (2023).

A Technical Evaluation Details

Here we report more details and results from our a technical evaluation (Section 3.4).

A.1 Expert Rating Guidelines

Here is the prompt we gave the expert raters for their task:

> "Score the workflow quality from 1 to 5, with 5 the highest score and 1 the lowest score, based on how well the workflow satisfied the requirement(s) in the task description. Keep in mind that these workflows are meant to pose a good starting point to kickstart user's explorations. Also focus on the quality of the workflow, rather than the quality of the interface."

To help align experts in the under-specified domain of workflow evaluation, we also provided the following specific descriptions of what constitutes a score of 1 (lowest and least aligned to the user requirements) to 5 (most aligned with no errors and no fixes required):

(1) Not aligned to the requirements at all.
(2) Slightly aligned to the requirements, but with some significant misunderstanding, missing features, or error.
(3) Aligned to the requirements. The workflow is a good starting point for users, but has at least one significant problem or missing feature (either missing an important part, or having 1-2 incorrect implementations of nodes or connections).
(4) Aligned to the requirements, with only 1-2 minor issues. Any errors are easily fixable and the workflow is extensible.
(5) Strongly aligned to the requirements. The workflow represents a great starting point for the user, and does not require any fixes to its existing content.

A.2 Results from Technical Evaluation

In the table below, we report all prompts from our technical evaluation, alongside rater scores and averaged scores, with the qualitative feedback summarized in the Issues column.

[48] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *CoRR* abs/2308.08155 (2023). https://doi.org/10.48550/ARXIV.2308.08155 arXiv:2308.08155

[49] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 359, 10 pages. https://doi.org/10.1145/3491101.3519729

[50] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 385, 22 pages. https://doi.org/10.1145/3491102.3517582

[51] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

[52] Chengbo Zheng, Dakuo Wang, April Yi Wang, and Xiaojuan Ma. 2022. Telling Stories from Computational Notebooks: AI-Assisted Presentation Slides Creation for Presenting Data Science Work. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 53, 20 pages. https://doi.org/10.1145/3491102.3517615

[53] Zhongyi Zhou, Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, et al. 2023. InstructPipe: Building Visual Programming Pipelines with Human Instructions. *arXiv preprint arXiv:2312.09672* (2023).

A 技术评估详情

此处我们报告更多来自技术评估（第3.4节）的细节与结果。

A.1 专家评分指南

以下是我们提供给专家评分者执行任务的提示：

> "根据工作流满足任务描述中需求的程度，从1到5分对工作流质量进行评分，5分为最高分，1分为最低分。请记住，这些工作流旨在为用户探索提供一个良好的起点。同时，重点关注工作流本身的质量，而非界面质量。"

为了帮助在未明确指定的工作流领域内协调专家意见，我们还提供了以下具体描述来说明1分（最低分且最不符合用户需求）到5分（完全符合且无需修正错误）的评分标准：

(1) 完全不符合需求。
(2) 略微符合需求，但存在显著误解、缺失功能或错误。
(3) 符合需求。该工作流程对用户来说是一个良好的起点，但至少存在一个重大问题或缺失功能（要么缺少重要部分，要么存在1-2处节点或连接的错误实现）。
(4) 符合需求，仅有1-2个小问题。任何错误都易于修复，且工作流程具有可扩展性。
(5) 与需求高度契合。该工作流程为用户提供了极佳的起点，且现有内容无需任何修正。

A.2 技术评估结果

下表中我们列出了技术评估中的所有提示，同时附上评分者分数与平均分数，并将定性反馈总结在问题列中。

Workflow	Prompt	S1	S2	Avg.Score	Summarized Issues
Prompt Engineering - Compare a prompt across models					
Q1.1.1	Design a workflow to test a single prompt across GPT-4, GPT-3.5, and Claude for creative writing tasks. Visualize response length and diversity.	5	5	5.0	No outstanding issues.
Q1.1.2	"I want to compare how GPT-4 and Claude respond to a prompt that provides recipes for making different types of pies. Then, I want to easily evaluate the number of steps in each recipe."	3	5	4.0	Included a prompt comparison when it should not.
Q1.1.3	"I want to see how different models respond to the same prompt for creating an imaginary conversation between characters in a video game."	5	3	4.0	One rater felt evaluation is extensive and appropriate, but another thought it was too extensive ("feels bloated") with too little inputs and too many evaluators.
Prompt Engineering - Compare different prompts					
Q1.2.1	I want to compare different prompts for understanding how professionals from different backgrounds overcome challenges in their lives.	4	4	4.0	Minor issues to Python evaluator, Vis Node choice of variable to plot; tweak needed to prompt in Prompt Node.
Q1.2.2	"Make a workflow to compare different prompts for shortening text while keeping the same content. Check and compare the length of the text across prompts."	5	5	5.0	Input data (example paragraphs) could be longer to test.
Q1.2.3	"Set up a workflow to compare prompts for generating formal yet concise emails given user descriptions of what the email should be about."	4	3	3.5	Could make minor improvement to prompt to clarify expected structure of output; LLM evaluator prompt "seems off" based on the high scores and might need further alignment
Prompt Engineering - Test the robustness of prompt across different inputs					
Q1.3.1	Can you help me check how my prompt responds to many different inputs? The prompt is: "Give me a fun recipe for dessert." I'd like to try this prompt across different desserts.	5	4	4.5	Python evaluator's objective is unclear.
Q1.3.2	Generate a workflow evaluating how commands like 'Be formal' or 'Be casual' influence conversational model outputs. Include a plot of response sentiment.	4	5	4.5	Visualization should plot on a different variable.
Q1.3.3	I have a prompt for rephrasing verbose sentences and want to test for robustness across many different inputs. Here is the prompt: "Rephrase the following sentence to be more concise but keep the same meaning: sentence."	5	2	3.5	Missing an evaluator node to check for length; one rater felt there was a structural issue with the template chaining setup

Table 1: Prompt Engineering Evaluation

工作流程	提示	S1	S2	平均分数	总结问题
提示工程 - 比较不同模型对同一提示的响应					
Q1.1.1	设计工作流程以测试GPT-4、GPT-3.5和Claude在创意写作任务中对单一提示的响应。可视化响应长度和多样性。	5	5	5.0	无未解决问题。
Q1.1.2	"我想比较GPT-4和Claude对提供不同类型派制作食谱的提示如何响应。然后，我想轻松评估每个食谱中的步骤数量。"	3	5	4.0	包含了不应出现的提示比较。
Q1.1.3	"我想看看不同模型对同一提示的响应，该提示用于创建视频游戏中角色之间的虚构对话。"	5	3	4.0	一位评分者认为评估全面且恰当，但另一位认为过于冗长（"感觉臃肿"），输入太少而评估器太多。
Prompt Engineering - Compare different prompts					
Q1.2.1	我想比较不同的提示，以了解来自不同背景的专业人士如何克服生活中的挑战。	4	4	4.0	Python评估器的小问题，可视化节点选择要绘制的变量；提示节点中的提示需要微调。
问题 1.2.2	"设置工作流程以比较不同的提示，用于在保持内容不变的情况下缩短文本。检查并比较不同提示下文本的长度。"	5	5	5.0	输入数据（示例段落）可以更长以便测试。
问题 1.2.3	"设置工作流程以比较提示，用于根据用户描述生成正式而简洁的电子邮件。"	4	3	3.5	可对提示进行小幅改进以明确输出的预期结构；LLM评估器提示"似乎不太对"，基于高分可能需要进一步对齐。
Prompt Engineering - 测试其鲁棒性 prompt跨越不同的输入 puts					
问题 1.3.1	你能帮我检查我的提示如何响应许多不同的输入吗？提示是："给我一个有趣的甜点食谱。"我想在不同的甜点上尝试这个提示。	5	4	4.5	Python评估器的目标不明确。
Q1.3.2	生成一个工作流程，评估诸如'正式'或'随意'等命令如何影响对话模型输出。包含响应情感的可视化图表。	4	5	4.5	可视化应基于不同的变量进行绘制。
Q1.3.3	我有一个用于重写冗长句子的提示，并想测试其在多种不同输入下的鲁棒性。提示如下："将以下句子重写得更简洁但保持相同含义：句子。"	5	2	3.5	缺少评估节点来检查长度；一位评分者认为模板链式设置存在结构问题。

表1: 提示工程评估

Workflow	Prompt	S1	S2	Avg.Score	Summarized Issues
Auditing - Checking accuracy					
Q2.1.1	Create a workflow to benchmark LLMs using a set of trivia questions. Evaluate responses for correctness.	5	3	4.0	Python evaluator logic must be amended.
Q2.1.2	"I want to compare how different models reply to the same math question across three different languages: English, German, and Russian." Evaluate the accuracy of the response."	4	5	4.5	Evaluation is limited only to Russian; evaluator choice could be better
Q2.1.3	"Design a workflow to test factual accuracy of LLM responses on historical questions about Ancient Rome."	5	4	4.5	One rater felt that LLM scorer was suboptimal and instead the system should use a RAG approach [note: the baseline interface does not have support for evaluating RAG]
Auditing - Bias Detection					
Q2.2.1	Generate a workflow to detect cultural bias in LLM outputs by varying names and locations in prompts.	5	5	5.0	LLM scorer prompt should be refined.
Q2.2.2	"I want to check whether giving a model a persona across different gender and racial identities changes how well they reply to mathematical problems."	3	4	3.5	Includes an unnecessary LLM scorer not asked for by the user; visualization node should plot a different variable
Q2.2.3	"Have the model describe different jobs by only varying demographic variables (e.g., gender, ethnicity) and see if responses differ across these variables. For instance, you might compare how it describes a female doctor's job versus a male doctor. Grade the sentiment of the outputs."	4	3	3.5	Prompt in LLM scorer needs to be revised; input data could vary more demographic variables
Auditing - Ethics and Red-teaming					
Q2.3.1	I want to check how different models respond to ethical dilemmas and whether they make more altruistic choices or not. Make it easy to evaluate the altruism of the response	5	5	5.0	No outstanding issues.
Q2.3.2	"I'm curious in what circumstances different models will refuse to respond given their safety filters. Generate a workflow to help me check across models whether the model refuses to reply to a given question. Evaluate whether the output is a refusal to answer the question or not."	5	4	4.5	Evaluation could be improved by adding more types of evaluations; visualization node should plot a different variable
Q2.3.3	"Design a workflow to help me classify the toxicity of social media comments, across different possible prompts to GPT-4. Score the toxicity on a scale of 1-5, with 5 the most toxic, and plot the results so that I can compare."	3	4	3.5	Prompts vary in quality; could possibly add multiple models.

Table 2: LLM Auditing Evaluation

工作流程	提示	S1	S2	平均分数	总结问题
审计 - 检查准确性					
问题 2.1.1	创建工作流程以使用一组琐事问题对大型语言模型进行基准测试。评估响应的正确性。	5	3	4.0	必须修改Python评估器逻辑。
Q2.1.2	"我想比较不同模型如何用三种不同语言（英语、德语和俄语）回答同一道数学问题。"评估响应的准确性。	4	5	4.5	评估仅限俄语；评分者选择可优化
Q2.1.3	"设计工作流程以测试大型语言模型对古罗马历史问题响应的史实准确性。"	5	4	4.5	一位评分者认为LLM评分器不够理想，建议系统采用RAG方法[注：基线界面不支持评估RAG]
审计 - 偏见检测					
问题 2.2.1	生成一个工作流程，通过改变提示中的名称和地点来检测LLM输出中的文化偏见。	5	5	5.0	LLM评分器提示应重新优化。
问题 2.2.2	“我想验证赋予模型跨越不同性别和种族身份的角色后，其回复数学问题的表现是否会发生变化。”	3	4	3.5	包含了用户未要求的冗余LLM评分器； 可视化节点应绘制不同的变量
问题 2.2.3	“让模型仅通过改变人口统计变量（如性别、种族）来描述不同职业，并观察响应是否随这些变量变化。例如，可以比较其对女医生与男医生职业描述的差异。对输出的情感倾向进行评分。”	4	3	3.5	LLM评分器的提示需修改；输入数据应涵盖更多人口统计变量
审计 - 伦理与红队测试					
问题 2.3.1	我想检查不同模型如何应对道德困境，以及它们是否会做出更多利他选择。让评估响应的利他主义变得容易。	5	5	5.0	无未解决问题。
Q2.3.2	“I'm curious in what 情况下 不同的 模型 will 拒绝 to respond given their safety filters. Generate a workflow to help me check across models whether the model refuses to reply to a given question. Evaluate whether the output is a refusal to answer the question or not.” 鉴于其安全过滤器，生成一个工作流程来帮助我检查。 在不同模型中，该模型是否拒绝回答给定的问题。 评估输出是否为拒绝回答问题。	5	4	4.5	可通过增加更多评估类型来改进评估； 可视化节点应绘制不同的变量
问题 2.3.3	“设计工作流程以帮助我对社交媒体评论的毒性进行分类，针对GPT-4的不同提示进行毒性评分 采用1-5等级标准（5为毒性最高），并绘制结果以便我进行比较。”	3	4	3.5	提示的质量参差不齐；可以考虑添加多个模型。

表2：LLM审计评估

Workflow	Prompt	S1	S2	Avg.Score	Summarized Issues
Data Processing - Process and clean input data					
Q3.1.1	Build a workflow to clean up ASR transcript excerpts that have some mistakes, revising the transcripts to correct grammar or spelling issues but not changing the meanings	4	5	4.5	Missing evaluation to double-check the output of the LLM.
Q3.1.2	"I want to extract information from different emails and output a JSON format with dictionary keys "importance", "sender", "recipient", and "description".	5	4	4.5	Should have use a Python evaluator instead of the LLM Scorer to check JSON format of outputs; prompt could be clarified slightly in what the definition of "importance" is
Q3.1.3	"I'm curious if asking for different formatting (e.g., asking a model to reply in JSON, YAML, or Markdown) impacts the content of the outputs. Make a workflow to help explore this."	4	4	4.0	Visualization should plot on different variable; LLM scorer prompt needs refinement
Data Processing - Data Augmentation and Generation					
Q3.2.1	Give me a workflow that generates paraphrased versions of sentences for data augmentation in sentiment analysis.	5	5	5.0	No outstanding issues.
Q3.2.2	"I want to use LLMs to generate example long paragraphs for a summarization task. Use multiple models for some diversity."	4	3	3.5	Could have added more models; might have added an LLM scorer to double-check quality of outputs
Q3.2.3	"Design a workflow that converts all input questions from English into a target language (e.g., French, Korean). Evaluate the language of the response to double-check."	4	4	4.0	Visualization should plot on different variable; could have explored more prompt variations for the translation task
Data Processing - Entity Extraction					
Q3.3.1	Create a workflow to extract all dates and locations mentioned in LLM responses to a historical question.	2	2	2.0	The prompt template in Prompt Node is not aligned; and the evaluation is incorrect/an approach that will struggle to generalize
Q3.3.2	"I want to check how well different models understand different famous inventors and the entities that they put in their responses. Extract the named entities so that I can compare."	3	3	3.0	Does not compare different models in the Prompt Node, which was a necessary requirement, but otherwise appropriate
Q3.3.3	"I want to extract the names and public figures mentioned in some sample tweets. Exclude entities which appear in hashtags and output the entities as a comma-separated list."	3	5	4.0	Code evaluator is "too stringent", does not address enough edge cases; visualization node plot appears empty

Table 3: Data Processing Evaluation

工作流程	提示	S1	S2	平均分数	总结问题
数据处理	**g - 处理并清理输入 p数据**				
Q3.1.1	构建一个工作流程来清理存在错误的ASR转录文本片段，修正转录文本中的语法或拼写问题但不改变原意	4	5	4.5	缺少对大型语言模型输出的二次核查评估。
Q3.1.2	“我想从不同的电子邮件中提取信息，并以JSON格式输出，包含字典键“importance”（重要性）、“sender”（发件人）、“recipient”（收件人）和“description”（描述）。”	5	4	4.5	应使用Python评估器而非LLM评分器来检查输出的JSON格式；提示中关于“importance”（重要性）的定义可稍作澄清
Q3.1.3	“我很好奇要求不同的格式化（例如，要求模型以JSON、YAML或Markdown回复）是否会影响输出的内容。建立一个工作流程来探索这一点。”	4	4	4.0	可视化应绘制不同的变量；LLM评分器提示需要改进
Data Pro处理 - 数据增强与生成					
问题 3.2.1	提供一个工作流程，用于生成句子的改写版本，以进行情感分析的数据增强。	5	5	5.0	无未解决问题。
问题 3.2.2	"我想使用大型语言模型生成示例长段落，用于摘要任务。使用多个模型以增加多样性。"	4	3	3.5	本可以添加更多模型；或许可以增加一个LLM评分器进行双重
Q3.2.3	“设计一个工作流程，将所有输入问题从英语转换为目标语言（例如法语、韩语）。评估响应语言以进行双重检查。”	4	4	4.0	-检查 q质量y 的输出puts 可视化应在不同变量上绘制；本可探索更多提示变化以完成翻译任务
数据处理	**g - 实体 y 提取**				
问题 3.3.1	创建工作流程以提取大型语言模型对历史问题的响应中提及的所有日期和地点。	2	2	2.0	提示节点中的提示模板未对齐；且评估不正确/一种难以泛化的方法
问题 3.3.2	“我想检查不同模型对著名发明家及其在响应中提及的命名实体的理解程度。提取命名实体以便我进行比较。”	3	3	3.0	未在提示节点中比较不同模型（这是必要要求），但其他方面合适
问题 3.3.3	“我想提取一些示例推文中提到的名称和公众人物。排除出现在标签中的实体，并将实体输出为逗号分隔的列表。”	3	5	4.0	代码评估器"过于严苛"，未能覆盖足够多的边缘用例；可视化节点图显示为空

表3：数据处理评估

Planner

```
###Role: You are a Planner. Your responsibility is to create a detailed plan to help your team
set up a correct chain ……

###Workflow Design Principles:
•Think in Layers:
[Explanation of how the workflow is divided into Input, Middle, and Evaluation Layers.]
•Node Responsibilities:
    •[Available Nodes in each layer.]
    •[Usage of TextFieldsNodes.]
    •[Usage of Intermediate PromptNodes.]
•Variable Traces: [Explanation of how variables are tracked across the workflow.]
•Reasoning and Skeleton Design:
 [Details of how the Planner creates a skeleton workflow design to justify the structure.]

###Process
1.Understand Requirements: [Description of analyzing user requirements.]
2.Layer-Based Design: [Steps to divide the workflow into logical layers and design connections.]
3.Justify Design: [Description of how to validate the workflow against user objectives.]
4.Output the Plan: [Steps to generate the final JSON plan, including nodes, variable traces.]

###examples:[example]
```

Figure 7: Planner agent system prompt. Note that this prompt is very long and hence truncated for length here, with details for each section summarized in square braces.

B Walkthrough of System Architecture: Comparing Prompts for Generating Catchy Tweets

Here we walk through an example of how our system architecture works, from the initial user requirements (prompt) to the generation of a workflow.

Imagine a user provides the following input to the system, which is similar to Task 2 in our user study:

> "I want to compare different prompts that can generate catchy tweets from long paragraphs. The generated tweets should be concise, staying within the 144-character limit."

The **Planner Agent** receives the user requirement as context to a long prompt (Figure 7). The Planner prompt decomposes the workflow into suggested nodes segmented by layers (described below), assigns variables, and traces connections across nodes. First, it generates the workflow skeleton following a *layer-based design* that segments the parts of evaluative LLM pipelines into three general sections (B.2):

- **Input Layer:** Provides fixed inputs to the workflow that do not include any template variables.
- **Middle Layer(s):** Process and transform input data using template variables and (potentially) prompt chaining. Includes nodes that generate prompts, process responses, or chain variables together.
- **Evaluation Layer:** Includes evaluator nodes for assessing outputs generated by the workflow.

These layers occur in sequence (left-to-right) in the resulting node-edge graph. The reason we use this layer-based design is because later layers like Evaluation depend upon the results of earlier layers (e.g., the type of input data and the specific names of prompt variables). More detail on how we described the layers in the system prompt can be found in Section B.2.

For instance, here is example JSON output of the Planner outlining its high-level workflow plan to address the user requirement via four layers (note that these example outputs are condensed for length):

```
{
  "layer_skeleton": {
    "justification":
    "This workflow transforms long paragraphs into catchy
        tweets.",
    "design": [
      { "layer_id": 1, "name": "Input Layer", "description":
          "Collects long paragraphs for tweet generation."
          },
      { "layer_id": 2, "name": "Template Layer", "
          description": "Generates prompt templates for
          creating concise tweets." },
      { "layer_id": 3, "name": "Prompt Node Layer", "
          description": "Queries LLMs using generated
          prompts to create tweets." },
      { "layer_id": 4, "name": "Evaluation Layer", "
          description": "Evaluates the tweets for adherence
          to character limits." }
    ]
  }
}
```

Expanding on this high-level plan, the Planner then provides implementation examples of nodes corresponding to each layer for the agents to get started. Task-specific agents are then provided with both the high-level plan and each task-specific example that relates to their task, but are not required to stick to the example. The implementation examples is an array of JSON objects like the following:

```
{
    "layer_id": 1,
    "name": "Input Layer",
    "nodes": [
        {
            "node_id": "TextFieldsNode1",
            "type": "TextFieldsNode",
            "description": "Define long paragraphs as
                input",
            "variables": {
                "long_paragraph": [
                    "The quick brown fox jumps over the
                        lazy dog...",
                    "In a world where technology is
                        advancing rapidly...",
                    "The art of communication is not just
                        about speaking..."
                ]
            }
        }
    ]
},
{
    "layer_id": 2,
    "name": "Template Layer",
    "nodes": [
        {
            "node_id": "TextFieldsNode2",
            "type": "TextFieldsNode",
```


Planner

```
###Role: You are a Planner. Your responsibility is to create a detailed plan to help your team
set up a correct chain ……

###Workflow Design Principles:
•Think in Layers:
[Explanation of how the workflow is divided into Input, Middle, and Evaluation Layers.]
•Node Responsibilities:
    •[Available Nodes in each layer.]
    •[Usage of TextFieldsNodes.]
    •[Usage of Intermediate PromptNodes.]
•Variable Traces: [Explanation of how variables are tracked across the workflow.]
•Reasoning and Skeleton Design:
 [Details of how the Planner creates a skeleton workflow design to justify the structure.]

###Process
1.Understand Requirements: [Description of analyzing user requirements.]
2.Layer-Based Design: [Steps to divide the workflow into logical layers and design connections.]
3.Justify Design: [Description of how to validate the workflow against user objectives.]
4.Output the Plan: [Steps to generate the final JSON plan, including nodes, variable traces.]

###examples:[example]
```

图7：规划代理系统提示。请注意此提示因长度截断，此处仅展示概要内容，详细部分以方括号内总结形式呈现。

B 系统架构详解：生成吸引人的推文的提示对比

在此我们将通过一个示例来展示我们的系统架构如何运作，从初始用户需求（提示）到最终生成工作流程的全过程。

假设用户向系统提供了如下输入，该输入与我们用户研究中的任务2类似：

> "我想比较不同的提示，这些提示能够从长段落生成吸引人的推文。生成的推文应当简洁，保持在144字符限制内。"

规划代理接收用户需求作为上下文生成一个长提示（图7）。规划器提示将工作流程分解为按层分段的建议节点（下文详述），分配变量，并追踪节点间的连接。首先，它按照基于层的设计生成工作流骨架将评估性LLM流水线的部分划分为三个通用章节(B.2)：

- 输入层: 为工作流程提供不包含任何模板变量的固定输入不包含任何模板变量。
- 中间层: 使用模板变量(可能包含提示链)处理和转换输入数据。包含生成提示、处理响应或将链变量关联的节点。
- 评估层: 包含用于评估工作流程所产生输出的评估节点工作流程生成的输出。

这些层在生成的节点-边图中按顺序（从左到右）排列。我们采用这种基于层的设计是因为后续层（如评估层）依赖于早期层的结果（例如输入数据的类型和提示变量的具体名称）。关于我们如何在系统提示中描述各层的更多细节，请参阅B.2节。

例如，以下是规划器生成的示例JSON输出，概述了其通过四个层解决用户需求的高层工作流计划（请注意这些示例输出已为长度而精简）：

```
{
  "layer_skeleton": {
    "justification":
    "This workflow transforms long paragraphs into catchy
        tweets." ,
    "design": [
      { "layer_id": 1, "name": "输入层", "description":
          "收集用于生成推文的长段落。"
          },
      { "layer_id": 2, "name": "模板图层", "
          description": "生成用于
          创建简洁推文的提示模板。" },
      { "layer_id": 3, "name": "提示节点层", "
          description": "使用生成的
          提示查询LLM以创建推文。" },
      { "layer_id": 4, "name": "评估层", "
          description": "评估推文是否符合
          字符限制。" }
    ]
  }
}
```

基于此高层计划，规划器随后会进一步提供各层对应节点的实现示例代理开始工作。随后会提供任务特定代理包括高层计划及与各自任务相关的每个任务特定示例但无需严格遵循该示例实现示例是一个JSON对象数组，类似如下：

```
{
    "图层_ID": 1,
    "name": "Input Layer",
    "节点": [
        {
            "节点_id": "文本字段节点1"     ,
            "type": "文本字段节点",
            "描述": "将长段落定义为
                输入",
            "变量": {
                "长_段落": [
                    "敏捷的棕色狐狸跳过
                        懒狗...",
                    "在一个技术飞速发展的
                        世界...",
                    "沟通的艺术不仅仅是
                        说话..."
                ]
            }
        }
    ]
},
{
    "层级_标识": 2,
    "名称": "模板图层",
    "节点": [
        {
            "node_id": "文本字段节点2",
            "type": "文本字段节点",
```

```
            "description": "Create prompts for generating
                  catchy tweets",
            "variables": {
                "prompt_template_tweet": [
                    "Summarize the following into a catchy
                          tweet: {long_paragraph}",
                    "Create a tweet from this paragraph,
                         keeping it under 144 characters:
                         {long_paragraph}"
                ]
            }
        }
    ]
},
... ### Other layers
```

Finally, the Planner agent generates a list of template variable names used in the flow, what we call **variable tracing**. Variable tracing ensures each variable is clearly defined, connected, and traceable throughout the workflow, which is particularly necessary for template chaining. Moreover, the tracing allows agents to identify which nodes generate or consume specific variables, making it easier to validate the workflow and reuse any previous information during evaluation. The example variable trace produced by the Planner is an array of JSON objects, e.g.:

```
[
  {
    "variable": "{long_paragraph}",
    "origin": "TextFieldsNode1",
    "consumed_by": ["TextFieldsNode2"]
  },
  {
    "variable": "{prompt_template_tweet}",
    "origin": "TextFieldsNode2",
    "consumed_by": ["PromptNode1"]
  },
  {
    "variable": "{tweet_response}",
    "origin": "PromptNode1",
    "consumed_by": ["EvaluatorNode1", "LLMScorerNode1"]
  }
]
```

After the planner generates its plan, its output is stored in a global state. Each task-specific agent then gets their task list by using tools to retrieve the corresponding plan for each node of their task type. Here are broken-down examples for each agent after the Planner generates the plan for the example user requirements:

B.0.1 TextFields Agent. This agent receives a task list that contains all `TextFieldsNode` specifications from the Planner's output, and populates `TextFieldsNode` with input data based on the task list and ensures correct braces are used for specific variables.

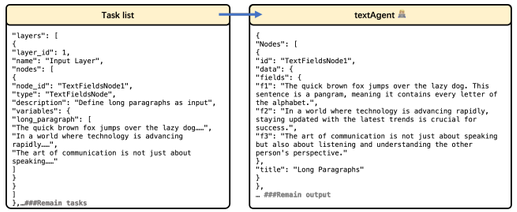

Figure 8: TextFields agent takes the task list from the Planner agent with its example sketch of the node in JSON, and implements it with a complete specification.

B.0.2 Prompt Agent. This agent receives a task list containing all `PromptNode` specifications from the Planner's output, and configures `PromptNodes` to query multiple LLMs (GPT-4, Claude, etc.).

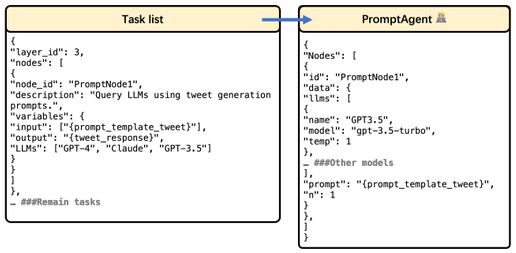

Figure 9: Prompt agent takes the task list from the Planner agent, and implements Prompt Nodes with a complete specification.

B.0.3 Evaluator Agent. This agent receives **user requirements**, **example prompts** provided from the previous agents, and **potential responses** as examples of LLM outputs that will be fed into evaluator:

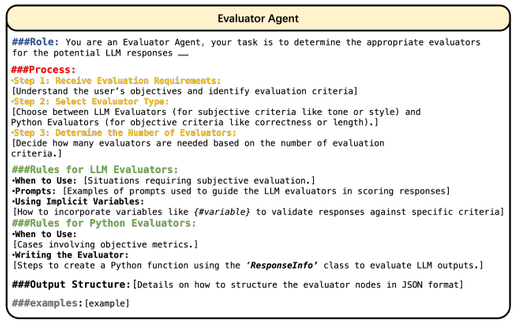

Figure 10: Evaluator agent system prompt (truncated for length)


```
            "description": "创建用于生成的提示
                  吸引人的推文",
            "variables": {
                "提示_模板_tweet": [
                    "将以下内容总结成一条吸引人的
                          推文: {long_paragraph}",
                    "根据这段文字创建一条推文,
                         保持在144个字符以内:
                         {long_paragraph}"
                ]
            }
        }
    ]
},
… ### 其他层
```

最后，规划代理生成一个模板变量列表
流程中使用的名称，我们称之为变量追踪。变量
追踪确保每个变量都有明确定义、相互关联且
在整个工作流程中可追溯，这对于
模板链式调用尤为必要。此外，追踪功能让代理能够识别哪些节点生成或消耗特定变量，
从而
更易于验证工作流程并复用先前的信息
在评估期间。该示例变量追踪由
规划器是一个JSON对象数组，例如：

```
[
  {
    "变量": "{long_paragraph}",
    "来源": "文本字段节点1",
    "consumed_by": ["文本字段节点2"]
  },
  {
    "变量": "{提示_模板_tweet}",
    "来源": "文本字段节点2",
    "consumed_by": ["提示节点1"]
  },
  {
    "变量": "{tweet_响应}"
    "origin": "提示节点1",
    "consumed_by": ["评估器节点1", "大语言模型评分器节点1"]
  }
]
```

规划器生成计划后，其输出会存储于全局状态中。每个任务专属代理随后通过工具检索与其任务类型对应的节点计划来获取任务列表。以下是规划器为示例用户需求生成计划后，各代理的分解示例：

B.0.1 文本字段代理。该代理接收的任务列表包含
规划器输出中的所有文本字段节点规范，以及
根据任务列表用输入数据填充文本字段节点
并确保为特定变量使用正确的大括号。

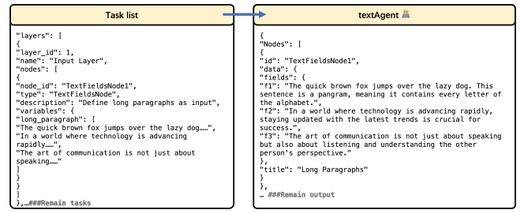

图8：文本字段代理从规划代理处获取任务列表及其JSON格式的节点示例草图，并用完整规范实现该节点。

B.0.2 提示代理。该代理接收来自规划器输出的包含所有提示节点规范的任务列表，并配置提示节点以查询多个大型语言模型（如GPT-4、Claude等）。

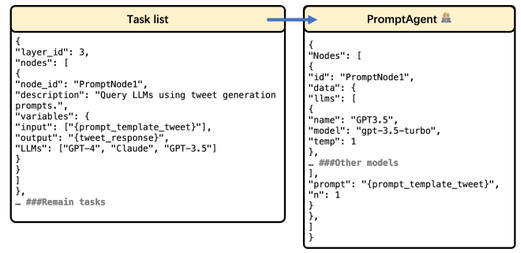

图9：提示代理从规划代理处获取任务列表，并用完整规范实现提示节点。

B.0.3 评估器代理。该代理接收用户需求，
来自先前代理提供的示例提示，以及作为LLM输出示例的潜在响应，这些将被输入评估器：

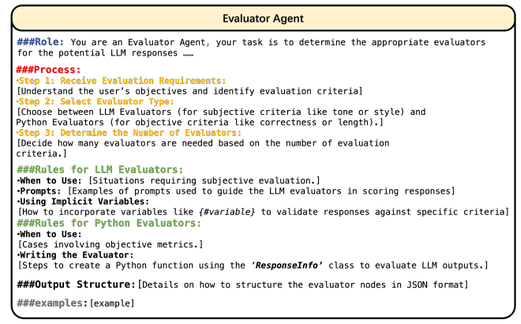

图10：评估器代理系统提示（截断部分长度）

The evaluator agent then produces one or more evaluation nodes that can be a mix of Python Code Evaluators and LLM Scorers, expressed as JSON objects:

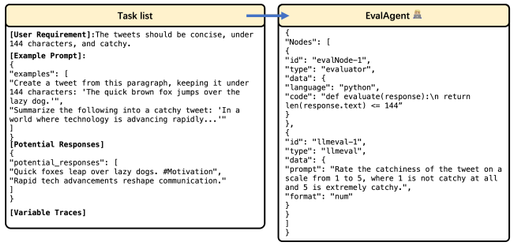

Figure 11: Evaluator agent input and output for the example. For the input (left), we provide example prompts and LLM outputs in addition to the user requirements, to help the evaluator to anticipate the general format of LLM responses that will be passed into it. This agent produces two evaluators: one Python Code Evaluator and one LLM Scorer.

B.0.4 Edge Agent. This agent receives variable traces from the Planner's output, and ensures proper data flow across the workflow by establishing edges between nodes in the format expected by ChainForge:

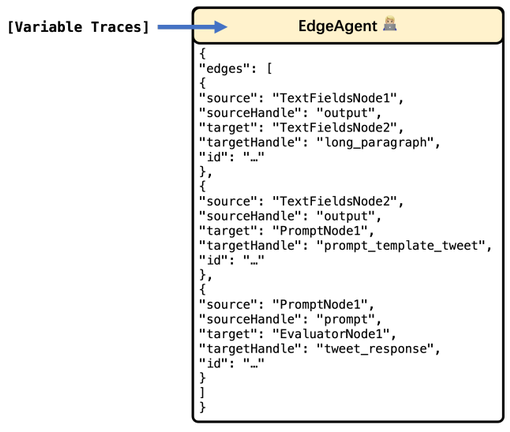

Figure 12: Edge agent takes the variable traces from the Planner agent and the generated nodes, and constructs a list of edges in the JSON format expected by ChainForge.

B.0.5 Layout Agent. This agent receives output from Edge Agent and its responsibility is to simply configure every nodes' position coordinates (from left to right) for the final workflow.

B.1 Final Workflow

Once all agents have completed their tasks, the workflow is assembled with a default configuration template for each type of node and then merged into a single JSON file which is then presented in the interface (see Fig. 2 for an example generation that addresses the same tweet task as here).

B.2 Layer-Based Workflow Skeleton

To generate the entire workflow, the Planner agent translates user requirements into a structured, **layer-based workflow skeleton**. Here we provide further details of how we describe layers in the Planner's system prompt (Fig. 7).

Layer Definitions:

Input Layer (Layer 0):

- **Purpose:** Provides fixed inputs to the workflow that do not include any template variables.
- **Structure:** Consists of nodes (e.g., `TextFieldsNode`) that define raw, static inputs. These inputs serve as the foundation for generating prompts in subsequent layers.
- **Characteristics:**
  - Contains data directly entered by the user (e.g., email content, comments, or datasets).
  - No dynamic variables are used within this layer.

Middle Layers (Layer 1, Layer 2, etc.):

- **Purpose:** Process and transform input data using template variables and prompt chaining.
- **Structure:** Includes nodes that generate prompts, process responses, or chain variables together.
- **Characteristics:**
  - Dynamically incorporates variables generated in earlier layers.
  - Variables and nodes are connected in a sequential, logical flow to ensure proper data transformation.
  - Each variable has a **unique ID** to maintain clarity and prevent conflicts across layers and nodes.

Evaluation Layer (Final Layer):

- **Purpose:** Includes evaluator nodes for assessing outputs generated by the workflow.
- **Structure:** Contains nodes (e.g., `LLMScorerNode`, `CodeEvaluatorNode`) designed to measure outputs against predefined criteria.
- **Characteristics:**
  - Designs different numbers and types of evaluators based on below context given to the evaluator agent:
    - **User Requirements:** Direct criteria for evaluation (e.g., "rate professionalism").
    - **Example Prompts:** Example prompts inside current workflow.
    - **Potential Responses:** Context-aware example LLM outputs that evaluators may expect.

C Participant Ideas in Free Exploration

For the curiosity of readers, here we list the ideas that each participant explored using the assistant in free exploration time.

评估器代理随后生成一个或多个评估节点
可以是Python代码评估器和LLM评分器的混合体，
以JSON对象形式表示：

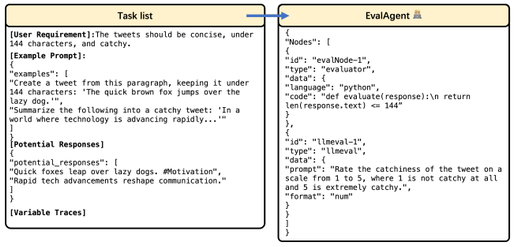

图11：评估器代理的输入与输出示例。
对于输入（左侧），我们提供了示例提示和LLM
除用户需求外的输出，以帮助
评估器预判LLM响应的通用格式
这些响应将被传入其中。该代理生成两个评估器：
一个Python代码评估器和一个LLM评分器。

B.0.4 边代理。该代理接收来自
规划器输出的变量追踪，并确保工作流程中数据流的正确性
通过按照预期格式在节点之间建立边连接
ChainForge:

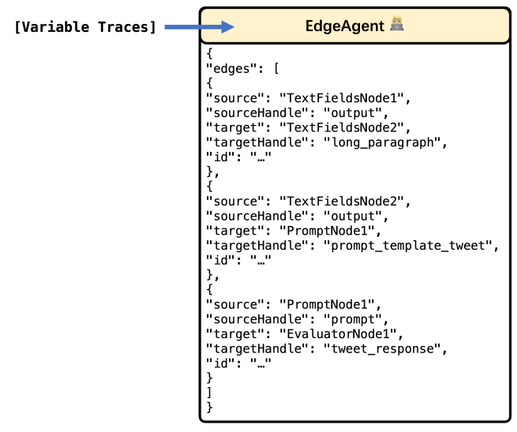

图12：边代理接收来自规划代理的变量追踪数据和生成的节点，
并以ChainForge预期的JSON格式构建边列表。

B.0.5 布局代理。该代理接收来自边代理的输出
其职责是简单配置每个节点的位置
最终工作流程的坐标（从左至右）。

B.1 最终工作流

当所有代理完成各自任务后，工作流程将使用每种节点类型的默认配置模板进行组装。

随后合并为一个JSON文件并呈现在
界面中（参见图2的示例生成，该示例处理了
与此处相同的推文任务）。

B.2 基于层的工作流骨架

为生成完整工作流程，规划代理将用户
需求转化为结构化的、基于层的工作流骨架
此处我们将进一步详述如何在
规划器系统提示（图7）中描述各层。

层定义：

输入层（层*0*）：

- 目的：为工作流程提供不包含任何模板变量的固定输入。
- 结构：由节点（如文本字段节点）组成，定义原始静态输入。这些输入作为基础用于在后续层中生成提示。
- **特性：**
  - 包含用户直接输入的数据（如电子邮件内容、评论或数据集）。
  - 该层中未使用动态变量。

中间层（层*1*、层*2*等）：

- 目的：使用模板处理和转换输入数据变量和提示链。
- 结构：包含生成提示、处理响应或将变量链接在一起的节点。
- **特性：**
  - 动态整合早期层生成的变量。
  - 变量与节点通过顺序逻辑流连接，确保数据转换正确进行。
  - 每个变量均配有唯一标识符以保持清晰性与防止跨层和节点间的冲突。

评估层（最终层）：

- 目的：包含用于评估工作流程生成输出的评估节点。
- 结构：包含设计用于根据预定义标准衡量输出的节点（如LLM评分节点、代码评估节点）。
- **特性：**
  - 根据提供给评估器代理的以下上下文，设计不同数量和类型的评估器：
    - 用户需求：直接评估标准（例如"评价专业度"）。
    - 示例提示：当前工作流程中的示例提示。
    - 潜在响应：评估器可能期望的上下文感知示例LLM评估器可能期望的输出。

C 参与者在自由探索中的想法

为满足读者的好奇心，此处我们列出每位参与者利用助手在自由探索时间中探索的想法。

P1. Compared different LLMs' abilities to translate JavaScript code into Python while ensuring both code validity and functional equivalence.
P2. Evaluated different LLMs' reasoning abilities by testing their responses to various math questions.
P3. Explored generating SPARQL queries from natural language inputs related to music data.
P4. Tested LLMs' responses to questions about local dishes served during Konkan marriages to see if LLMs' response in a very specific cultural topics.
P5. Tested how LLMs respond to sensitive topics like reincarnation through storytelling prompts.
P6. compared how different LLMs generate the context if asked to compare incomes for various doctor specialties, including dentists for 2024.
P7. Examined MBTI personality test results from different LLMs.
P8. Compared prompts to assess business risks based on current news events.
P9. Explored prompts of structuring a conference panel discussion.
P10. Compared prompts to create a productive weekly schedule incorporating tasks and personal goals.
P11. Tested LLMs' responses to potentially harmful queries, exploring how phrasing could bypass content restrictions (jailbreaking).
P12. Developed a workflow to standardize data entries from a range of inconsistent formats into a uniform JSON structure.

P1. Compared different LLMs' abilities to translate JavaScript code into Python while ensuring both code validity and functional equivalence.
P2. Evaluated different LLMs' reasoning abilities by testing their responses to various math questions.
P3. Explored generating SPARQL queries from natural language inputs related to music data.
P4. Tested LLMs' responses to questions about local dishes served during Konkan marriages to see if LLMs' response in a very specific cultural topics.
P5. Tested how LLMs respond to sensitive topics like reincarnation through storytelling prompts.
P6. compared how different LLMs generate the context if asked to compare incomes for various doctor specialties, including dentists for 2024.
P7. Examined MBTI personality test results from different LLMs.
P8. Compared prompts to assess business risks based on current news events.
P9. Explored prompts of structuring a conference panel discussion.
P10. Compared prompts to create a productive weekly schedule incorporating tasks and personal goals.
P11. Tested LLMs' responses to potentially harmful queries, exploring how phrasing could bypass content restrictions (jailbreaking).
P12. Developed a workflow to standardize data entries from a range of inconsistent formats into a uniform JSON structure.