PC-Agent：一种用于 PC 上复杂任务自动化的分层多智能体协作框架

Haowei Liu[1,2*], **Xi Zhang**[3*], **Haiyang Xu**[3†], **Yuyang Wanyan**[1,2], **Junyang Wang**[4], **Ming Yan**[3†], **Ji Zhang**[3], **Chunfeng Yuan**[1,2†], **Changsheng Xu**[1,2], **Weiming Hu**[1,2,5], **Fei Huang**[3]
[1]MAIS, Institute of Automation, Chinese Academy of Sciences, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, China
[3]Alibaba Group [4]Beijing Jiaotong University
[5]School of Information Science and Technology, ShanghaiTech University, China
liuhaowei2019@ia.ac.cn, cfyuan@nlpr.ia.ac.cn
{ shuofeng.xhy, ym119608 } @alibaba-inc.com

Abstract

在基于 MLLM 的 GUI 代理领域，与智能手机相比，PC 场景不仅具有更复杂的交互环境，还涉及更复杂的应用内和应用间的工作流程。为了解决这些问题，我们提出了名为 PC-Agent 的分层代理框架。具体而言，从感知视角来看，我们设计了一个主动感知模块（APM），以克服当前 MLLM 在感知截图内容能力上的不足。从决策角度来看，为了更有效地处理复杂的用户指令和相互依赖的子任务，我们提出了一种分层多代理协作架构，将决策过程分解为指令、子任务和行动三个层次。在这个架构内，设置了三个代理（即管理、进度和决策）分别负责指令分解、进度跟踪和逐步决策。此外，还采用了反思代理，以实现及时的自下而上错误反馈和调整。我们还引入了一个包含 25 个复杂真实指令的新基准 PC-Eval。在 PC-Eval 上的实验证明，我们的 PC-Agent 在任务成功率上比先前最先进的方法提高了绝对值 32 %。代码将公开可用。

1 介绍

最近，多模态大语言模型（MLLM）(**????**) 在各个领域取得了显著进展。基于 MLLM 强大的感知和推理能力，研究人员将其扩展为多模态代理，以协助人类完成各种任务。在这一领域，图形用户界面（GUI）代理受到了广泛关注 (**????**)，因为由代理实现的智能设备（如智能手机、个人电脑）的自动化具有巨大的应用潜力。

与智能手机相比，PC 场景的复杂性体现在两个方面：(1) 更复杂的交互环境。PC 的图形用户界面包含更为密集和多样的交互元素（即图标和小部件），以及不同的文本布局（例如 Word 文档和 VS Code 中的代码），这对屏幕感知构成了重大挑战。例如，如图 1 所示，Word 的顶部功能区包含大量的图标和小部件，但缺乏指示其功能的文本标签。因此，即便是最先进的多模态大模型（例如 Claude-3.5）在感知和定位 PC 屏幕上的图标和文本时也表现出不足的能力，(2) 更复杂的任务序列。与智能手机相比，个人电脑通常用于生产力场景中，这些场景涉及更复杂的应用内和应用间工作流，并需要更长和更复杂的操作步骤。以个人电脑上制定旅行计划（如图 1 ）为例，它可能涉及跨四个应用程序的多个子任务。因此，一方面，冗长的操作序列（即，共 28 个步骤）增加了感知任务进度的难度。另一方面，子任务之间的依赖关系要求代理在做决策时考虑前面子任务的执行结果，进一步增加了决策的难度。如图 1 （b）所示，单个代理（GPT-4o **?** ）的指令级成功率（SR）相较于子任务的成功率从 41.8 % 急剧下降到 8 %，这突显了在个人电脑上完成现实世界指令的挑战。为处理跨应用程序的任务，以往的工作 UFO (**?**) 设计了一个双代理框架，一个用于应用选择，另一个用于具体的控制交互。为解决复杂的个人电脑任务，Agent-S (**?**) 结合在线搜索和本地记忆，以增强经验的计划。然而，这些方法缺乏对屏幕文本的细粒度感知和操作能力，而在生产力场景中（例如 Word 文档编辑），这至关重要。此外，它们通常忽视了子任务之间的复杂依赖关系，因此在处理现实的应用内和应用间复杂任务上能力有限。

在本文中，我们提出了 PC-Agent 框架，以处理 PC 场景中的复杂交互环境和复杂任务，其中包含三个核心设计：(1) 主动感知模块。为了增强代理的细粒度感知和操作能力，我们提出了一个主动感知模块（APM）。对于交互元素，我们使用可访问性树来提取它们的位置和含义。对于文本，我们采用一个由多模态大模型驱动的意图理解代理来提取目标文本，然后通过 OCR 获得精确的位置。(2) 分层多代理协作。为了提高处理复杂指令的能力，我们采用分而治之的方法，并提出了一个分层多代理协作架构。具体来说，我们将决策过程分为三个层次：指令、子任务和动作。在指令层面，管理代理（MA）将用户指令分解成参数化子任务，其操作步骤显著减少，决策难度降低。MA

[*] The first two authors contributed equally to this work.
[†] Corresponding authors.

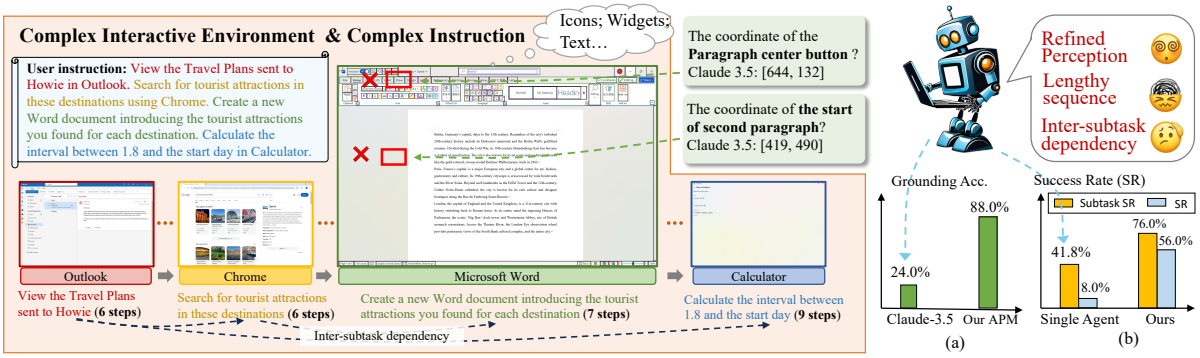

Figure 1: PC 场景复杂性的说明：(1) 复杂的交互环境，具有密集且多样化的元素。(2) 包含软件内部和软件之间工作流的长而复杂的任务序列。

还负责管理子任务之间的通信，以处理它们之间的复杂依赖关系。在子任务层面，进度代理（PA）跟踪并总结操作历史，以实现准确的进度意识。在动作层面，决策代理（DA）通过结合 APM 的感知信息和 PA 的进度信息逐步做出决策，并与 PC 环境交互以完成分解的子任务。基于反思的动态决策。在上述架构基础上，我们还引入了一种基于反思的动态决策机制，用于检测执行结果中的错误，并进行及时反馈和调整。在动作层面设置了额外的反思代理（RA），观察 DA 决策前后的屏幕变化，评估该步骤的正确性，并将反馈传递给 DA 和 PA。图 2 展示了整个过程。结合分层多代理协作架构和基于反思的动态决策，我们的 PC-Agent 框架能够自顶向下分解复杂的用户指令，并在执行过程中自底向上提供精确反馈。因此，四个代理合作以缓解 PC 上交互环境和复杂工作流任务的难度。

为了更好地评估代理在复杂任务中的能力，我们为 PC 生产力环境提出了一个新的基准测试 PC-Eval。PC-Eval 由八个流行的应用程序和 25 个复杂的用户指令组成，每个指令包含几个相互依赖的子任务。它通过强调复杂的工作流程和长时间决策，提供了一个具有挑战性且现实的基准。将我们的 PC-Agent 与基于先进 MLLM 的单一代理和现有开源 PC 代理在 PC-Eval 上的表现进行比较，我们发现 PC-Agent 在指令和子任务级别的成功率上都有显著提高，证明了所提出框架的有效性。

我们的贡献可以总结如下：

(1) 我们提出了一个 PC-Agent 框架，以克服现有方法在处理复杂交互环境和 PC 场景中的复杂任务时的限制。设计了一个主动感知模块（APM），使 PC-Agent 具有更精细的感知和操作能力。

(2) 为了应对复杂的 PC 任务，我们提出了一种分层多智能体合作架构，将决策过程分解为三个层次（即指令-子任务-动作），并引入了一种基于反思的动态决策机制，以便及时进行错误反馈和调整。

(3) 我们创建了一个包含 8 个常用 PC 应用程序的 PC-Eval 基准，以更好地评估代理处理复杂用户指令的能力。实验结果表明，所提出的 PC-Agent 在完成复杂 PC 任务方面大大优于以往的方法。

2 PC-代理

给定一个 GUI 环境和一个用户指令 $\mathcal{I}$，GUI 代理（记为 ρ）获取关于该环境的观测 $\mathcal{O}$（例如，截图）。基于内部推理和规划，它对当前步骤的动作 $\mathcal{A}$ 做出决策，该动作与 GUI 环境交互，并改变环境的状态。这个过程通常逐步进行。该过程可以形式化为：

$$\mathcal{A}_i = \rho(\mathcal{I}, \mathcal{O}_i, \mathcal{H}_{i-1}), \tag{1}$$

，其中 $\mathcal{A}_i$ 和 $\mathcal{O}_i$ 分别表示第 i 步的动作和观察，$\mathcal{H}_{i-1}$ 是直到第 $(i-1)$ 步的操作历史。PC 中的复杂交互环境和任务序列增加了 $\mathcal{I}$、$\mathcal{O}$ 和 $\mathcal{H}$ 的复杂性，因此需要设计一个适合复杂 PC 场景的代理框架。

首先，为了精确感知和操作交互元素和文本，我们提出了一种主动感知模块（APM）。

交互元素感知。我们首先使用 pywinauto API 提取屏幕界面的可访问性（A11y）树，过滤并解析交互元素的坐标和描述。然后，我们以一种 SoM (?) 方式在截图中标注这些元素的边界框，以帮助 MLLM 理解元素的位置和含义。

文本感知。文本信息无法通过 A11y 树获取，用户指令通常会含糊地引用文本，使得直接获取目标文本的内容和位置变得困难。例如，加粗本文档的最后两段。为了解决这个问题，我们建议利用主动感知来获取目标文本的内容和位置。如图 3 所示，对于涉及精细文本操作的任务（例如，选择或编辑），决策代理输出选择（目标文本）动作。随后，APM 使用一个由 MLLM 驱动的意图理解代理来确定目标文本的开始和结束范围，接着使用 OCR 工具精确

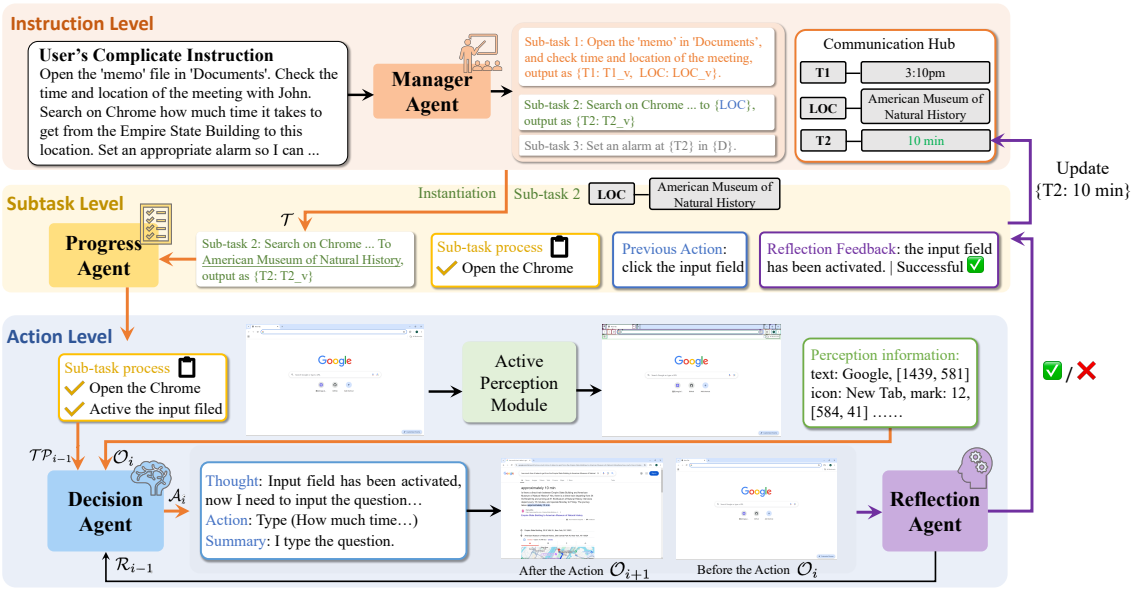

Figure 2: 提出的 PC-Agent 的概述，它将决策过程分解为三个层次。橙色线表示自上而下的决策分解，紫色线表示自下而上的反思过程。

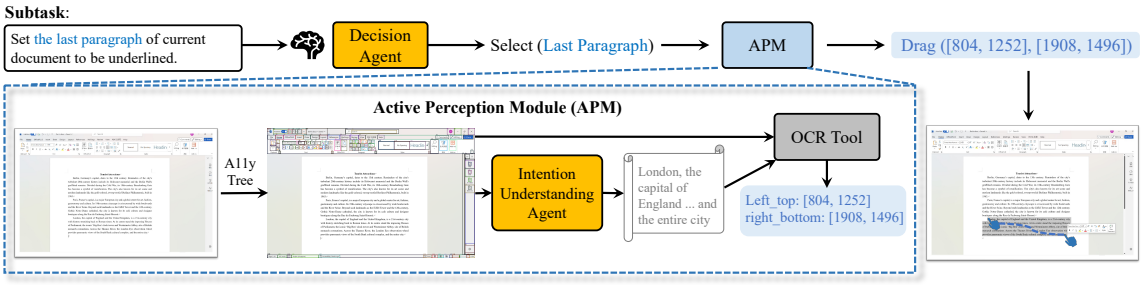

Figure 3: 主动感知模块的示意图。对于交互元素，采用 A11y 树获取边界框和功能信息。对于文本，使用意图理解代理和 OCR 工具进行精确选择或编辑。

定位目标文本，以便进行后续的详细操作，如拖动。更详细的案例在附录的图 6 中展示。

2.1 层次化多智能体协作

PC 场景通常涉及应用程序内部和应用程序之间的工作流程，增加了用户指令的复杂性。为了解决这个问题，我们采用分而治之的方法，将决策过程分解为三个层次：指令、子任务和动作。如图 2 所示，基于这种自上而下的层次化分解，我们设计了一个多智能体协作架构：(1) 指令层次：设置一个管理代理 (MA) 进行高层次任务管理，包括将指令分解为子任务、子任务之间的通信以及整体进度。(2) 子任务层次：设立一个进度代理 (PA) 来管理子任务的进展。(3) 动作层次：指定一个决策代理 (DA) 来完成子任务。针对具体的子任务，DA 根据环境感知和由 PA 提供的操作历史，逐步进行决策。

通过这种层次化的多智能体协作，复杂的用户指令被分解为若干相互依赖的子任务。管理者、进度和决策智能体的协作努力有效地降低了整体决策难度，提高了成功率。

在我们的分层多智能体协作架构中，受大模型驱动的经理代理（MA）在高层任务管理中起着至关重要的作用：

(1) 指令分解。如图 2 所示，给定一个复杂的用户指令，MA 首先将其分解为一系列参数化的子任务。每个子任务一旦被实例化，就可以由进程代理和决策代理独立执行，从而有效地降低单个任务的复杂性。

(2) 子任务之间的通信。分解后的子任务通常具有复杂的相互依赖关系。具体来说，有四种类型的子任务：(a) 子任务的执行结果可用于实例化后续子任务（例如，图 2 中的子任务 1）；(b) 子任务依赖于前面子任务的执行结果进行实例化（例如，图 2 中的子任务 3）；(c) 子任务既依赖于前面子任务进行实例化，又为后续子任务产生执行结果（例如，图 2 中的子任务 2）；(d) 子任务独立于其他子任务（例如，在时钟应用中设置上午 10 点的闹钟）。如图 2 所示，在整个过程中，管理代理负责管理子任务之间的通信和复杂的参数传递关系。它维护一个通信中心，将成功执行的子任务的输出更新到该中心，并利用该中心来实例化后续的子

任务。

在管理代理完成指令分解和必要的子任务间通信以实例化参数化子任务后，当前独立可执行的子任务被交给进度代理（PA）。PA 也是由 LLM 驱动的，负责基于决策代理的决策和反思代理的反馈（将在 2.2 节介绍）来跟踪和总结子任务的进展。一旦当前子任务完成，PA 会将输出结果反馈给 MA。

在 MA 和 DA 之间建立独立的 PA 的目的有两个：（1）它通过分而治之实现更精确的进度跟踪。PA 单独跟踪每个子任务的进展。这避免了总结整个指令级历史，这可能会冗长且繁琐。（2）它通过为决策代理提供对操作历史及哪些子任务部分尚未完成的清晰理解，来促进决策。这避免了在决策过程中受到冗长历史信息的干扰。

具体来说，PA 在第 i 步的输入包括四个部分：(1) MA 分配的当前子任务 $\mathcal{T}$；(2) 以前的任务进度 $\mathcal{TP}_{i-1}$；(3) 第 i 步的 DA 输出的动作 $\mathcal{A}_i$；以及 (4) 执行第 i 步的动作后的反思 $\mathcal{R}_i$。基于这些信息，PA 输出更新后的进度 $\mathcal{TP}_i$。上述过程可以形式化为：

$$\mathcal{TP}_i = PA(\mathcal{T}, \mathcal{TP}_{i-1}, \mathcal{A}_i, \mathcal{R}_i). \tag{2}$$

在多层多节点学习模型（MLLM）的驱动下，决策代理（DA）是整个 PC-Agent 框架中的核心代理，负责生成行动决策并直接与环境交互。给定一个子任务 $\mathcal{T}$，在每一个步骤中，DA 首先使用感知模块获取当前环境的观测结果 $\mathcal{O}_i$。接着，将其与上一步中 PA 输出的进度信息 $\mathcal{TP}_{i-1}$，以及 RA 输出的反思信息 $\mathcal{R}_{i-1}$ 相结合，以生成当前步骤的决策 $\mathcal{A}_i$。这个过程可以形式化为：

$$\mathcal{A}_i = DA(\mathcal{T}, \mathcal{O}_i, \mathcal{TP}_{i-1}, \mathcal{R}_{i-1}). \tag{3}$$

在这里，决策是以一种链式思考 (**?**) 方式生成的。首先会生成当前步骤的内心独白，然后是相应的行动决策。这种方法不仅帮助 MLLM 做出更好的决策，还帮助 RA 判断执行结果是否符合预期。

在获得当前步骤的决策后，我们将决策信息转换为特定的动作类型及其对应的参数，然后使用 pyautogui API 执行相应的键盘和鼠标操作。为了简化操作并使决策易于解析，我们定义了一个受限的动作空间，其中包括点击、双击、输入、选择、拖动、滚动、快捷键和停止（详见附录 **??**）。这种受限的动作空间确保了 DA 能够有效地生成和执行决策，从而实现高效和准确的任务完成。

2.2 基于反思的动态决策

由于幻觉和有限的推理能力等因素，即使是最先进的 MLLMs（例如，GPT-4o **?**，claude-3.5 **?**）也难以避免在感知和决策中出错。当任务需要长的操作序列时，这个问题更加突出，因为在任何步骤中的一个错误都可能导致整个任务的失败。

为了检测执行结果中的潜在错误并提供及时的反馈和调整，我们设计了一种基于反思的动态决策机制。该动态决策机制建立在第 2.1 节介绍的分层架构之上，以反思代理为核心，以自下而上的方式运行。

在分层架构的动作层中，我们设置了一个与决策代理（DA）并行的反射代理（RA）。在 DA 做出决策并执行相应动作之后，RA 观察动作前后系统状态的变化，以确定该步骤的结果是否符合预期。这个过程可以形式化为：

$$\mathcal{R}_i = RA(\mathcal{T}, \mathcal{A}_i, \mathcal{O}_{i-1}, \mathcal{O}_i). \tag{4}$$

根据执行结果，RA 会做出三种判断：（1）动作的执行导致截图发生了不符合预期的变化。这可能是由于 DA 的决策中动作类型或位置参数不正确，需重新规划以纠正错误。（2）执行动作后，截图上未产生有效响应。这可能是因为动作执行在没有互动元素的位置上，或者元素（如输入框）尚未激活，需要调整动作执行位置。（3）动作执行产生了正确的结果，使得 DA 可以基于此继续进行下一步决策。

在前两种情境中，RA 的输出将反馈给 DA，使得 DA 能够基于反思信息做出决策，以纠正错误或避免重复无效的行动。RA 的反思信息还将反馈给进度代理（PA），使 PA 能够检测错误并实现更准确的进度跟踪。

3 实验

3.1 PC-Eval

尽管现有的大规模真实计算机环境基准测试（例如，OSWorld **?** 和 WindowsAgentArena **?**）包含了许多基本任务，但这些任务可能并未与实际的工作流程要求相一致，比如打开画图并画一个红色圆圈。为了更好地评估代理在复杂 PC 任务上的能力，我们提出了一个新的基准 PC-Eval，该基准由 25 个复杂指令（总共 79 个子任务）组成，涉及 8 个常用的 PC 应用程序（即 Chrome、Microsoft Word、Microsoft Excel、Notepad、Clock、Calculator、Outlook 和 File Explorer）。每个指令包含几个相互依赖的子任务，并强调精细操作、实际工作流程和长期决策。三位注释者创建并检查了这些指令，以确保它们是真实且具有挑战性的。表 1 显示

Table 1: PC-Eval 中的复杂指令示例。

Applications	Instruction	Steps
File Explorer Notepad, Clock Calculator	In the Notepad app, open the 'travel_plan' file in 'Documents', and check the time and location of the travel plans. Add the travel destination to the World Clock list on the Clock app. Calculate the interval between February 18 and the start time of the travel on the Calculator.	20
Chrome Excel	Search on Chrome for the total population of China, the United States, and India in 2024 respectively. Create a new spreadsheet in Excel, write the three countries' names in column A in descending order of population, and the corresponding populations in column B.	23
File Explorer Word	Open the 'test_doc1' file in 'Documents' in File Explorer, set the title to be bold, and set the line spacing of the first two paragraphs to 1.5x in Word.	8

Table 2: 在 PC-Eval 基准上的动态评估结果。

Model	Type	Subtask SR (%) ↑	Success Rate (%) ↑
Gemini-2.0	Single-Agent	35.4 %	0.0 %
Claude-3.5		15.2 %	0.0 %
Qwen2.5-VL		46.8 %	12.0 %
GPT-4o		41.8 %	8.0 %
UFO (?)	Multi-Agent	43.0 %	12.0 %
Agent-S (?)		55.7 %	24.0 %
PC-Agent (Ours)		76.0 %	56.0 %

了三个示例指令，完整列表见附录 A.3 。由于不同的子任务对应不同的页面和成功标准，为每个子任务创建独立的自动评估脚本成本将非常高。因此，我们在本研究中使用人工评估，并采用以下两个评估指标：
(1) 成功率 (SR)：成功率指标指的是代理成功完成指令的比例。(2) 子任务成功率 (SSR)：为了综合评估代理的能力，我们对 PC-Eval 指令的子任务进行了标注，并计算了代理完成的子任务的成功率。

3.2 结果

实验设置。在实验中，除非另有说明，我们使用 GPT-4o 作为 PC-Agent 框架中经理、进度、决策和反思代理的基础模型。我们在 APM 中使用 OpenOCR 工具进行 OCR。我们将我们的 PC-Agent 与多种单代理和多代理方法进行比较，包括先进的 MLLMs，如 GPT-4o (?) 、Gemini-2.0 (?) 、Claude-3.5 (?) 、Qwen2.5-VL 72B (?) ，以及以前的开源 PC 代理方法，如 UFO (?) 和 Agent-S (?) 。为了尽可能公平地比较，我们通过提示设置 MLLMs 相同的动作空间，使它们能够作为单一决策代理运行。至于多代理方法 UFO 和 Agent-S，我们也采用 GPT-4o 作为它们的基础模型。
单一代理的结果。表 2 展示了 PC-Agent 与其他方法在 PC-Eval 上的性能比较。可以看到，那些基于 MLLM 的单一代理在所有指令上几乎全部失败。即使是表现最好的 Qwen2.5-VL 也仅仅达到 12 % 的成功率。这个结果表明，单凭当前 MLLM 的能力来完成 PC 上复杂的用户指令是极具挑战性的。同时，这些模型的成功率显著低于子任务的成功率。这表明，由于操作序列的冗长和子任务之间的复杂依赖关系，完成整条指令比完成个别子任务要困难得多。
多智能体方法的结果。UFO 和 Agent-S 是为 PC 场景量身定制的两个智能体框架。然而，在 PC-Eval 上，UFO 仅比使用 GPT-4o 的单一智能体取得了略微的优势。虽然 Agent-S 在 SSR 方面较单一智能体有所提升，但其指令级的 SR 仍然较低。详细分析揭示了它们在感知和决策方面的问题：

（1）现有方法在细粒度感知和操作能力上有限。例如，在如图 4 所示的 Excel 场景中，UFO 可能会将多条信息输入到同一个单元格中。在如图 6 所示的 Word 场景中，UFO 和 Agent-S 都无法执行编辑操作（例如，“给最后一段加下划线”）。（2）现有方法在处理复杂指令中子任务之间的依赖关系时不够充分，特别是在后续子任务的执行依赖于之前任务结果的场景中。例如，在指令“……并写下内容的翻译”中，Agent-S 会直接写下“内容的翻译”这段文字，而不是之前获得的翻译内容。

相比之下，我们提出的 APM 使 PC-Agent 具备更精细的操作能力。此外，通过分层的多代

https://github.com/Topdu/OpenOCR

Table 3: 关于 APM 模块、管理代理和反射代理的消融研究结果。

Ablation study			Subtask Success Rate	Success Rate
APM	Manager Agent	Reflection Agent		
	✓	✓	58.2 %	20.0 %
✓		✓	50.6 %	12.0 %
✓	✓		48.1 %	12.0 %
✓	✓	✓	76.0 %	56.0 %

理协作，PC-Agent 实现了有效的指令分解、子任务间通信、进度管理和错误反映，这显著提高了复杂任务的性能。结果显示，我们的 PC-Agent 大大优于所有先前的方法，在 SR 指标上分别超过了 UFO 和 Agent-S，分别提升了 44 % 和 32 %。

3.3 消融研究

表 3 展示了对 PC-Agent 框架的不同组件进行消融研究的结果，我们可以从中得出结论：
(1) 主动感知模块对性能有显著影响。比较第一行和第四行，可以看出在移除 APM 后，SSR 下降了近 20 % ，而 SR 大幅下降了超过 30 % 。一方面，没有 APM 时，决策代理无法掌握交互元素的意义，因此会犯更多错误。另一方面，PC-Agent 丧失了精确感知和操纵所指文本的能力。结果是，指令完成率显著下降。
(2) 管理代理在复杂工作流场景中有效地提升了 PC-Agent 的能力。对比第二行和第四行可以看出，移除 MA 将导致 SR 显著下降到 12 % 。这是因为没有 MA，复杂指令会被视为单一任务交给 PA 和 DA 执行。冗长的操作序列和子任务之间的复杂依赖性对进度跟踪构成了巨大的挑战，并干扰了 DA 的决策。
(3) 基于反思的动态决策机制可以帮助模型从错误中恢复。通过比较第三行和第四行，可以看出去除 RA 会导致性能显著下降（即在 SSR 中降低 27.9 %，在 SR 中降低 44.0 %）。这是因为在执行复杂指令的过程中，感知和决策上的错误是不可避免的。去除 RA 会导致模型缺乏对错误的意识和及时纠正，容易陷入无意义的重复或错误的步骤中。

我们还对基础模型进行了消融实验，详细信息请参见附录 A.1 。

3.4 案例研究

图 4 展示了我们 PC-Agent 框架的完整操作过程。对于一个复杂的用户指令，管理代理首先将其分解为四个子任务。对于前三个子任务，当每一个成功执行时，相应的搜索结果会在通信中心中更新。然后，管理代理使用通信中心来实例化第四个子任务，从而降低长时间决策过程的难度。此外，Excel 中精确的点击和输入操作展示了我们所提出的 APM 在感知复杂屏幕元素方面的有效性。我们还提供了一个基于反思的动态决策案例研究。详情请参见附录 A.2 。

4 相关工作

最近在多模态大模型（MLLMs）(???) 方面的进展激发了将这些模型扩展到各个领域的智能代理的研究。其中，对于智能设备上任务自动化的图形用户接口（GUI）代理存在显著关注。目前，该领域的研究更多地集中在移动 (???) 和网络 (??) 场景。在 PC 场景下，Cradle (?) 着重于利用 MLLM 的推理能力实现 AAA 游戏中的操作，而 PC Agent (?) 旨在使代理能够创建和修改 PowerPoint 演示文稿。尽管取得了显著进展，但它们的通用性仍然相对有限。为了解决跨应用任务，UFO (?) 设计了一个双代理框架，其中一个代理负责应用选择，另一个代理负责具体的控制互动。为了将 PC 任务知识注入决策过程，Agent-S (?) 结合在线搜索和本地记忆进行经验增强的规划。与以前的方法相比，我们的 PC-Agent 专注于复杂的 PC 任务。我们通过设计的 APM 实现了更细致的感知和操作（例如，编辑 Word 文档）。而所提出的分层框架实现了复杂指令的分而治之流程，有效解决了子任务间的依赖性，大幅提升了复杂任务的性能。

在这项工作中，我们提出了一个 PC-Agent 框架，以处理 PC 场景中的复杂互动环境和任务。我们设计了一个主动感知模块，以提高感知和操作能力。我们提出了一种分层多智能体协作架构，将决策过程分解为三个级别，并采用基于反思的动态决策，以提供及时的错误反馈和调整。我们创建了一个现实且复杂的用户指令 PC-Eval 基准。实验结果验证了 PC-Agent 在复杂 PC 任务上比以往的方法具有更优的性能。

在本文中，我们探索了多种 MLLMs 作为基础模型。目前，表现最好的模型依然是闭源的

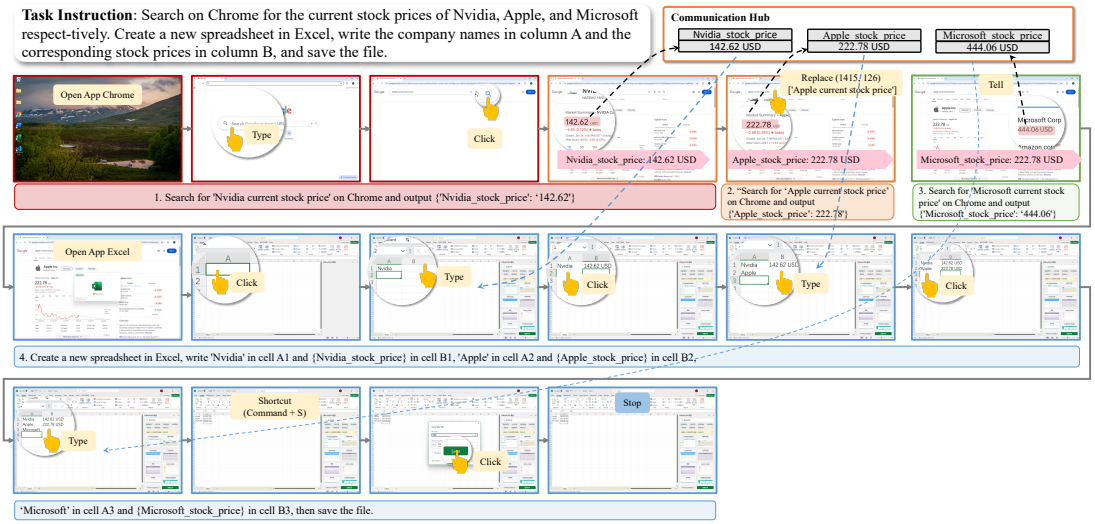

Figure 4: 多次搜索信息并相应地建立一个 Excel 表格的案例。

GPT-4o。然而，在通过调用闭源模型完成复杂任务的效率方面，仍有很大的提升空间。与闭源模型相关的隐私和安全问题也值得关注。此外，我们在这项工作中的重点主要集中在生产力场景。未来的工作中，我们将探索扩展到更多的场景，如社交互动和娱乐等。

A 附录

A.1 基础模型的消融研究

表 4 比较了不同 MLLM 的性能。这里我们介绍了两个除了 SR 和 SSR 之外的指标，用以比较使用不同 MLLM 作为基础模型的结果：

- 恢复率：它衡量发生恢复的指令比例。恢复行为表明代理检测到错误并通过反思进行纠正（无论指令是否最终完成）。

- 经理 SR: 此指标评估经理代理正确分解用户指令的能力。

从表格中，我们可以观察到由 GPT-4o 驱动的 PC-Agent 的 SSR 和 SR 显著优于使用 Gemini-2.0、Claude-3.5 和 Qwen2.5-VL 的结果。而在恢复率方面，GPT-4o 领先 Gemini-2.0 40 %。这可能得益于 GPT-4o 更好的感知和推理能力。此外，值得注意的是，相比使用 Qwen2.5-VL 的单一代理，使用 Qwen2.5-VL 的 PC-Agent 的 SSR 和 SR 实际上有所下降。详细分析显示，这归因于 Qwen2.5-VL 在遵循输出动作格式的文本能力有限，以及判断任务是否完成的能力不理想。而后一问题在指令被分解成子任务后变得更为严重。总之，表格 4 中的结果强调了 MLLMs 的能力是该框架有效性的基础。

A.2 更多案例研究

图 5 展示了 PC-Agent 框架中的一个例子，其中提出的反思机制防止了重复的无效操作。如图所示，在决策代理（DA）点击 Chrome 浏览器的前进按钮而未产生有效响应后，反思代理（RA）检测到了此错误并将其反馈给 DA。基于这一反馈，DA 在下一步重新考虑并执行了正确的操作（即，使用快捷键 Command + T 打开新标签页）。

我们定义动作空间如下：

A.3 PC-Eval 中的指令

我们展示了 PC-Eval 的完整指令列表如下：

- 在记事本应用程序中，打开“文档”中的‘memo‘文件，查看早上第二个事件。在时钟应用程序中设置一个在该事件前 1 小时的闹钟。

- 在记事本应用中，打开 ‘Documents‘中的‘memo‘文件，查看与 John 会面的地点。在 Chrome 上搜索从帝国大厦到这个地点需要多长时间。

- 在记事本应用中，打开位于“Documents”中的“memo”文件，检查与 John 会议的时间和地点。在 Chrome 上搜索从帝国大厦到此地点需要多少时间，并在时钟应用中设定一个合适的闹钟，以便我能及时离开帝国大厦，准时到达会议地点。

- 在记事本应用程序中，打开“Documents”中的“travel_plan”文件，并查看旅行目的地。使用 Chrome 搜索该目的地的交通是靠左还是靠右行驶。

Table 4: 在 PC-Eval 上，PC-Agent 与不同基础模型的性能结果。

Model	Subtask SR (%) ↑	Success Rate (%) ↑	Recovery Rate (%) ↑	Manager SR (%) ↑
Gemini-2.0	55.7 %	28.0 %	24.0 %	84.0 %
Claude-3.5	63.3 %	40.0 %	48.0 %	88.0 %
Qwen2.5-VL	32.9 %	12.0 %	40.0 %	88.0 %
GPT-4o	76.0 %	56.0 %	64.0 %	96.0 %

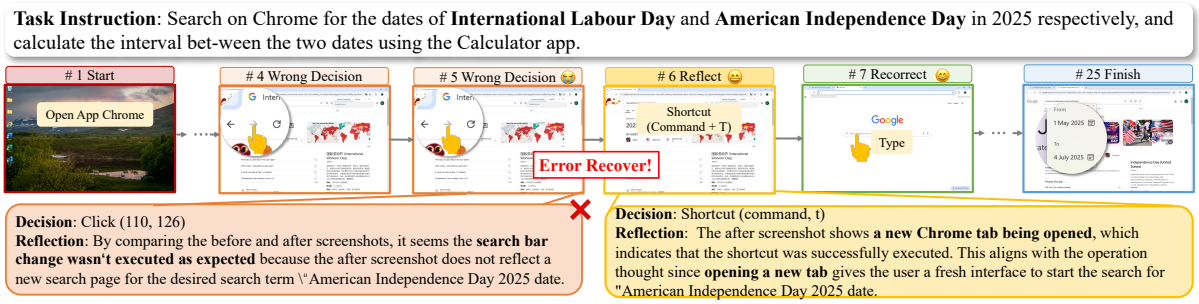

Figure 5: 在 Chrome 中执行多次连续搜索时的一种反思情况。

- 在 Chrome 上分别搜索 2025 年国际劳动节和美国独立日的日期，并使用计算器应用计算这两个日期之间的间隔。
- 在记事本应用中打开 “Documents” 中的 “travel_plan2” 文件，查看旅行计划中的三个候选目的地。在 Chrome 中搜索从北京到每个目的地的航班时间，并告诉我哪个候选目的地的航班时间最短。
- 在 Chrome 上分别搜索 Nvidia、Apple 和 Microsoft 当前的股票价格。在 Excel 中创建一个新的电子表格，在 A 列写上公司名称，并在 B 列写入相应的股票价格。
- 在 Chrome 上分别搜索 2024 年中国、美国和印度的总人口。在 Excel 中创建一个新的电子表格，将三个国家的名称按人口数量从高到低的顺序写在 A 列，并在 B 列填写相应的人口数字。
- 在 Word 中创建一个新文档。分别写下两个段落介绍阿里巴巴和 OpenAI。将文档保存为 ‘TechCcompanies’。
- 在 Chrome 上搜索论文《Attention is all you need》，下载这篇论文并记录其摘要。创建一个新的 Word 文档，写下这篇论文的摘要，并将其保存为《Transformer》。
- 在 Chrome 上搜索 ‘星际穿越 ‘ 和 ‘十二怒汉 ‘ 在 imdb.com 上的评分。在文件资源管理器中的 ‘Documents‘ 文件夹中打开 ‘movie_rate‘ Excel 文件，并填写相应的电影评分。
- 在文件资源管理器中的 “Documents” 中打开 “test_doc1” 文件，将标题设置为加粗，并在 Word 中将前两段的行距设置为 1.5 倍行距。
- 在文件资源管理器中的 “文档” 里打开 “test_doc2” 文件，将标题设置为居中，并将 Word 中的最后一段设置为加下划线。
- 在文件资源管理器中的 “文档” 中打开 “test_doc3” 文件，并写下正文下方内容的翻译。
- 在 Chrome 中访问 https://arxiv.org/，搜索与 “多模态代理” 相关的论文，并下载第一篇论文。
- 在 Outlook 中阅读发送给 Howie 的"Travel" 邮件，记录行程的出发地、目的地和开始日期。在 Chrome 上通过 booking.com 搜索单程航班。
- 在 Chrome 中搜索《这个杀手不太冷》、《肖申克的救赎》和《2001 太空漫游》的 IMDb 评分。使用记事本将它们记录在一个新的.txt 文件中，并按从高到低的顺序进行排序。
- 在 Outlook 中检查发送给 Howie 的邮件 ‘Code‘，下载附件 ‘homework.py‘并在 Vi-

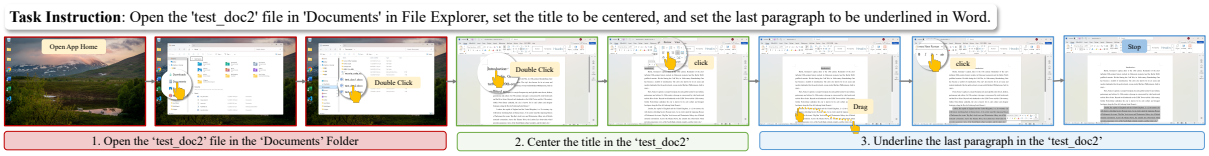

Figure 6: 在 Word 应用程序中细化文本编辑操作的一个案例。

sual Studio Code 中打开它。修复这个 Python 代码中的错误。

- 在 Visual Studio Code 中创建一个新的 Python 文件，编写一个函数，该函数接受一个列表作为输入并输出列表中第 k 大的数字。通过 Outlook 将此代码文件发送给 Howie。
- 分别在 Chrome 中搜索东京和京都的旅游景点，并在新的 Word 文档中记录信息。
- 打开位于文件资源管理器中的“文档”中的“test_doc3”文件，注意其中文内容，创建一个新的 Word 文档，并写下 test_doc3 中文内容的英文翻译。
- 在文件资源管理器中打开位于“Documents”内的“test_doc1”文件，将标题的字体大小提高一个级别。
- 在记事本应用程序中，打开“文档”中的‘travel_plan‘文件，检查旅行计划的时间和地点。将旅行目的地添加到时钟应用程序的世界时钟列表中。使用计算器计算 2 月 18 日与旅行开始时间之间的间隔。
- 在 Chrome 上分别搜索 2024 年中国、美国和印度的总人口。在 Excel 中创建一个新电子表格，在 A 列中按人口数量降序排列写下这三个国家的名称，并在 B 列中写下相应的人口数量。
- 在文件资源管理器中的“Documents”中打开“test_doc1”文件，将标题设置为加粗，并在 Word 中将前两段的行距设置为 1.5 倍。
- 在 Chrome 浏览器中比较亚马逊、沃尔玛和百思买的新任天堂 Switch 游戏机的价格，并在记事本上记录价格最低的网站和价格。
- 阅读 Outlook 中的邮件“Travel”，记录旅行的出发地、目的地和日期。在 Chrome 上使用 booking.com 搜索往返航班。

A.4 我们的 GUI 基础数据集

在 Booking.com 的网页上：
在 Excel 页面上：

- 点击选择 A3
- 点击选择 E5
- 点击选择顶部对齐
- 点击选择底部对齐
- 点击选择左对齐
- 点击选择右对齐
- 点击保存
- 点击更改文件名
- 点击更改保存位置

在文件资源管理器页面上：
在 Outlook 页面上：

- 点击查看收件箱
- 点击查看垃圾邮件/垃圾邮箱
- 点击查看已发送邮件
- 点击查看发送给 Howie 的旅游邮件
- 点击查看发送给 Howie 的代码邮件
- 点击搜索
- 点击创建新邮件
- 点击标记为已读

在 Chrome 页面上：
在 Word 页面上：

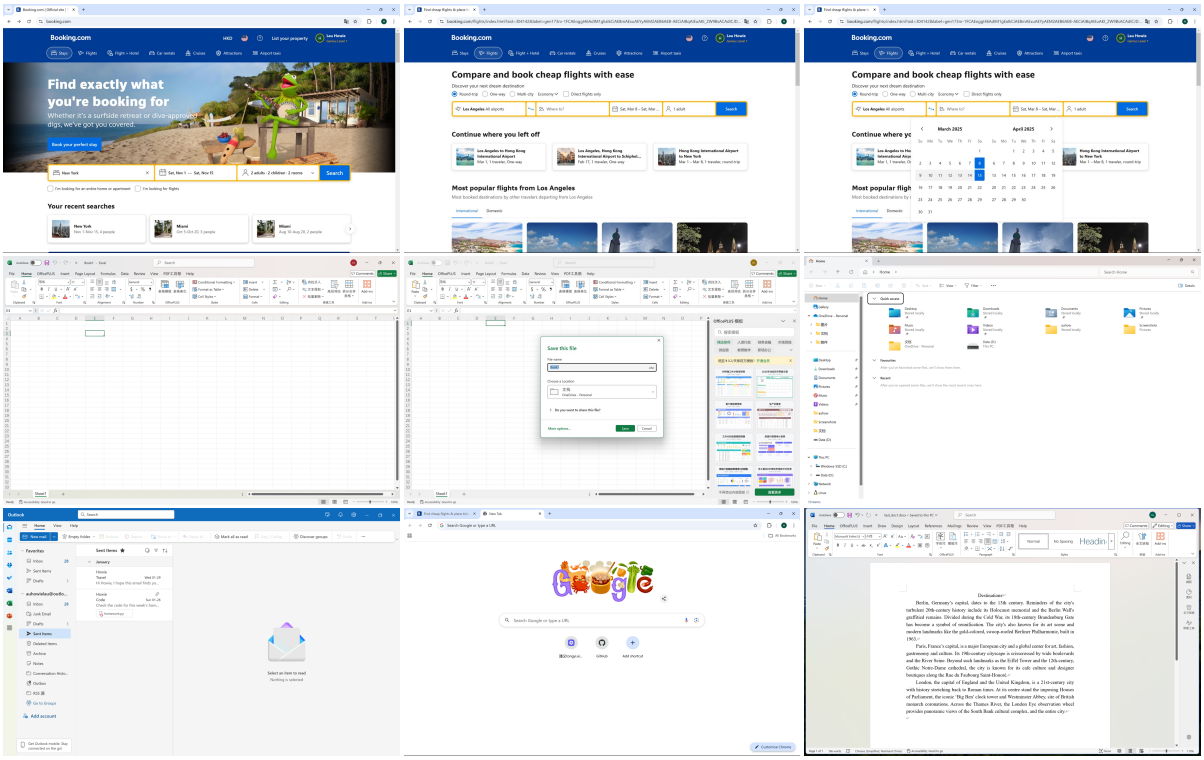

Figure 7: 我们为 PC 场景中常用应用程序构建的 GUI 基础数据集中示例截图。