

PG-Agent: An Agent Powered by Page Graph

Weizhi Chen*

Zhejiang Key Lab of Accessible
Perception & Intelligent Systems,
Zhejiang University
Hangzhou, China
chenweizhi@zju.edu.cn

Ziwei Wang*

Zhejiang Key Lab of Accessible
Perception & Intelligent Systems,
Zhejiang University
Hangzhou, China
wangziwei98@zju.edu.cn

Leyang Yang

Zhejiang Key Lab of Accessible
Perception & Intelligent Systems,
Zhejiang University
Hangzhou, China
yangleyang@zju.edu.cn

Sheng Zhou†

Zhejiang Key Lab of Accessible
Perception & Intelligent Systems,
Zhejiang University
Hangzhou, China
zhousheng_zju@zju.edu.cn

Xiaoxuan Tang

Ant Group
Beijing, China
leahxx1226@outlook.com

Jiajun Bu

Zhejiang Key Lab of Accessible
Perception & Intelligent Systems,
Zhejiang University
Hangzhou, China
bjj@zju.edu.cn

Yong Li†

Ant Group
Hangzhou, China
liyong.liy@antgroup.com

Wei Jiang

Ant Group
Beijing, China
jonny.jw@antgroup.com

Abstract

Graphical User Interface (GUI) agents possess significant commercial and social value, and GUI agents powered by advanced multimodal large language models (MLLMs) have demonstrated remarkable potential. Currently, existing GUI agents usually utilize sequential episodes of multi-step operations across pages as the prior GUI knowledge, which fails to capture the complex transition relationship between pages, making it challenging for the agents to deeply perceive the GUI environment and generalize to new scenarios. Therefore, we design an automated pipeline to transform the sequential episodes into page graphs, which explicitly model the graph structure of the pages that are naturally connected by actions. To fully utilize the page graphs, we further introduce Retrieval-Augmented Generation (RAG) technology to effectively retrieve reliable perception guidelines of GUI from them, and a tailored multi-agent framework PG-Agent with task decomposition strategy is proposed to be injected with the guidelines so that it can generalize to unseen scenarios. Extensive experiments on various benchmarks demonstrate the effectiveness of PG-Agent, even with limited episodes for page graph construction. Our codes will be publicly available at <https://github.com/chenwz-123/PG-Agent>.

*Both authors contributed equally to this research.

†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '25, Dublin, Ireland

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-2035-2/2025/10
<https://doi.org/10.1145/3746027.3755189>

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI); Interaction design.**

Keywords

GUI Agent; Retrieval-Augmented Generation; Multimodal Large Language Model

ACM Reference Format:

Weizhi Chen, Ziwei Wang, Leyang Yang, Sheng Zhou, Xiaoxuan Tang, Jiajun Bu, Yong Li, and Wei Jiang. 2025. PG-Agent: An Agent Powered by Page Graph. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25)*, October 27–31, 2025, Dublin, Ireland. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3746027.3755189>

1 Introduction

The Graphical User Interface (GUI) has become crucial for humans in interacting with mobile devices and websites. Recently, there has been a notable increase in interest in GUI agents that can autonomously perform tasks by interacting with the user interface [27]. It is emerging as a significant topic of study in disciplines such as software engineering and human-computer interaction [19, 31, 36], among several others. Early works employed parsing tools to transform the pages into HTML presentations, utilizing large language models (LLMs) to analyze the page layouts to make decisions [14, 39]. With the rapid development of multimodal large language models (MLLMs) [1, 2, 6, 28, 42], MLLM-based GUI agents become the mainstream architecture, which are able to analyze the screen and generate actions end-to-end.

The GUI agents are conducted in a structured enclosed space where different pages are naturally interconnected through operations like clicking. Therefore, it is essential for the agent to possess the awareness of possible actions and their consequent pages. However, existing works [8, 22, 32] have collected abundant knowledge

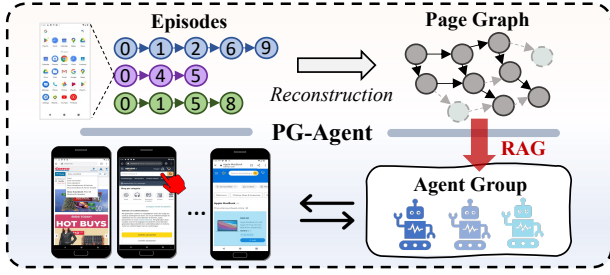


Figure 1: Illustration of PG-Agent. (i) Convert chain-like episodes into a semantically rich page graph; (ii) With page graph as GUI prior knowledge, RAG technology assists the tailored multi-agent workflow to enhance GUI navigation.

from diverse devices, but usually treat them as independent items. For example, the navigation tasks on GUI involving sequences of multi-step operations across different pages, where each step provides crucial semantics and functionality for the task. Such linear knowledge restricts the agent to focusing mainly on the consecutive steps and thus lacks the perceptions of possible actions leading to other pages simultaneously. As a result, the semantic association information between pages is not fully integrated into the model, leading to significant challenges when the model performs complex tasks, especially on new tasks. During the deployment phase, it is probable that a single sequential episode cannot directly instruct the agent to finish the task, while the transition relationships from multiple episodes can provide more clues about possible actions and corresponding results to pave a new way to the target page. This raises a natural question: “How to explicitly model the semantic relationships among pages and enhance the perception capability of GUI Agents in new scenarios?”

Fortunately, the pages of GUI screens naturally form a page graph connected by the actions, and a sequential episode is essentially a path sampling on this graph. Inspired by this concept, we can reconstruct the sequential episodes into the page graph, which offers a more comprehensive understanding of the page transitions, rather than the fragmented page connections brought by discrete episodes. Any traversal path in the page graph can be regarded as an effective recombination of original independent knowledge items. Meanwhile, positioning at a node, the agent can easily obtain possible actions from the outgoing edges and perceive consequent pages to assist the navigation process.

Besides, to utilize the page graph as prior knowledge for the agent planning, the Retrieval-Augmented Generation (RAG) technology [15] is able to effectively leverage it without any parameter modifications, which enables the agent to adapt to different scenarios by simply switching the page graph. In this way, RAG is also able to explicitly retrieve the graph-structured information from the page graph, offering superior semantic perceptions of page transitions. Moreover, the exploration in the page graph of RAG is actually an accessing and integrating process of real actions in episodes, providing an authentic and reliable set of possible actions as guidelines. Previous works have adopted the RAG to retrieve guidelines like descriptions of widget functions [16] or reference trajectories in similar tasks [45], which are usually discrete without

graph structure to perceive the current scenario deeply. Therefore, a tailored retrieval strategy applied in the page graph is also critical.

To tackle the aforementioned challenges, we propose an automated pipeline to transform the sequential episodes into the page graphs, including three stages of page jump determination, node similarity check and page graph update. During the process, we check every action tuple (i.e., the action and the pages before and after it) and gradually update the page graph by combining consecutive in-page operations into one edge and similar pages as one node. Moreover, to retrieving guidelines from the page graphs and fully utilize them, we also design a multi-agent framework PG-Agent enhanced by tailored RAG technology. First, we use the summary of the current screen to locate similar nodes in the page graph and conduct breadth first search (BFS) to explore available actions, deriving guidelines like “perform *some actions* can lead to accomplish *some tasks*”. With comprehensive guidelines, we divide the reasoning process into several agents following existing works [35, 38], incorporate the task decomposition and inject the guidelines into the sub-task planning process, where the perceptions of the GUI scenario are particularly critical. We conduct extensive experiments on three benchmarks and the results demonstrate the effectiveness of PG-Agent, even if we only sample a few episodes to construct the page graph. The primary contributions of this paper can be summarized as follows:

- To model the transition relationships between GUI pages, We design an automated pipeline to reconstruct the discrete episodes into page graphs, which serve as external prior knowledge bases.
- We propose PG-Agent, a tailored multi-agent framework augmented by RAG technology. With the incorporation of task decomposition, guidelines retrieved from page graphs offer more targeted planning reference.
- Experimental results on three benchmarks demonstrate that PG-Agent exhibits superior navigation ability with the page graphs. Even if the episodes for page graph construction is limited, the effectiveness remains evident.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) can solve the issues of knowledge out-of-distribution in large models, such as output hallucination, lack of domain-specific knowledge, and outdated information by dynamically parsing the input content and retrieving relevant external knowledge [46]. Previous research on RAG mainly focused on question-answering related tasks [24, 33], such as using Table-to-Text methods to convert tabular data into textual form to enhance the document QA capability of LLM [25], using multi-modal embedding technology to uniform knowledge of different modalities to enhance multi-modal QA of the foundational model [13, 21], and utilizing the chunking methods to truncate the query and realize multi-granularity retrieval of external knowledge [5, 43]. Given that graph structures effectively represent complex data relationships and enable efficient information retrieval, the application of RAG technology to graph data has emerged as an interesting research focus [30]. GraphRAG [10] adopts a clustering approach by connecting small text blocks via semantic similarity, then applying community detection to group

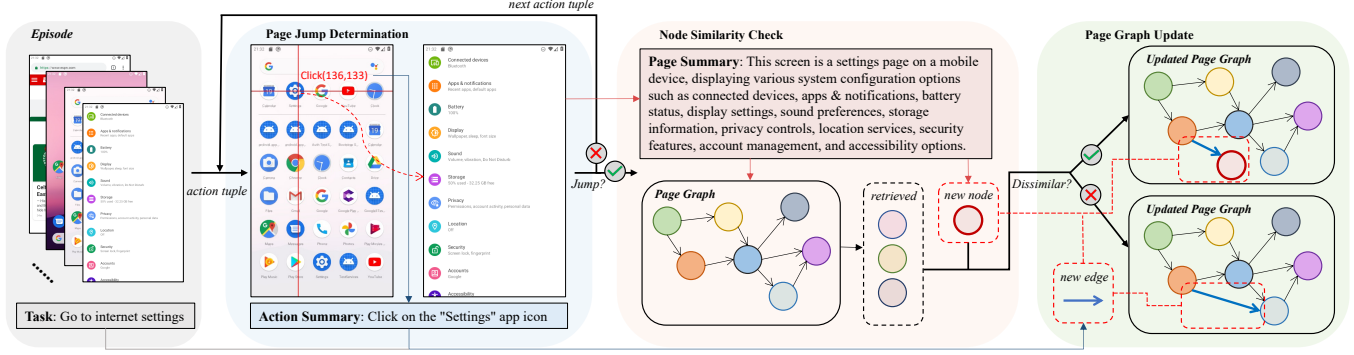


Figure 2: The overall pipeline of page graph construction. It comprises three stages: page jump determination, node similarity check and page graph update.

nodes, and finally summarizing query answers by analyzing node community responses. To model document relationships: Munikoti et al. [26] developed a heterogeneous document graph capturing multiple document-level relations, while Li et al. [17] and Wang et al. [37] established passage-level connections based on shared keywords. In the mobile agent scenario, there are also some works that use RAG to enhance the base model by providing additional interaction knowledge [16, 45]. However, they treat the traffic data as independent trajectory chains. We argue that in GUI scenarios, the data formed by the jump relationship between different screen pages is a global graph structure rather than discrete chains. Thus, ignoring the structured signals between pages will limit the model’s knowledge learning in this domain.

2.2 GUI Agent

Recent progress has begun to adopt LLMs [34, 41, 45] to build autonomous agents, leveraging LLMs’ extensive world knowledge and strong reasoning capabilities for task planning and execution to achieve human-like capabilities. Structural text replaces the original GUI image input into the LLMs. With the emergence of MLLMs, visual signals of images are projected into natural language space. Therefore, existing research tends to directly use MLLMs to build agents, so as to naturally process the visual information in the GUI field. One notable approach is to use large-scale general models, such as GPT-4v [28], as GUI agents. Many studies use prompt engineering to guide these models to perform complex tasks. AppAgent [44] is built on GPT-4v, generating guidance documents through exploration phase to assist decision-making. Mobile-Agent-v2 [35] first proposes multi-agent collaboration in GUI scenarios to improve the decision-making effect of each step. Another research direction focuses on fine-tuning smaller MLLMs [7, 12, 18] using GUI-specific datasets to bridge the domain gap between common images and GUI screens.

3 Method

In this section, we will illustrate how to transform chained action episodes into structured page graphs, along with readable guideline documents. Subsequently, we introduce **PG-Agent** that is tailored to leverage the page graphs to achieve precise GUI navigation.

3.1 Page Graph Construction

Naturally, the pages and their links within a website or an application form a graph structure, and an episode to complete a navigation task actually represents a walking path in this graph. Thus, with existing episodes in a specific GUI scenario where relevant websites or applications are limited, we can construct the corresponding page graph as future guidance in this scenario. We design our pipeline purely based on visual clues without additional modal inputs such as the page’s DOM or HTML architecture. The overall pipeline for page graph construction is shown in Figure 2, including three stages: page jump determination, node similarity check and page graph update.

Page Jump Determination. Assuming an action tuple in the episode E with task T is $(I_{before}, A, I_{after})$, where I_{before} and I_{after} represent the screenshot images before and after the action A , respectively. First, the actions need to be converted into natural language. For actions involving specific coordinates, their meanings will be lost when separated from the corresponding images. Therefore, a MLLM [2] is used to summarize them:

$$S_{action} = \text{MLLM}(I_{before}, A, \mathbb{P}_{action}), \quad (1)$$

where S_{action} is the summary of the action and \mathbb{P}_{action} is the prompt for action summary. Then, considering fewer redundant nodes and retrieval effectiveness, only unique pages are adopted to build the page graph. Therefore, it is first necessary to determine whether the action A triggers the page jump.

$$Y_{jump} = \text{MLLM}(I_{before}, I_{after}, S_{action}, \mathbb{P}_{jump}), \quad (2)$$

where $Y_{jump} \in [\text{Yes}, \text{No}]$ represents the determination result and \mathbb{P}_{jump} is the prompt template for page jump assessment. If the result is "No", the action is usually in-page operation like typing words or opening a drawer that do not lead to new pages, we will store this action summary S_{action} in a queue Q_{action} and directly process the next action tuple in the episode.

Node Similarity Check. When the result Y_{jump} is "Yes", it means this action successfully lead to different page. Then we take the following step to summarize the image after action based on the overall function of the page and key components displayed for node similarity check:

$$S_{page} = \text{MLLM}(I_{after}, \mathbb{P}_{page}), \quad (3)$$

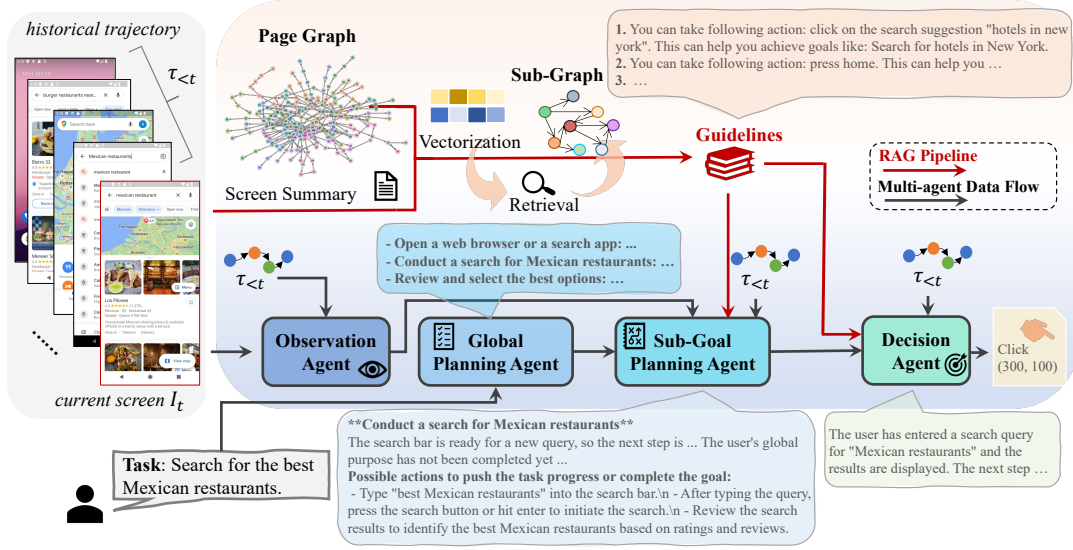


Figure 3: The framework of multi-agent workflow. It comprises two parts: RAG pipeline and multi-agent group.

where S_{page} denotes the summary of the page and \mathbb{P}_{page} is the template for page summary processing. Next, a dual-level similarity check from semantic aspect and pixel aspect was carried out. From semantic aspect, we employ similarity search with retrieval model to retrieve the top- n most similar page summaries from nodes of page graph $\mathcal{G} = (\mathcal{N}, \mathcal{V})$, which is empty initially:

$$S_{node} = \text{Retrieval}(S_{page}, \mathcal{G}), \quad (4)$$

where $S_{node} = [S_1, S_2, \dots, S_n]$ is composed of retrieved page summaries. Subsequently, we use the MLLM to further select the index of the most similar one:

$$id = \text{MLLM}(I_{after}, S_{node}, \mathbb{P}_{select}), \quad (5)$$

where $id \in [1, 2, \dots, n]$ and \mathbb{P}_{select} is the prompt for index selection. From pixel aspect, we extract original image I_{id} corresponding to selected page summary and compare it with image I_{after} to finally conclude whether the page can pass the node similarity check:

$$Y_{dissimilar} = \text{MLLM}(I_{after}, I_{id}, \mathbb{P}_{dissimilar}), \quad (6)$$

where $Y_{dissimilar} \in \{\text{Yes}, \text{No}\}$ is the check result and $\mathbb{P}_{dissimilar}$ is the prompt for similarity check.

Page Graph Update. When the result $Y_{dissimilar}$ is "Yes", it means that this new page is unique enough among existing nodes in the page graph \mathcal{G} , and we will create a new node $N_{new} = (S_{after}, L_{after})$ to represent the image I_{after} , where L_{after} is the image location of I_{after} . Image location L will only be utilized in Equation 6 to get original image I_{id} , so the final page graph will not contain the pixel information of images, avoiding the huge space occupancy of page graph. Besides, the node representing the image I_{before} can be formulated as N_{before} . Following, we incorporate the action summary S_{action} into stored action queue Q_{action} and combine it with the task description T of the episode as a new edge $E_{new} = (Q_{action}, T)$. In this way, we can guide the agent to follow these actions when completing similar tasks. Next, we insert directed tuple $(N_{before}, E_{new}, N_{new})$ into page graph \mathcal{G} to finish

update:

$$\mathcal{G} = \mathcal{G} \cup (N_{before}, E_{new}, N_{new}). \quad (7)$$

If the decision of similarity check is "No", page I_{after} can actually be represented by existing node N_{id} of image I_{id} , so the tuple to be inserted will be changed to $(N_{before}, E_{new}, N_{id})$:

$$\mathcal{G} = \mathcal{G} \cup (N_{before}, E_{new}, N_{id}). \quad (8)$$

3.2 Multi-agent Workflow

The workflow of agent framework could be formalized as a Markov Decision Process (MDP) [4, 34, 41]. Previous work mainly use a LLM, such as GPT-4 [29], to structure image text with the help of additional parsing tools [11, 45], or a separate MLLM [23, 44], such as GPT-4V [28], to preprocess the image using the Set-of-Marks strategy [40]. Then the design of the agent workflow is completed based on the prompt engineering. However, under the warper paradigm, the long-content poses a challenge to the reasoning performance of the model, making the model at risk of being "lost-in-the-middle" [20].

In this section, we adopt the multi-agent workflow that logically connects multiple MLLM-based agents with different roles. Each agent receives different input content and only completes specific tasks, which alleviates the context processing pressure of the model, and then spends more efforts on the task reasoning stage. Based on this architecture proposed before [35], we further introduce the task decomposition concept into agent group.

Our multi-agent workflow is shown in Figure 3 and it mainly consists of two key parts: (1) RAG pipeline, which retrieves helpful guidelines from page graph based on screen status; (2) Multi-agent group: agents with different roles, i.e., global planning agent, observation agent, sub-task planning agent and decision agent.

Guidelines Retrieval. The guidelines retrieved from the page graph are the core mechanism to enhance the generalization capability of agents in new GUI scenarios. First, we prompt the MLLM to analyze the current screen status I_t and generate a screen state

description S_{I_t} . Subsequently, S_{I_t} is vectorized to retrieve the top n most similar nodes \mathcal{N} from page graph \mathcal{G} :

$$S_{I_t} = \text{MLLM}(I_t, \mathbb{P}_{sum}), \quad (9)$$

$$\mathcal{N} = \text{Retrieval}(S_{I_t}, \mathcal{G}), \quad (10)$$

where \mathbb{P}_{sum} represents the template for screen summary. Then we extract the action queues Q stored in the outgoing edges \mathcal{E} of node set \mathcal{N} . Besides, starting from every outgoing edge \mathcal{E}_i , we conduct BFS with l layers and gather the tasks stored in the explored edges:

$$T_i = \text{BFS}(\mathcal{E}_i, \mathcal{G}), \quad (11)$$

where T_i is the gathered achievable tasks from the edge \mathcal{E}_i . We combine the action queue and achievable tasks as the guidelines:

$$G_{I_t} = [(Q_1, T_1), (Q_2, T_2), \dots, (Q_k, T_k)], \quad (12)$$

where k is the number of retrieved guidelines. Each tuple (Q_i, T_i) donates that the agent could follow the action queue Q_i to complete tasks recorded in set T_i .

Global Planning Agent. \mathcal{P}_{agent}^G is used to perform a global high-level sub-task decomposition of the user’s task T_g , breaking down the complex task into relatively simple and abstract sub-tasks (i.e., the global plan). In this way, the guidelines can inspire the agent to focus on completing the current sub-task. This process can formulated as:

$$\mathcal{R}_g = \mathcal{P}_{agent}^G(I_t, T_g). \quad (13)$$

Observation Agent. \mathcal{O}_{agent} is responsible for transforming the pixel information into textual perceptions. It observes the screen and provides useful visual clues along with a high-level abstract functional description. In this stage, we introduce the historical interaction record $\tau_{<t}$ from the previous moment to help \mathcal{O}_{agent} perceive task progress. With user’s task T_g , \mathcal{O}_{agent} can be formulated as:

$$\mathcal{R}_o = \mathcal{O}_{agent}(I_t, T_g, \tau_{<t}), \quad (14)$$

where $\tau_{<t} = (I_0, a_0, I_1, a_1, \dots, I_{t-1}, a_{t-1})$ and a_t represents the action executed at time-step t . The goal of the \mathcal{O}_{agent} is to directly provide explicit screen details to the decision agent \mathcal{D}_{agent} , so that it can pay more attention in reasoning.

Sub-Task Planning Agent. \mathcal{P}_{agent}^S selects a sub-task that matches the current screen state from the global plan \mathcal{R}_g , provides a detailed description of the current task suggestion, and generates a candidate action list. Based on screen observation \mathcal{R}_o , global plan \mathcal{R}_g , retrieved guidelines G_{I_t} , and historical trajectory $\tau_{<t}$, this process can be formulated as:

$$\mathcal{R}_s = \mathcal{P}_{agent}^S(I_t, \mathcal{R}_o, \mathcal{R}_g, G_{I_t}, \tau_{<t}). \quad (15)$$

Decision Agent. \mathcal{D}_{agent} eventually chooses the specific action to be performed in the current screen state I_t from the candidate action list \mathcal{R}_s via analyzing the previously generated content. The decision process can be formulated as:

$$\mathcal{R}_d = \mathcal{D}_{agent}(I_t, \mathcal{R}_o, \mathcal{R}_s, G_{I_t}, \tau_{<t}). \quad (16)$$

As shown in Figure 3, when given a screen image I_t , **RAG pipeline** will summarize I_t , vectorizes[9] the summary to retrieve similar nodes from page graph \mathcal{G} and explore them to generate guidelines G_{I_t} . Then, **Observation Agent** \mathcal{O}_{agent} will carefully perceive the screen status I_t and produce detailed description of

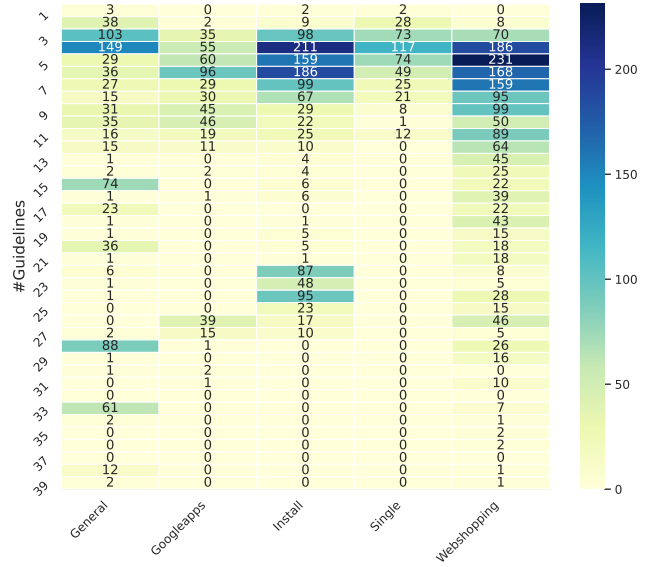


Figure 4: The data distribution of guidelines in AITW dataset. The x-axis represents the scenario category and the y-axis represents the number of retrieved guidelines at each step.

the page. Next, **Global Planning Agent** \mathcal{P}_{agent}^G will decouple user’s global task T_g into several clear, coherent and relatively easy sub-tasks \mathcal{R}_g . Afterwards, **Sub-Task Planning Agent** \mathcal{P}_{agent}^S will conduct in-depth analysis of the context information, including $I_t, \mathcal{R}_o, \mathcal{R}_g, G_{I_t}$ and $\tau_{<t}$, and complete the fine-grained plan \mathcal{R}_s of the current sub-task under the help of guidelines G_{I_t} . Finally, the **Decision Agent** \mathcal{D}_{agent} will use $\mathcal{R}_o, \mathcal{R}_s, G_{I_t}$, and $\tau_{<t}$ to generate the final decision \mathcal{R}_d to predict the action that should be performed in the current state to advance the task T_g .

For more details on page graph and multi-agent workflow, please refer to the **Supplementary Material**.

4 Experiment

4.1 Experimental Setting

Benchmark Dataset. To assess the navigation ability in both mobile and website environment, we evaluate our PG-Agent on two GUI agent datasets: Android in the Wild (AITW) [32], Mind2Web [8] and GUI Odyssey [22]:

- **AITW:** The episodes in AITW dataset are collected in Android mobile phone environment, which are divided into five scenarios: General, Install, GoogleApps, Single, and WebShopping. We follow the split setting of SeeClick [7]. For simplicity, we randomly sample 1/10 episodes from training split to construct concise page graphs based on different scenarios. The specific statistics are shown in Table 8.
- **Mind2Web:** Mind2Web dataset contains over 2,000 open-ended tasks collected from 137 real websites, covering five scenarios: Entertainment, Travel, Shopping, Service, and Info. Also, we randomly sample episodes from the training set. The benchmark on Mind2Web is not divided by scenarios, but categorized into cross-task, cross-website, and cross-domain to test the generalization

Table 1: Comparison of PG-Agent with different methods on Mind2Web dataset. The best and second-best results in each column are highlighted in bold font and underlined.

Method	Cross-Task			Cross-Website			Cross-Domain		
	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR
MindAct	<u>55.1</u>	75.7	<u>52.0</u>	<u>42.0</u>	65.2	38.9	42.1	66.5	39.6
GPT-4V	46.4	73.4	40.2	38.0	67.8	32.4	42.4	69.3	36.8
Qwen2.5-VL-72B	31.9	84.6	26.2	35.7	80.7	27.9	32.0	83.2	25.0
OmniParser	42.4	87.6	39.4	41.0	84.8	36.5	45.5	85.7	<u>42.0</u>
PG-Agent	59.0	<u>84.7</u>	52.9	57.3	<u>81.2</u>	48.7	60.2	<u>84.5</u>	53.3

Table 2: Comparison of PG-Agent with different methods on GUI Odyssey dataset. The best and second-best results in each column are highlighted in bold font and underlined.

Method	Tool	Information	Shopping	Media	Social	Multi-Apps	Overall
GeminiProVision	3.3	4.0	2.3	4.3	1.5	3.2	4.9
CogAgent	11.8	15.7	10.7	9.2	11.7	13.1	10.7
GPT-4V	18.8	23.5	20.2	19.2	16.9	13.8	19.0
GPT-4o	20.4	20.8	16.3	31.9	15.4	21.3	16.7
Qwen2.5-VL-72B	<u>46.6</u>	<u>60.0</u>	<u>44.0</u>	<u>32.4</u>	<u>46.1</u>	54.6	<u>42.4</u>
PG-Agent	48.6	61.5	47.2	35.5	46.9	<u>52.6</u>	47.7

ability of the agent. Therefore, the Service and Info scenarios that only appear at cross-domain test do not have corresponding page graphs, so we will use the page graphs of other scenarios during evaluation.

- **GUI Odyssey:** GUI Odyssey dataset is designed to evaluate the navigation ability of the agent in cross-app tasks. This dataset contains more than 7,000 episodes with an average of 15+ steps, including 6 different scenarios from 201 apps. Similarly, we sample part of training episodes of GUI Odyssey to build page graphs.

The specific statistics of episodes sampling of dataset Mind2Web and GUI Odyssey are listed in supplementary materials.

Model. In this paper, we adopt BGE-M3 [3] as our vectorization model and FAISS technique [9] for similarity retrieval. Besides, we utilize Qwen2.5-VL-72B [2] as our base MLLM considering its strong ability at understanding GUI screens.

Hyperparameters. According to the statistics of retrieved guidelines (*GL*), as shown in Figure.4, we set the maximum number of *GL* (Equation 12) to 20 for AITW dataset. Besides, we set 20 for GUI Odyssey and 10 for Mind2Web, whose distribution of *GL* in different scenarios can be seen in the supplementary materials. Besides, we set the maximum number of layers *l* for BFS to 3 and the number of retrieved nodes of page similarity search *n* to 4.

4.2 Main Result

AITW. We follow the setting of AITW to calculate the action matching score as the metric. As shown in Table 3, PG-Agent yields the best average performance compared to current API-based agents. Among the scenarios, the action accuracy in GoogleApps is the most prominent, which exceeds state-of-art result by 13.4%. The result demonstrates that graph RAG technique can help API-based

agent improve the execution accuracy in scenarios where prior knowledge is available. For error cases in Single scenario, we find the length of these episodes is short and the step where the task is supposed to end has ambiguity, i.e., our agent tends to continue executing some actions to completely finish the task. We further analyze this situation in the Supplementary Materials.

Table 3: Comparison of PG-Agent with different methods on AITW dataset. The best and second-best results in each column are highlighted in bold font and underlined.

Method	General	Install	G.Apps	Single	WebShop.	Overall
ChatGPT-CoT	5.9	4.4	10.5	9.4	8.4	7.7
PaLM2-CoT	-	-	-	-	-	39.6
GPT-4V	41.7	42.6	49.8	<u>72.8</u>	45.7	50.5
Qwen2.5-VL-72B	35.9	<u>58.5</u>	<u>58.8</u>	50.7	36.6	48.1
OmniParser	<u>48.3</u>	57.8	51.6	77.4	<u>52.9</u>	<u>57.5</u>
PG-Agent	51.9	62.4	65.0	64.7	53.7	59.5

Mind2Web. In Mind2Web dataset, we calculate element accuracy (Ele.Acc), operation f1 (Op.F1) and step success rate (Step SR) as the metrics. Results in Table 1 show that PG-Agent achieves the optimal performance in both Ele.Acc and Step.SR metric, and the second-best in Op.F1. Besides, the significant improvements can be observed in cross-domain split, where we lack for relevant prior knowledge in Service and Info scenarios. This indicates that the episodes from other scenarios also provide valuable reference, which proves the generality of constructed page graph.

GUI Odyssey. For GUI Odyssey, we adhere the original metric setting [22]. From the results in Table 2, we can observe PG-Agent produces the best results in most scenarios and surpasses other

Table 4: Ablation results on Mind2Web. The best and second-best results in each column are highlighted in bold font and underlined. GL, STP-Agent, D-Agent are the abbreviations of guidelines, sub-task planning agent and decision agent respectively.

Method	Cross-Task			Cross-Website			Cross-Domain		
	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR	Ele.Acc	Op.F1	Step SR
PG-Agent	<u>59.0</u>	84.7	52.9	57.3	81.2	48.7	60.2	84.5	53.3
w/ GL in D-Agent	58.1	84.0	50.7	<u>57.5</u>	<u>82.0</u>	<u>48.5</u>	<u>59.9</u>	82.7	51.5
w/ GL in STP-Agent	59.4	<u>84.5</u>	<u>52.4</u>	56.9	83.1	48.0	59.5	<u>83.4</u>	<u>52.1</u>
w/o GL	58.9	<u>82.8</u>	50.2	57.6	80.6	47.6	59.4	81.3	50.4

Table 5: Ablation results of guidelines for different actions on Mind2Web. The metric is Op.F1 value, where the best result is highlighted in bold, and w/o GL means removing the RAG pipeline.

Method	Cross-Task			Cross-Website			Cross-Domain		
	CLICK	SELECT	TYPE	CLICK	SELECT	TYPE	CLICK	SELECT	TYPE
PG-Agent	90.9	31.2	49.5	88.0	47.3	44.0	88.8	47.1	56.3
w/o GL	92.9	29.8	29.6	91.0	40.5	26.0	89.6	40.6	31.7

API-based agents, while only in Multi-Apps scenario it is sub-optimal. This demonstrates that the guidelines retrieved from page graph cast some insights into unfamiliar scenario for the agent and actually improve the planning process and execution process during the navigation.

4.3 Ablation Study

In our PG-Agent, as shown in Figure 3, our graph RAG pipeline extracts relevant guidelines (GL) from the page graph (Section 3.1) and acts in the planning stage (Sub-Task Planning Agent) and decision stage (Decision Agent) respectively. In this section, we use the control variable method to analyze the impact of GL on PG-Agent. Specifically, we prompt the retrieved GL to different agents, and the results are shown in Table 6 and Table 4. It can be seen that on AITW[32], our PG-Agent achieves the best results, while removing GL (w/o GL) leads to a general decline in performance.

Table 6: Ablation results on AITW. The best and second-best results in each column are highlighted in bold font and underlined. GL, STP-Agent, D-Agent are the abbreviations of Guidelines, Sub-Task Planning Agent and Decision Agent respectively.

Method	General	Install	G.Apps	Single	WebShop.	Overall
PG-Agent	51.9	62.4	65.0	64.7	53.7	59.5
w/ GL in D-Agent	50.8	<u>60.5</u>	63.8	66.6	53.4	<u>59.0</u>
w/ GL in STP-Agent	<u>51.4</u>	59.5	62.8	<u>66.1</u>	52.8	58.5
w/o GL	50.0	59.8	63.4	65.4	52.7	58.3

Meanwhile, introducing GL to different agents also brings performance improvements. Furthermore, we find that the benefits of introducing GL in the Decision Agent (D-Agent) are greater than those in the Sub-Task Planning Agent (STP-Agent). The same results are also observable in Table 4, but in the web navigation

tasks [8], agents have different preferences for GL under different tasks; for example, in the Cross-Task and Cross-Domain split, introducing GL to STP-Agent is better than D-Agent, but this result is reversed in the Cross-Website split. We attribute these results to differences in interaction logic across scenarios and varying navigation preferences of the base model for different devices.

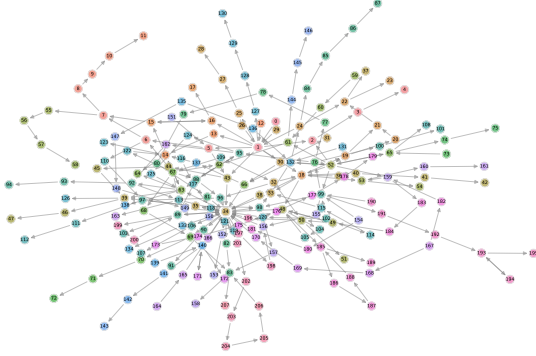
To further validate GL’s advantages, we conduct a fine-grained analysis of its impact on each decision step in the Mind2Web[8] dataset. As shown in Table 5, the introduction of GL greatly improves the Opt.F1 score of the ‘SELECT’ and ‘TYPE’ actions. Regarding the decrease in performance of the ‘CLICK’ action, we analyze the data and find that the reason is due to the inconsistency of the labels in the dataset itself; that is, the ‘SELECT’ action has two label definitions at the same time: 1) two consecutive ‘CLICK’ actions; 2) a single ‘SELECT’ action. Our PG-Agent tends to choose more reasonable ‘SELECT’ actions. However, this can lead to situations where it is judged as having failed, even when it executes the correct action. As a result, the Opt.F1 score for the ‘SELECT’ type decreases.

4.4 Page Graph Analysis

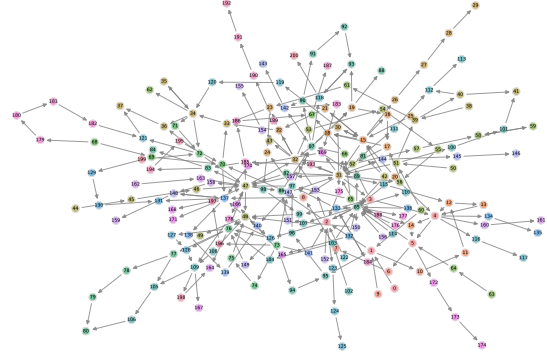
For publicly available datasets (e.g., AITW, Mind2Web and GUI Odyssey), there are abundant data of episodes for the construction of page graphs, but it actually takes substantial costs for data collection. Therefore, in the previous evaluation, we only sample a small subset of episodes to build the graph, demonstrating the practicality of our framework. In this section, we use the full episodes from the dataset to build the graph and compare the result with Section 4.2. As shown in Table 7, we find that PG-Agent utilizing full episodes only yields better performance in specific scenarios, where the random sampling version remains a competitive accuracy score. The results indicate that even if there are only limited episodes for reference, the page graph built from them can still provide effective guidance for PG-Agent.

Table 7: The impact of the page graph on PG-Agent constructed with different data of episodes, where random sampling follows the setting in Section 4.1, while full episodes means we utilize all episodes from the training set.

Data Source	AITW					Mind2Web		
	General	Install	G.Apps	Single	WebShop.	Cross-Task	Cross-Website	Cross-Domain
Random Sampling	51.9	62.4	65.0	64.7	53.7	52.9	48.7	53.3
Full Episodes	50.5	59.5	63.2	61.4	54.6	52.6	50.0	54.0



(a) Install



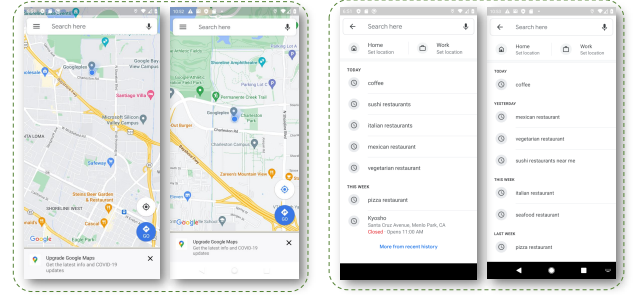
(b) WebShopping

Figure 5: Examples of page graph visualizations of scenarios in AITW dataset.**Table 8: Statistics of sampled episodes in different scenarios of dataset AITW and corresponding page graph.**

Scenario	# Episodes	# Images	# Nodes	# Edges
General	43	341	132	168
Install	55	538	208	286
G.Apps	24	198	67	93
Single	55	194	92	70
WebShop.	56	712	201	323
Total	231	1983	700	940

To analyze the page graph deeply, we also collect the statistics of sampled episodes and the page graph. From Table 8, we can find that the nodes in page graph is much less than the images of corresponding episodes, suggesting that there are lots of repeated pages in the same scenario. Meanwhile, the number of edges is also smaller than the number of actions (usually the same as number of images), which suggests that some consecutive in-page actions have been combined at one edge for the simplicity of the graph structure. More statistics on Mind2Web and GUI Odyssey datasets are listed in the Supplementary Materials.

Besides, we visualize some page graphs constructed by the episodes from different scenarios in AITW dataset. As shown in Figure 5, we can see that the in-degree or out-degree of some nodes in the graphs is greater than 1, indicating that some similar pages in the episodes share the same nodes. We also visualize some cases of

**Figure 6: Cases of original images sharing the same node.**

these similar pages in Figure 6, demonstrating the effectiveness of the dual-level similarity check.

5 Conclusion

In this paper, we design an automated pipeline to reconstruct the discrete chained episodes into the page graph, capturing the complex transition relationships between screen pages. To fully utilize this prior knowledge as the perceptions of GUI scenarios, we further propose a tailored multi-agent framework PG-Agent equipped with the RAG technology to retrieve perception guidelines from the graph to improve the planning and execution process. Extensive experiments on three benchmark datasets illustrate the effectiveness of PG-Agent powered by page graphs, even when the available episodes are limited.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62372408, 62476245). This work was supported by Ant Group Research Fund.

References

- [1] Meta AI. 2024. Llama 3. <https://github.com/meta-llama/llama3> Accessed: 2024-11-12.
- [2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-VL Technical Report. *arXiv preprint arXiv:2502.13923* (2025).
- [3] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv:2402.03216* [cs.CL]
- [4] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. 2024. AutoManual: Constructing Instruction Manuals by LLM Agents via Interactive Environmental Learning. *arXiv:2405.16247* [cs.AI] <https://arxiv.org/abs/2405.16247>
- [5] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, and Hongming Zhang. 2024. Dense X Retrieval: What Retrieval Granularity Should We Use?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Miami, Florida, USA, 15159–15177. doi:10.18653/v1/2024.emnlp-main.845
- [6] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. 2024. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821* (2024).
- [7] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. 2024. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 9313–9332. <https://aclanthology.org/2024.acl-long.505>
- [8] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- [9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). *arXiv:2401.08281* [cs.LG]
- [10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitanansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv:2404.16130* [cs.CL] <https://arxiv.org/abs/2404.16130>
- [11] Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. AutoGuide: Automated Generation and Selection of Context-Aware Guidelines for Large Language Model Agents. *arXiv:2403.08978* [cs.CL] <https://arxiv.org/abs/2403.08978>
- [12] Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14281–14290.
- [13] Ziniu Hu, Ahmet Iscen, Chen Sun, Zirui Wang, Kai-Wei Chang, Yizhou Sun, Cordelia Schmid, David A. Ross, and Alireza Fathi. 2023. REVEAL: Retrieval-Augmented Visual-Language Pre-Training with Multi-Source Multimodal Knowledge Memory. *arXiv:2212.05221* [cs.CV] <https://arxiv.org/abs/2212.05221>
- [14] Hanyu Lai, Xiao Liu, Lat Long Long, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. AutoWebGLM: A Large Language Model-based Web Navigating Agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5295–5306.
- [15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv:2005.11401* [cs.CL] <https://arxiv.org/abs/2005.11401>
- [16] Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. 2024. AppAgent v2: Advanced Agent for Flexible Mobile Interactions. *arXiv:2408.11824* [cs.HC] <https://arxiv.org/abs/2408.11824>
- [17] Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2024. Graph Neural Network Enhanced Retrieval for Question Answering of LLMs. *arXiv:2406.06572* [cs.CL] <https://arxiv.org/abs/2406.06572>
- [18] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465* (2024).
- [19] Mario Linares-Vásquez, Kevin Moran, and Denys Poshyvanyk. 2017. Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 399–410.
- [20] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranajpe, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. *arXiv:2307.03172* [cs.CL] <https://arxiv.org/abs/2307.03172>
- [21] Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. 2024. Generative Multi-Modal Knowledge Retrieval with Large Language Models. *arXiv:2401.08206* [cs.IR] <https://arxiv.org/abs/2401.08206>
- [22] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. GUI Odyssey: A Comprehensive Dataset for Cross-App GUI Navigation on Mobile Devices. *arXiv:2406.08451* [cs.CV] <https://arxiv.org/abs/2406.08451>
- [23] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. OmniParser for Pure Vision Based GUI Agent. *arXiv:2408.00203* [cs.CV] <https://arxiv.org/abs/2408.00203>
- [24] Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, and Qianren Wang. 2024. Exploring the Impact of Table-to-Text Methods on Augmenting LLM-based Question Answering with Domain Hybrid Data. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*. Association for Computational Linguistics, Mexico City, Mexico, 464–482. <https://aclanthology.org/2024.naacl-industry.41/>
- [25] Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, and Qianren Wang. 2024. Exploring the Impact of Table-to-Text Methods on Augmenting LLM-based Question Answering with Domain Hybrid Data. *arXiv:2402.12869* [cs.CL] <https://arxiv.org/abs/2402.12869>
- [26] Sai Munikoti, Anurag Acharya, Sridevi Wagle, and Sameera Horawalavithana. 2023. ATLANTIC: Structure-Aware Retrieval-Augmented Language Model for Interdisciplinary Science. *arXiv:2311.12289* [cs.CL] <https://arxiv.org/abs/2311.12289>
- [27] Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. 2024. GUI Agents: A Survey. *arXiv:2412.13501* [cs.AI] <https://arxiv.org/abs/2412.13501>
- [28] OpenAI. 2023. GPT-4V(ision) System Card. (1 2023). doi:10.26181/25479208.v1
- [29] OpenAI. 2024. GPT-4 Technical Report. *arXiv:2303.08774* [cs.CL] <https://arxiv.org/abs/2303.08774>
- [30] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohu Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. *arXiv:2408.08921* [cs.AI] <https://arxiv.org/abs/2408.08921>
- [31] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. UI-TARS: Pioneering Automated GUI Interaction with Native Agents. *arXiv preprint arXiv:2501.12326* (2025).
- [32] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P. Lillicrap. 2023. Android in the Wild: A Large-Scale Dataset for Android Device Control. *CoRR* abs/2307.10088 (2023). *arXiv:2307.10088*
- [33] Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. 2024. Generate-then-Ground in Retrieval-Augmented Generation for Multi-hop Question Answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 7339–7353. doi:10.18653/v1/2024.acl-long.397
- [34] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 8634–8652.
- [35] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2025. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems* 37 (2025), 2686–2710.
- [36] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (March 2024). doi:10.1007/s11704-024-02031-1
- [37] Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2023. Knowledge Graph Prompting for Multi-Document Question Answering. *arXiv:2308.11730* [cs.CL] <https://arxiv.org/abs/2308.11730>

- [38] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-Agent-E: Self-Evolving Mobile Assistant for Complex Tasks. *arXiv:2501.11733 [cs.CL]* <https://arxiv.org/abs/2501.11733>
- [39] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456* (2024).
- [40] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V. *arXiv:2310.11441 [cs.CV]* <https://arxiv.org/abs/2310.11441>
- [41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- [42] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv preprint arXiv:2408.01800* (2024).
- [43] Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial Report Chunking for Effective Retrieval Augmented Generation. *arXiv:2402.05131 [cs.CL]* <https://arxiv.org/abs/2402.05131>
- [44] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771* (2023).
- [45] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19632–19642.
- [46] Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. 2024. Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely. *arXiv:2409.14924 [cs.CL]* <https://arxiv.org/abs/2409.14924>