# Task Planning and Execution Based on LLM-powered Mobile Agents

**Junpeng Chen[1*], Ruichi Zhang[1*], Zenghui Huang[2*], Jun Xiao[2*], Shihao Hou[1*]**

[1]School of Informatics, Xiamen University, [2]Institute of Artificial Intelligence, Xiamen University
23020241154372@stu.xmu.edu.cn, 23020241154363@stu.xmu.edu.cn, 36920241153220@stu.xmu.edu.cn,
36920241153261@stu.xmu.edu.cn, 23020241154399@stu.xmu.edu.cn

## Abstract

In this paper, we present an enhanced approach to Mobile-Agent-v2, leveraging Retrieval-Augmented Generation (RAG) to improve intelligent agents' performance in mobile device operations. While existing mobile operation assistants have shown promise, they often struggle with complex navigation and focused content retrieval. Our method integrates RAG with a multi-agent architecture, enabling dynamic access to relevant knowledge during task execution. The framework consists of five key modules: zero-shot instruction classification, command generation, retrieval-augmented generation, multi-agent architecture, and instruction execution. Through extensive experiments across four fundamental mobile tasks (ordering drinks, booking tickets, subscribing, and playing videos), we demonstrate that our RAG-enhanced approach significantly outperforms baseline methods. Results show that our method with the Qwen model achieves perfect performance across all metrics, while requiring fewer interaction steps compared to alternatives. This work contributes to advancing the capabilities of mobile device automation through more efficient and accurate task execution.

## Introduction

In recent years, multimodal large language models (MLLMs(Bai et al. 2023; Wang et al. 2023; Hu et al. 2024)) have demonstrated significant potential in task planning and reasoning. The automation of tasks on smartphones, representing a practical application of multi-modal technology, is increasingly seen as a pivotal innovation in the evolution of AI-enhanced mobile devices(Yao et al. 2022a; Deng et al. 2024). However, existing models face limitations in visual perception, making it challenging to effectively perform operations based on screen content. This is particularly evident in scenarios requiring complex task navigation and focused content retrieval, where the performance of traditional single-agent architectures is hindered by lengthy context sequences and interleaved multimodal data formats.

To address these challenges, we Consider using Mobile-Agent(Wang et al. 2024c), an autonomous mobile device operation assistant that leverages advanced visual perception tools to identify and interact with elements on mobile screens. Unlike approaches reliant on system-level files like XML(Zhang et al. 2023) or HTML, Mobile-Agent operates purely through screen-based visual inputs, offering enhanced adaptability across diverse environments. By integrating capabilities such as self-planning, task decomposition, and error reflection, Mobile-Agent achieves high accuracy and completion rates, even in complex multi-app scenarios.

Building on this, Mobile-Agent-v2(Wang et al. 2024a) employs a multi-agent architecture to further improve navigation and operational efficiency. This framework introduces specialized roles—planning agent, decision agent, and reflection agent—to manage task progress, maintain focused content, and correct errors dynamically. Experimental evaluations reveal that Mobile-Agent-v2 significantly outperforms single-agent systems, achieving notable improvements in task success rates and operational accuracy across various apps and operating systems. These advancements highlight the potential of multi-agent collaboration in enhancing mobile device automation.

Leveraging RAG(Lewis et al. 2020), we have enhanced Mobile-Agent-v2 to incorporate a dynamic retrieval system that enriches its operational context with relevant, up-to-date information. This integration empowers Mobile-Agent-v2 to execute tasks with heightened accuracy, particularly in scenarios demanding intricate navigation and precise content manipulation, thus pushing the boundaries of mobile device automation. By harnessing the strength of RAG, Mobile-Agent-v2 can access a vast knowledge base to inform its decision-making process, ensuring that it operates with a depth of understanding that was previously unattainable. This is achieved by combining parametric memory with non-parametric memory, a dense vector index of Wikipedia accessed with a pre-trained neural retriever, thereby addressing some of the issues related to knowledge revision and expansion, and allowing for the inspection and interpretation of accessed knowledge(Guu et al. 2020; Karpukhin et al. 2020; Petroni et al. 2020).

The RAG system within Mobile-Agent-v2 works in tandem with the multi-agent architecture, providing the planning agent with the necessary context to devise effective strategies, the decision agent with the most current data to execute commands accurately, and the reflection agent with the historical insights to learn from past actions. This synergy not only streamlines the operation process but also sig-

[*]These authors contributed equally.

nificantly boosts the overall performance, making Mobile-Agent-v2 a formidable tool in the realm of mobile device operation.

## Related Work

**Agent** The basic components of an agent based on a large model should include planning, tools, actions, and memory. The execution process of Agent is similar to the way people do things. (Yao et al. 2022b) suggest that when the Agent performs the next action, the big model's own thinking process should be added, and the thinking process, execution tools and parameters, and execution results should be put into the prompt words. This will enable the model to better reflect on the completion of the current and previous tasks, thereby improving the model's problem-solving ability.

**Mobile agent** With the development of large language models(LLMs), agents have become an important way for them to demonstrate their strong capabilities. Among them, the applications of multimedia agents in different fields such as network management, e-commerce, energy efficiency and metering; wireless multimedia sensors, grid computing and grid services, distributed data mining, multimedia, human tracking, security, affective computing, climate environment and weather, e-learning, positioning, recommendation and semantic web services show its wide range of uses(Abdelkader 2009). (Al-Jaljouli and Abawajy 2010) implemented a mobile agent in e-commerce to search and filter information of interest from an e-marketplace; (Li et al. 2023) proposed a mobile agent-based framework that allows different agents to collaborate and communicate with each other to jointly complete task planning; (Kakiuchi et al. 2009) proposed a mobile device-based approach that allows large language models to make full use of existing devices to track human traces in real time, demonstrating the ability of mobile agents to use tools.

**Muiti-agents** Although single-agent systems excel in specific tasks, they may encounter limitations when dealing with com plex problems that require extensive collaboration and collective intelligence. This is where multi-agent sys tems (MAS) come into play. MAS is a complex system composed of multiple interacting intelligent agents (Hu et al. 2021), capable of simulating social interactions and teamwork in the real world, enhancing overall adaptability and effi ciency through decentralized decision-making processes and information sharing. (Abdelnabi et al. 2023)(Xu et al. 2023)(Mukobi et al. 2023)(Wang et al. 2024b) integrate multi-agent interaction with game theoretic strategies, aiming to enhance both the cooperative and decision abilities, opening up a framework for multimedia agents to solve tasks.

## Method

In this section, we will provide a detailed overview of the architecture of our proposed framework. The operation of our proposed framework is iterative, and its process is depicted in Figure 1. Our proposed framework consists of five modules: (a) **Zero-Shot Instruction Classification**, for categorizing user instructions; (b) **Command Generation**, which generates execution commands based on task names and predefined templates; (c) **Retrieval-Augmented Generation (RAG)**, which retrieves task-relevant external knowledge; (d) **Multi-Agent Architecture**, a decision-making framework with reflection mechanisms; and (e) **Instruction Execution Module**, which primarily executes mobile commands and also includes visual perception functionalities (OCR and icon detection).

### Zero-Shot Instruction Classification

To efficiently classify user instructions across multiple task categories, we propose a Zero-Shot Instruction Classification approach. Given a user instruction $I$ and a set of task categories $C = \{c_1, c_2, \ldots, c_n\}$, the goal is to map $I$ to the most relevant category $c_k$, where $k \in [1, n]$.

The classification process is performed by a large language model (LLM) through the conditional probability $P(c_k \mid I)$, formulated as:

$$\hat{c} = \arg\max_{c_k \in C} P(c_k \mid I) \tag{1}$$

To improve classification robustness, the historical classification results $M$ are updated iteratively through a memory mechanism:

$$M_t = f(M_{t-1}, \hat{c}_t) \tag{2}$$

where $M_t$ is the memory state at step $t$, and $\hat{c}_t$ represents the current classification result.

At each iteration, the updated memory $M_t$ is used to refine the classification, obtaining the next predicted category $\hat{c}_{t+1}$. This process continues until the results meet the desired performance criteria. Formally, we express this iterative process as:

$$\hat{c}_{t+1} = \arg\max_{c_k \in C} P(c_k \mid M_t) \tag{3}$$

This iterative classification continues until we reach the final classification $\hat{c}_T$, where $T$ is the stopping time when the classification result is deemed satisfactory. The final category is selected as the output:

$$\hat{c} = \hat{c}_T \tag{4}$$

where $T$ corresponds to the number of iterations required to converge to the optimal classification.

### Command Generation

Once the instruction is classified, execution commands are generated based on the task name $c_k$ and associated templates. Let $PT = \{PT_1, PT_2, \ldots, PT_n\}$ be a set of prompt templates where $PT_k$ corresponds to task $c_k$. The command generation process can be expressed as:

$$INS = f(I, PT_k, R_{topk}) \tag{5}$$

where $INS$ is the generated command and $R_{topk}$ represents the top-$k$ most relevant pieces of knowledge.

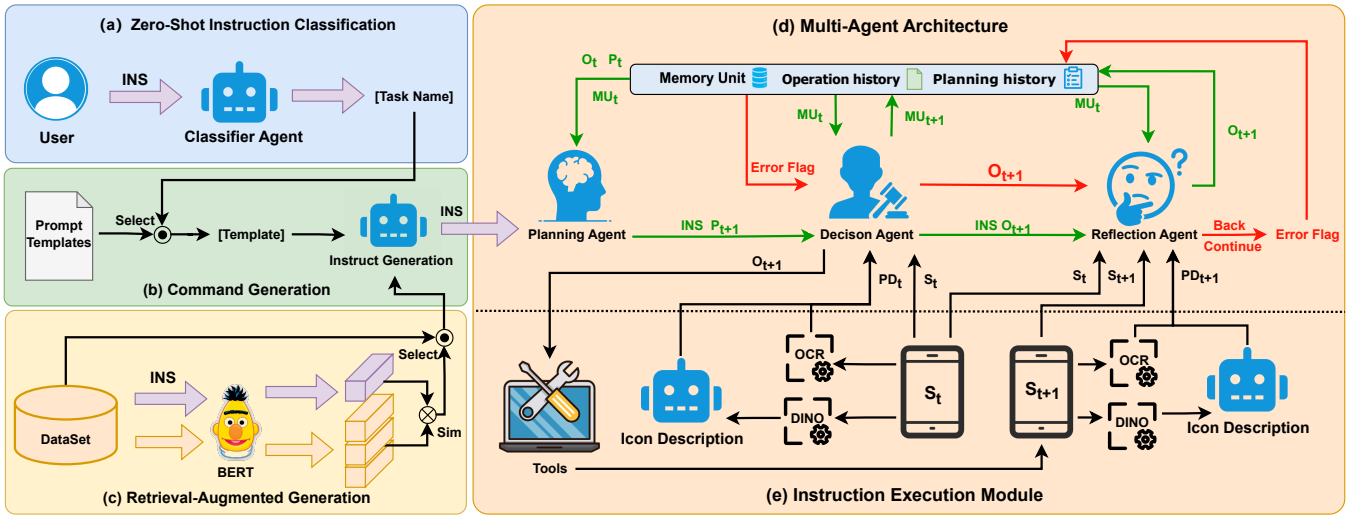


Figure 1: Illustration of the overall framework of our method, which consists of five modules: (a) **Zero-Shot Instruction Classification**, for categorizing user instructions; (b) **Command Generation**, which generates execution commands based on task names and predefined templates; (c) **Retrieval-Augmented Generation (RAG)**, which retrieves task-relevant external knowledge; (d) **Multi-Agent Architecture**, a decision-making framework with reflection mechanisms; (e) **Instruction Execution Module**, which primarily executes mobile commands and also includes visual perception functionalities (OCR and icon detection).

## Retrieval-Augmented Generation (RAG)

To enhance the accuracy of command generation, we adopt a **Retrieval-Augmented Generation** (RAG) mechanism. Given a user instruction $I$ and a dataset $D = \{d_1, d_2, \ldots, d_m\}$, the textual embedding for $I$ and $D$ is obtained using a pre-trained embedding model (e.g., BERT(Kenton and Toutanova 2019)):

$$\mathbf{E}_I = Embed(I) \tag{6}$$

$$\mathbf{E}_D = \{Embed(d_j) \mid j = 1, 2, \ldots, m\} \tag{7}$$

The similarity scores between $I$ and each $d_j$ are computed as:

$$S_j = cos(\mathbf{E}_I, \mathbf{E}_{D_j}) \tag{8}$$

where $cos$ is the cosine similarity. The top-$K$ relevant results $R_{topk} = \{d_{k_1}, d_{k_2}, \ldots, d_{k_K}\}$ are retrieved and used to enrich the command generation process.

## Multi-Agent Architecture

In this subsection, we describe the Multi-Agent Architecture of our proposed framework. This Multi-Agent Architecture(Wang et al. 2024a) with reflection mechanisms to handle decision-making and execution refinement. The architecture consists of three agents: a planning agent, a decision agent and a reflection agent.

- **Planning Agent**: The Planning Agent is designed to reduce the reliance on lengthy operation histories during decision making. It summarizes historical operations and tracks planning history across iterations. At the $t$-th iteration, the Planning Agent observes the previous operation $O_t$ generated by the Decision Agent and updates the planning history $P_t$ to $P_{t+1}$, reflecting the planning completed so far. The updated planning history $P_{t+1}$ is then passed to the Decision Agent, assisting it in generating the next operation by focusing on unfinished planning. The process can be expressed as:

$$P_{t+1} = PA(INS, O_t, P_t, MU_t) \tag{9}$$

where $PA$ represents the large language model (LLM) employed by the Planning Agent, $INS$ denotes the instruction, $MU_t$ represents the import content from the memory unit, and $P_{t+1}$ denotes the updated planning history.

- **Decision Agent**:The Decision Agent is responsible for generating operations $O_{t+1}$ and executing them on the device. It utilizes the planning history $P_t$, import content from the memory unit $MU_t$, and the page description result of the screen at iteration $t$ $PD_t$ to decide the next action. Additionally, it updates the memory unit by observing task-relevant content on the current screen. The Decision Agent's operation generation is represented as:

$$O_{t+1} = DA(INS, P_{t+1}, MU_t, R_t, S_t, PD_t) \tag{10}$$

where $DA$ represents the LLM of the Decision Agent, $R_t$ denotes the reflection result from the Reflection Agent, $S_t$ represents the current screen state, and $PD_t$ represents the page description result of the screen at iteration $t$. The memory unit update is represented by:

$$MU_{t+1} = DA(INS, MU_t, S_t, PD_t) \tag{11}$$

enabling the agent to store and leverage task-relevant focus content for future decisions.

| Method | Order drink | | Book ticket | | Subscribe | | Play video | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SR | SA | SR | SA | SR | SA | SR | SA | SR | SA |
| Mobile-Agent V2 (zhipu) | 0/2 | 0.375 | 0/2 | 0.214 | 0/2 | 0.083 | 0/2 | 0.100 | 0.00 | 0.212 |
| Mobile-Agent V2 (qwen) | 0/2 | 0.375 | 0/2 | 0.143 | 1/2 | 0.500 | 1/2 | 0.500 | 0.38 | 0.365 |
| Ours (zhipu) | 0/2 | 0.500 | 0/2 | 0.571 | 1/2 | 0.583 | 1/2 | 0.800 | 0.38 | 0.596 |
| Ours (qwen) | **2/2** | **1.000** | **2/2** | **1.000** | **2/2** | **1.000** | **2/2** | **1.000** | **1.00** | **1.000** |

Table 1: Comparison of RAG completion success rate (SR) and step accuracy (SA) across different methods and tasks. Our method with qwen achieves perfect performance across all metrics.

| Method | Order drink | | Book ticket | | Subscribe | | Play video | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SR | Times | SR | Times | SR | Times | SR | Times | SR | Times |
| zhipu (w/o RAG) | 0/2 | 2 | 1/2 | 3 | 0/2 | 3 | 0/2 | 4 | 0.13 | 3 |
| zhipu (RAG) | 2/2 | 3 | 2/2 | 2 | 2/2 | 4 | 2/2 | 3 | 1.00 | 3 |
| qwen (w/o RAG) | 0/2 | 3 | 1/2 | 2 | 2/2 | 3 | 0/2 | 4 | 0.38 | 3 |
| qwen (RAG) | **2/2** | **2** | **2/2** | **2** | **2/2** | **2** | **2/2** | **2** | **1.00** | **2** |

Table 2: Comparison of success rates (SR) and required interaction times for completing advanced instructions with and without RAG across different LLMs and tasks.

- **Reflection Agent**: The Reflection Agent enhances the robustness of the framework by evaluating the effectiveness of the operations executed by the Decision Agent. By comparing the screen states and perception results before and after an operation, it determines whether the operation is erroneous, ineffective, or correct. This process is defined as:

$$R_t = RA(INS, MU_t, O_{t+1}, S_t, PD_t, S_{t+1}, PD_{t+1}) \tag{12}$$

where $RA$ represents the LLM of the Reflection Agent, $S_{t+1}$ and $PD_{t+1}$ denote the screen state and perception results after the operation, respectively. Based on $R_t$, the Reflection Agent takes the following actions, as shown in the Figure 1(where red indicates erroneous and ineffective operations, and green indicates correct operations):

  - **Erroneous Operation**: Reverts the page to its previous state without recording the operation in the history.
  - **Ineffective Operation**: Maintains the current state without recording the operation in the history.
  - **Correct Operation**: Updates the operation history and progresses the task state accordingly.

  This mechanism ensures that the system avoids following suboptimal operations, continuously refining its decision-making process.

## Instruction Execution Module

The Instruction Execution Module, which primarily executes mobile commands, supports the following mobile commands via ADB:

- **Open app (app name)**: Opens the app named "app name" if the current page is the home page.
- **Tap (x, y)**: Taps on the position with coordinates (x, y).
- **Swipe (x1, y1), (x2, y2)**: Swipes from the position with coordinates (x1, y1) to the position with coordinates (x2, y2).
- **Type (text)**: Types the content of "text" in the input box, assuming the keyboard is active.
- **Home**: Returns to the home page from any page.
- **Stop**: Terminates the entire operation process once the decision agent determines that all requirements have been fulfilled.

The Instruction Execution Module also integrates three tools—text recognition, icon detection, and icon description—to enhance the understanding of screen content. Given a screenshot $S_t$ as input during the $t$-th iteration, the module processes it to extract text and icon information, along with their respective coordinates. This process is represented by:

$$PD_t = IconD(DINO(S_t)) + OCR(S_t) \tag{13}$$

where $PD_t$ is the page description result of the screen at iteration $t$. $OCR$ represents Optical Character Recognition. $IconD$ represents the Icon Description model, which utilizes a visual language model (VLM) to generate the icon description.

# Experiment

To evaluate the performance of our proposed method on real mobile devices, we adopted a dynamic evaluation method. We tested our method on Android system, and we used Android Debug Bridge (ADB) as a tool to operate mobile devices. ADB can simulate all operations of Ours in the operation space. To test the performance of Ours, we designed

four basic tasks, namely: 1. Order milk tea, 2. Book tickets, 3. Click to follow, 4. Play video. For each task, we also designed 10 different high-level instructions. High-level instructions require certain application operation experience to complete. There are a total of 10 high-level instructions of application operation in the experiment.

We designed the following two indicators for dynamic evaluation:

• Success rate (SR): When all requirements of the user instruction are met, the agent is considered to have successfully executed the instruction. The success rate refers to the proportion of successfully executed user instructions.

• Step accuracy (SA): The ratio between the operation steps correctly executed by the agent and the complete correct operation steps.

In addition to comparing our proposed method with the baseline method Mobile-Agent V2, we also compared the performance differences of Ours method when using different visual language models qwen-plus and glm-4v-plus. From the experimental results, it can be seen that compared with Mobile AgentV2 using the base instruction, the advanced instruction using RAG completion has greatly improved the performance of Ours on four types of tasks, and can maintain an amazing success rate under multiple advanced instructions. In addition, there is an amazing gap in the performance of the two domestic visual language models on the agent, and the performance of qwen-plus is far ahead of the glm-4v-plus model. This shows that qwen-plus is a better choice when designing mobile agents.

The above experimental results show that the advanced instruction completed by RAG can significantly improve the performance of the agent model. This is because the UI layouts in different mobile apps are very different, and the content and functions of the apps are different. Simple base instructions are too difficult for today's large models. It can be seen from Table2 that the success rate of the method using a fixed template to complete the base instruction is much lower than that of RAG. RAG can effectively utilize external knowledge bases and reference a large amount of strongly relevant information to provide more in-depth, accurate, and valuable answers, thereby improving the reliability of generated text. In addition, the qwen-plus model can generate advanced instructions with the same performance in fewer times.

## Conclusion

This paper presents a novel approach to enhancing mobile device agents through the integration of RAG with multi-agent architectures. Our experimental results demonstrate several key findings:

First, the incorporation of RAG significantly improves the performance of mobile agents, particularly in complex scenarios requiring intricate navigation and precise content manipulation. The perfect success rates and step accuracy achieved by our method with the Qwen model (SR: 1.00, SA: 1.000) across all four test scenarios validate the effectiveness of our approach.

Second, our comparative analysis reveals that RAG-enhanced agents require fewer interaction steps to complete tasks successfully, with an average of just 2 steps compared to 3-4 steps in baseline methods. This efficiency gain is particularly notable in advanced instruction scenarios, where the combination of RAG and multi-agent architecture enables more precise and direct task execution.

Third, our investigation into different language models shows that the choice of base model significantly impacts performance, with Qwen consistently outperforming zhipu across all metrics. This finding highlights the importance of model selection in developing effective mobile agents.

Looking forward, our work opens several promising directions for future research, including the potential for expanding the range of supported mobile tasks, improving the robustness of RAG integration, and exploring additional ways to optimize agent collaboration. These advancements could further enhance the capabilities of autonomous mobile device operation systems, making them more practical for real-world applications.

## References


Abdelkader, O. 2009. Mobile agent-based applications: A survey. *International Journal of Computer Science and Network Security*, 9(11): 331–339.

Abdelnabi, S.; Gomaa, A.; Sivaprasad, S.; Schönherr, L.; and Fritz, M. 2023. LLM-Deliberation: Evaluating LLMs with Interactive Multi-Agent Negotiation Games.

Al-Jaljouli, R.; and Abawajy, J. 2010. Agents based e-commerce and securing exchanged information. *Pervasive Computing: Innovations in Intelligent Multimedia and Applications*, 383–404.

Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; and Zhou, J. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; and Su, Y. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M.-W. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. *arXiv: Computation and Language,arXiv: Computation and Language*.

Hu, A.; Shi, Y.; Xu, H.; Ye, J.; Ye, Q.; Yan, M.; Li, C.; Qian, Q.; Zhang, J.; and Huang, F. 2024. mplug-paperowl: Scientific diagram analysis with the multimodal large language model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 6929–6938.

Hu, J.; Bhowmick, P.; Jang, I.; Arvin, F.; and Lanzon, A. 2021. A decentralized cluster formation containment framework for multirobot systems. *IEEE Transactions on Robotics*, 37(6): 1936–1955.

Kakiuchi, H.; Kawamura, T.; Shimizu, T.; and Sugahara, K. 2009. An algorithm to determine neighbor nodes for automatic human tracking system. In *2009 IEEE International Conference on Electro/Information Technology*, 96–102. IEEE.

Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2. Minneapolis, Minnesota.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; and Ghanem, B. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36: 51991–52008.

Mukobi, G.; Erlebach, H.; Lauffer, N.; Hammond, L.; Chan, A.; and Clifton, J. 2023. Welfare diplomacy: Benchmarking language model cooperation. *arXiv preprint arXiv:2310.08901*.

Petroni, F.; Lewis, P.; Piktus, A.; Rocktäschel, T.; Wu, Y.; Miller, A.; and Riedel, S. 2020. How Context Affects Language Models' Factual Predictions. *Automated Knowledge Base Construction,Automated Knowledge Base Construction*.

Wang, J.; Xu, H.; Jia, H.; Zhang, X.; Yan, M.; Shen, W.; Zhang, J.; Huang, F.; and Sang, J. 2024a. Mobile-Agent-v2: Mobile Device Operation Assistant with Effective Navigation via Multi-Agent Collaboration. *arXiv preprint arXiv:2406.01014*.

Wang, J.; Xu, H.; Jia, H.; Zhang, X.; Yan, M.; Shen, W.; Zhang, J.; Huang, F.; and Sang, J. 2024b. Mobile-Agent-v2: Mobile Device Operation Assistant with Effective Navigation via Multi-Agent Collaboration. *arXiv preprint arXiv:2406.01014*.

Wang, J.; Xu, H.; Ye, J.; Yan, M.; Shen, W.; Zhang, J.; Huang, F.; and Sang, J. 2024c. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*.

Wang, W.; Lv, Q.; Yu, W.; Hong, W.; Qi, J.; Wang, Y.; Ji, J.; Yang, Z.; Zhao, L.; Song, X.; et al. 2023. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*.

Xu, Z.; Yu, C.; Fang, F.; Wang, Y.; and Wu, Y. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*.

Yao, S.; Chen, H.; Yang, J.; and Narasimhan, K. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhang, C.; Yang, Z.; Liu, J.; Han, Y.; Chen, X.; Huang, Z.; Fu, B.; and Yu, G. 2023. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.