MindSearch 思·索: Mimicking Human Minds Elicits Deep AI Searcher

Zehui Chen[1*], **Kuikun Liu**[2*], **Qiuchen Wang**[1], **Jiangning Liu**[2], **Wenwei Zhang**[2], **Kai Chen**[2†], **Feng Zhao**[1†]

[1] University of Science and Technology of China
[2] Shanghai AI Laboratory

Abstract

Information seeking and integration is a complex cognitive task that consumes enormous time and effort. Search engines reshape the way of seeking information but often fail to align with complex human intentions. Inspired by the remarkable progress of Large Language Models (LLMs), recent works attempt to solve the information-seeking and integration task by combining LLMs and search engines. However, these methods still obtain unsatisfying performance due to three challenges: (1) complex requests often cannot be accurately and completely retrieved by the search engine once; (2) corresponding information to be integrated is spread over multiple web pages along with massive noise; and (3) a large number of web pages with long contents may quickly exceed the maximum context length of LLMs. Inspired by the cognitive process when humans solve these problems, we introduce MindSearch (思·索) to mimic the human minds in web information seeking and integration, which can be instantiated by a simple yet effective LLM-based multi-agent framework consisting of a WebPlanner and WebSearcher. The WebPlanner models the human mind of multi-step information seeking as a dynamic graph construction process: it decomposes the user query into atomic sub-questions as nodes in the graph and progressively extends the graph based on the search result from WebSearcher. Tasked with each sub-question, WebSearcher performs hierarchical information retrieval with search engines and collects valuable information for WebPlanner. The multi-agent design of MindSearch enables the whole framework to seek and integrate information parallelly from larger-scale (*e.g.*, more than 300) web pages in **3 minute**, which is worth **3 hours** of human effort. Based on either GPT-4o or InternLM2.5-7B models, MindSearch demonstrates significant improvement in the response quality in terms of depth and breadth, on both closed-set and open-set QA problems. Besides, responses from MindSearch based on InternLM2.5-7B are preferable by humans to ChatGPT-Web (by GPT-4o) and Perplexity.ai applications, which implies that MindSearch with open-source models can already deliver a competitive solution to the proprietary AI search engine. Code and models are available at https://github.com/InternLM/MindSearch.

1 Introduction

Information seeking and integration is a necessary cognitive process before analysis and decision-making in all walks of life, which usually consumes enormous human efforts and time. The birth of search engines (Brin & Page, 1998; Berkhin, 2005) significantly has reshaped and eased the information-seeking process of human society, however, it still suffers in integrating web information based on complex human intentions. Recently, Large Language Models (LLMs) have showcased

Project Page: https://mindsearch.netlify.app
[*] Equal Contribution. Alphabetical order [†] Corresponding author

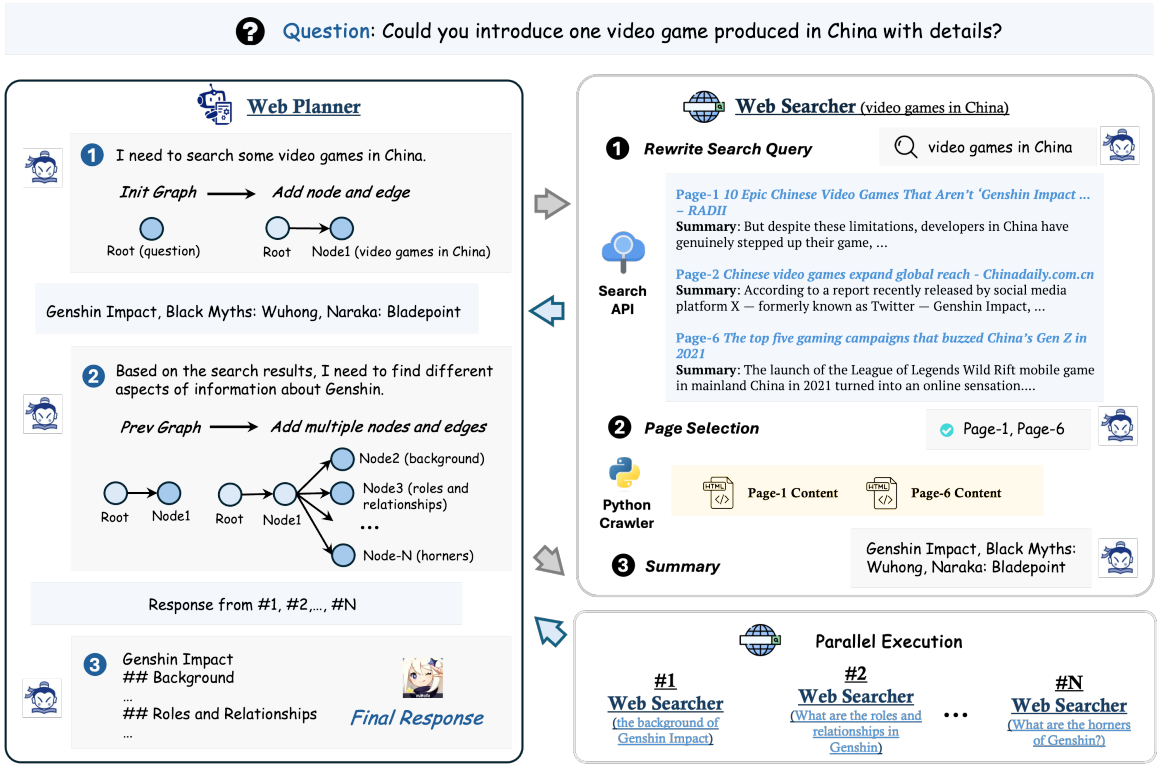

Figure 1: **The overall framework of MindSearch.** It consists of two main ingredients: WebPlanner and WebSearcher. WebPlanner acts as a high-level planner, orchestrating the reasoning steps and multiple WebSearchers. WebSearcher conducts fine-grained web searches and summarizes valuable information back to the planner, formalizing a simple yet effective multi-agent framework.

remarkable progress in reasoning, language understanding, and information integration across a variety of domains (Achiam et al., 2023; Team et al., 2024; Touvron et al., 2023; Cai et al., 2024), whereas they struggling to deliver accurate knowledge in responses (Ji et al., 2023; Gu et al., 2024).

The complementary advantages of LLMs and search engines highlights a compelling opportunity for their combination, where the reasoning prowess of LLMs can be complemented by the extensive web information accessible via search engines, potentially revolutionizing the solution of web information seeking and integration. Previous works (Asai et al., 2023; Chan et al., 2024) simply treat the information seeking and integration task as a vanilla retrieve-augmented generation (RAG) task (Chen et al., 2017; Lin et al., 2023). Such a formulation, although straightforward, often results in sub-optimal performance due to a superficial engagement with the depth and complexity of web-based information retrieval, facing three major challenges for more complex user queries:

(1) Real-world problems often require in-depth analysis and proper decomposition of the question before retrieving the related information, which cannot be done by retrieving web pages at once.
(2) The overwhelming volume of searched web pages and massive information noise pose great challenges for LLMs for efficient information integration.
(3) The rapid proliferation of web search content can quickly exceed the maximum context length of LLMs, which further decreases the information integration performance.

Inspired by how human experts solve real-world problems, we propose MindSearch 思·索[1], a simple yet effective LLM-based multi-agent framework, which consists of a WebPlanner (mimic human minds for problem reasoning) and multiple WebSearcher (manage the information seeking

[1]The Chinese name '思·索' means thinking as human and exploring by searching

process). Given a user query, the WebPlanner first decomposes the query into multiple atomic sub-questions that can be parallelly solved and dispatches them to the respective WebSearcher. To further enhance the reasoning ability, WebPlanner models the complex problem-solving process as an iterative graph construction: by predefining a list of standard code interfaces related to the construction of the topological mind graph, WebPlanner is able to progressively decompose the question into sequential/parallel sub-problems by adding nodes/edges in the graph via Python code generation. Meanwhile, the WebSearcher, tasked with each sub-problem, employs a hierarchical retrieval process to extract valuable data for LLMs, which significantly improves the information aggregation efficiency facing massive search pages. By distributing different aspects of the reasoning and retrieval process to specialized agents, MindSearch effectively reduces the load on each single agent, facilitating a more robust handling of long contexts. It seamlessly bridges the gap between the raw data retrieval capabilities of search engines and the context-understanding power of LLMs.

To validate the effectiveness of MindSearch, we conducted extensive evaluations on both closed-set and open-set question-answering (QA) problems using GPT-4o and InternLM2.5-7B-Chat models. The experimental results demonstrate a substantial improvement in response quality, both in the dimensions of depth and breadth. Moreover, comparative analysis shows that the responses of MindSearch are more preferred by human evaluators over those from existing applications like ChatGPT-Web (based on GPT-4o) and Perplexity Pro. These findings suggest that MindSearch with open-source LLMs can offer a highly competitive solution for AI-driven search engines.

2 MindSearch

To effectively synergize the web information retrieval capabilities of search engines and the reasoning and information integration capability of LLMs, MindSearch consists of a WebPlanner and a group of WebSearchers (Fig. 1). WebPlanner first decomposes the user question into sequential or parallel search tasks via reasoning on the graph and determines the next step based on the search feedback (Sec. 2.1). WebSearcher is tasked with the query and performs hierarchical information retrieval on the Internet to answer sub-questions (Sec. 2.2). We also discuss the context management within the scope of the multi-agent design in Sec. 2.3.

2.1 WebPlanner: Planning via Graph Construction

The WebPlanner functions as a high-level planner, orchestrating the reasoning steps and coordinating other agents. However, we observed that merely prompting the LLM to plan the entire data workflow architecture does not yield satisfactory performance. Specifically, current LLMs struggle with decomposing complex questions and understanding their topological relationships, leading to coarse-grained search queries. This approach underutilizes the potential of LLMs to serve as intermediaries between humans and search engines, transforming human intentions into step-by-step search tasks and delivering accurate responses.

To enhance the capability of LLM in addressing complex questions, we model the problem-solving process as a directed acyclic graph (DAG). Given a user question Q, the solution trajectory is represented as $G(Q) = \langle V, E \rangle$, where V is a set of nodes v, each representing an independent web search, including an auxiliary START node (the initial question) and an END node (the final answer). E represents directed edges indicating the reasoning topological relationships between nodes (search contents). This DAG formalism captures the complexity of finding the optimal execution path, providing a more formal and intuitive representation for LLMs.

Leveraging the superior performance of current LLMs on code tasks (Guo et al., 2024; Roziere et al., 2023), we explicitly prompt the model to interact with the graph through code writing. To achieve this, we predefined atomic code functions to add nodes or edges to the graph (Step 1 and 2 in Figure 2). At each turn, the LLM first reads the entire dialogue, including previously generated code and web search results, then outputs thoughts and new code for reasoning on the mind graph, which is executed with a Python interpreter. During execution, once a node is added to the reasoning graph, it invokes a WebSearcher to execute the search process and summarize the information. Since

Figure 2: **A concrete example of how WebPlanner addresses the question step by step via planning as coding.** During each turn, WebPlanner outputs a series of thoughts along with the generated code. The code will be executed and yield the search results to the planner. At the last turn, the WebPlanner directly provides the final response without any code generation.

the newly added nodes are only dependent on nodes generated in previous steps, we can parallel them to achieve a much faster information aggregation speed. When all information is collected, the planner produces the final response by adding the end node (Step 3 in Figure 2).

By integrating with the Python interpreter, WebPlanner interacts with the graph through unified code actions, dynamically constructing the reasoning path. This "code as planning" process enables the LLM to fully leverage its superior code generation ability, benefiting control and data flow in long-context scenarios and leading to better performance in solving complex problems.

2.2 WebSearcher: Web Browsing with Hierarchical Retrieval

WebSearcher acts as a sophisticated RAG (Retrieve-and-Generate) agent with internet access, summarizing valuable responses based on search results (Figure 3). Due to the massive content available on the web, it is challenging for LLMs to process all related pages within a limited context length (*e.g.* 8K tokens). To address this, we employ a straightforward coarse-to-fine selection strategy. Initially, the LLM generates several similar queries based on the assigned questions from the WebPlanner to broaden the search content and thus improve the recall of relevant information. These queries are then executed through various search APIs, such as Google, Bing, and DuckDuckGo, which return key contents including web URLs, titles, and summaries. The search results are automatically merged based on the web URLs, and the LLM is prompted to select the most valuable pages for detailed reading. The full content of the selected web URLs is then added to the input of LLM. After reading these results, the LLM generates a response to answer the original question based on the search results. This hierarchical retrieval approach significantly reduces the difficulty of navigating massive web pages and allows to efficiently extract highly relevant information with in-depth details.

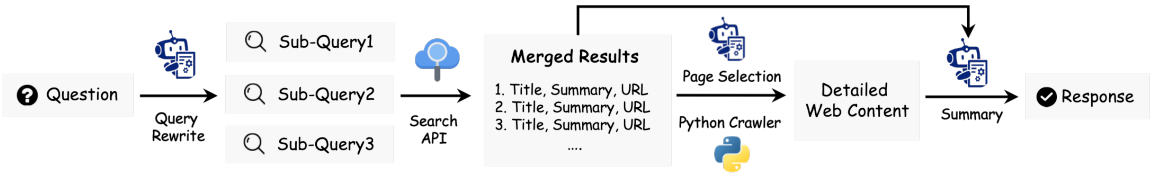

Figure 3: **A detailed working pipeline of WebSearcher.** It comprises at most 4 steps: query rewrite, search content aggregation, detailed page selection, and final summarization.

2.3 LLM Context Management in MindSearch

MindSearch provides a simple multi-agent solution to complex information seeking and integration with search engines. Such a paradigm also naturally enables long-context management among different agents, which improves the overall efficiency of the framework, especially under circumstances that require the model to quickly read plenty of web pages. Since the WebPlanner distributes the search tasks into separate search agents and only relies on the searched results from WebSearcher, WebPlanner can purely focus on the decomposition and analysis of the user question without being distracted by the over-length web search results. Meanwhile, each WebSearcher only needs to search contents for its tasked sub-query, without distraction from other contents. Thanks to the explicit role distribution, MindSearch greatly reduces context computation during the whole process, delivering an efficient context management solution to long-context tasks for LLM. Such a multi-agent framework also provides a straightforward and simple long-context task construction pipeline for training single LLMs, which is also observed in (Team, 2024). Eventually, MindSearch collects and integrates related information from more than 300 pages in less than 3 minute, which could take human experts about 3 hours to finish a similar cognitive workload.

Due to the explicit context state transfer across multiple agents, we need to carefully handle the context during the whole workflow. We empirically find simply focusing the decomposed query from the Planner may lose useful information during the information collection phase due to the local receptive field inside the search agent. How to effectively handle the context between multiple agents is non-trivial. We find that the constructed topological relations through the directed graph edges help us easily handle the context across different agents. More specifically, we simply prefix the response from its father node as well as the root node when executing each search agent. Therefore, each WebSearcher can effectively focus on its sub-task without losing the previous related context as well as the final goal.

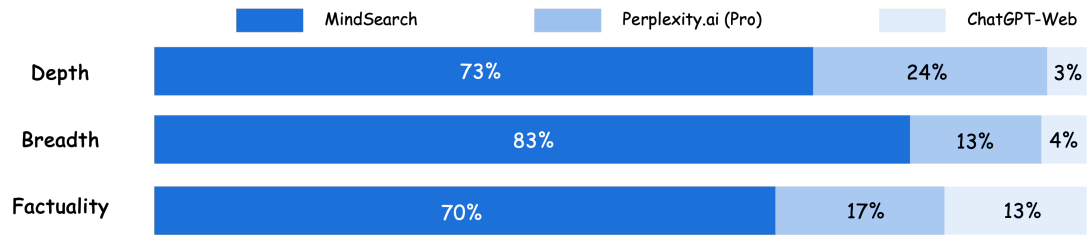

Figure 4: **Subjective evaluation results judged by human experts on open-set QA questions.** MindSearch outperforms ChatGPT-Web and Perplexity.ai Pro by a large margin in terms of depth, breadth, and facticity.

3 Experiments

We evaluate MindSearch on two primary categories of Question Answering (QA) tasks: closed-set QA and open-set QA, which reflects both the subjective and objective judgment of MindSearch. For a fair comparison, all models only have access to the Internet through BING search API, and no extra reference sources are considered.

3.1 Open-Set QA

3.1.1 Implementation Details

To better gauge the utility and search performance, we carefully curate 100 real-world human queries and collect responses from MindSearch (InternLM2.5-7b-chat (Cai et al., 2024)), Perplexity.ai (its Pro version) , and ChatGPT with search plugin Achiam et al. (2023). We ask five human experts to manually select their preferred responses, in terms of the following three aspects:

- **Depth**: Depth refers to the thoroughness and profundity of an answer. A response with depth provides detailed information and delves into the intricacies of a question.
- **Breadth**: Breadth pertains to the scope and diversity covered by an answer. A response with breadth touches on various aspects of the question or multiple related fields, offering different perspectives or solutions.
- **Factuality**: Factuality is the degree to which an answer is accurate and fact-based. It should be grounded in reliable data and information, avoiding errors or misleading content, and ensuring the truthfulness and credibility of the information provided.

The final results are determined based on major votes. During the evaluation, the correspondence between the response and its method is invisible to the evaluators to guarantee fairness.

3.1.2 Results and Analysis

The evaluation results are depicted in Figure 4 and we also provide quantitative results in Figure 5. From Figure 4, we can observe an absolute improvement in terms of the depth and breadth of the model response, which validates the superiority of our proposed WebPlanner. By integrating code into the DAG construction phase, LLM is able to progressively decompose the complex problem into executable queries while balancing the tradeoff between time efficiency and the exploration of the search space. Besides, MindSearch goes through more fine-grained search topics about the question, therefore providing more compact and detailed responses compared to other models. However, MindSearch does not yield better performance in terms of facticity. We suspect that more detailed search results may distract the concentration of the model on the initial problem, especially when LLM holds incomplete long-context capability. Therefore, a natural future work of MindSearch is to alleviate the hallucination issues during the web browsing process.

Figure 5: **Solution trajectory comparison between MindSearch and Perplexity.ai (Pro) on the same question.** MindSearch provides more detailed and proper responses thanks to its fine-grained searches.

Table 1: **Performance comparison on various closed-set QA tasks**. We select two representative LLMs: GPT-4o (close-sourced) and InternLM2.5-7b-chat (open-sourced).

Model	Bamboogle	Musique			HotpotQA			AVG
		2-hop	3-hop	4-hop	Easy	Medium	Hard	
Closed-Souced LLM (GPT-4o)								
w/o Search Engine	70.4	54.0	22.0	**20.0**	73.0	69.0	66.0	53.5
ReAct Search	75.2	48.0	25.0	13.3	**81.0**	73.0	70.0	55.1
MindSearch	**76.8**	**60.0**	**35.0**	14.6	80.0	**74.0**	**78.0**	**59.8**
Open-Sourced LLM (InternLM2.5-7b-chat)								
w/o Search Engine	34.0	28.0	10.0	17.3	47.0	26.0	40.0	28.9
ReAct Search	55.2	38.0	17.0	16.0	69.0	56.0	49.0	42.9
MindSearch	**67.8**	**46.0**	**20.0**	**18.6**	**69.0**	**66.0**	**57.0**	**49.2**

In addition to quantitative results, we also provide a qualitative response comparison between Perplexity.ai (Pro) and MindSearch to deliver an intuitive understanding of their performance. From Figure 5, we can observe that MindSearch yields more concrete and detailed responses. We empirically find that our better responses can be attributed to the proper planning search paths compared to Perplexity.ai, which also indicates that how to decompose the human intention is the key step to the final problem.

3.2 Closed-Set QA

3.2.1 Implementation Details

We extensively evaluate our approach on a wide range of closed-set QA tasks, including Bamboogle (Press et al., 2022), Musique (Trivedi et al., 2022), and HotpotQA (Yang et al., 2018). To further validate the generalization of our approach, we select both closed-source LLM (GPT-4o) and open-source LLM (InternLM2.5-7b-chat) as our LLM backend. Since our approach adopts a zero-shot experimental setting, we utilize a subjective LLM evaluator (GPT4-o) to gauge the correctness of HotpotQA.

3.2.2 Results and Analysis

In Table 1, we compare our approach with two straight-forward baselines: raw LLM without search engines (w/o Search Engine), and simply treating search engines as an external tool and adopting a ReAct-style interaction (ReAct Search). We can conclude that MindSearch significantly outperforms its vanilla baselines by a large margin, validating the effectiveness of the proposed method. These advantages are amplified when transferring from closed-sourced LLMs to open-sourced LLMs, which further proves that MindSeach provides a simple approach to enhance weak LLMs with broader knowledge and alleviate hallucination issues.

4 Related Work

4.1 Tool Utilization with LLM

The Tool Learning framework empowers LLMs to seamlessly integrate with a variety of tools (Qin et al., 2023; Hao et al., 2024; Zhuang et al., 2024; Chen et al., 2023), such as search engines (Chan et al., 2024), databases (Parisi et al., 2022), and APIs (Li et al., 2023; Patil et al., 2023), offering dynamic solutions to complex problems. This integration is not only beneficial for enhancing the interpretability and trustworthiness of LLMs but also for improving their robustness and adaptability across diverse tasks, including reducing hallucinations (Ji et al., 2024), code generation (Gou et al., 2023), and question answering (Chen et al., 2024). Recent research has focused on enhancing the tool integration component of Tool Learning systems. Works such as (Huang et al., 2023; Shen et al., 2023; Schick et al., 2024) have concentrated on improving the retrieval mechanisms, ensuring that LLMs can access the most pertinent tools for a given task. Other studies, like (Qian et al., 2023; Yuan et al., 2023), aim at refining the LLMs' ability to effectively utilize the retrieved information, optimizing the reading and comprehension processes within the framework.

4.2 RAG with LLM

RAG demonstrates significant advantages in addressing knowledge-intensive problems, especially in open-domain scenarios with the integration of search engines (Chen et al., 2017; Li et al., 2017). RAG allows LLMs to integrate with the retriever, providing timely information and offering effective solutions. Moreover, RAG is also applied in various tasks such as reducing hallucinations (Shuster et al., 2021; Gu et al., 2024), code generation (Zhou et al., 2022), and question answering (Lewis et al., 2020). Recently, some work (Karpukhin et al., 2020; Xiong et al., 2020; Qu et al., 2020) focuses on enhancing the retrieval component of RAG systems, while others (Izacard & Grave, 2020; Borgeaud et al., 2022; Yu et al., 2021; Lei et al., 2017) enhances the language model's ability as a reader to optimize the framework.

With the advancement of LLM capabilities, some researchers have begun to reoptimize frameworks and redesign methodologies for model training. SAIL (Luo et al., 2023) trains LLM to be more focused on credible and informative search results. Self-RAG (Asai et al., 2023) enables LMMs to independently fetch, introspect, and augment their text generation capabilities. RQ-RAG (Chan et al., 2024) enhances query formulation by learning to refine queries through an iterative process. Our work integrates web search capabilities into LLMs, enhancing response quality by retrieving valuable information from the Internet.

4.3 Web Agents

Web automation agents have evolved from question-answering tools to sophisticated systems capable of complex web interactions. Early models like WebGPT (Nakano et al., 2021) and WebGLM (Liu et al., 2023) primarily addressed QA tasks, while recent advancements have shifted towards more dynamic operations (Yao et al., 2022; He et al., 2024). MindAct (Deng et al., 2024) and WebAgent (Gur et al., 2023) represent this progression, with the latter showing exceptional web navigation despite deployment challenges due to its size. AutoWebGLM (Lai et al., 2024) offers a

practical alternative with robust capabilities and a more compact model size. The incorporation of reinforcement learning (Bai et al., 2024) and behavior cloning techniques (Zheng et al., 2024; Patel et al., 2024) paves the way for even more autonomous and efficient web automation, moving the field towards scalable and versatile solutions for real-world applications. This paper mainly focuses more on the web information-seeking and integration task with search engines instead of web browsing, and solves the main challenges with a multi-agent framework.

5 Conclusion

This paper introduces MindSearch, a novel LLM-based multi-agent framework for complex web information-seeking and integration tasks, by more comprehensively leveraging the strengths of both search engines and LLMs. MindSearch conducts effective and sufficient decomposition of complex queries followed by hierarchical information retrieval to improve the precision and recall of the retrieved relevant web information, by modeling the problem-solving process as an iterative graph construction. The multi-agent design distributes the cognitive load among specialized agents, facilitating robust handling of complex and lengthy contexts. Extensive evaluations on closed-set and open-set QA problems using GPT-4o and InternLM2.5-7B models demonstrated significant advantages in the response quality of MindSearch. The results that human evaluators preferred the responses from MindSearch over those from ChatGPT-Web and Perplexity.ai indicate its competitive edge in AI-driven search solutions. We wish this work pave the way for future research on multi-agent framework for solving human-level complex cognitive tasks.

6 Acknowledgement

We would like to express our sincere gratitude to Jiaye Ge for her outstanding technical insights, product design skills, and coordination abilities. We also thank Zhongying Tu, Ying Zhao, Fang Fang, and Yiting Wang for their efficient execution in developing the project demo. Our gratitude extends to Xingyuan Liu, Shuaike Li, Zike Pan, Weijia Song, and Yuzhe Gu for their efforts in project suggestion.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*, 2024.

Pavel Berkhin. A survey on pagerank computing. *Internet mathematics*, 2(1):73–120, 2005.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*, 2024.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, et al. T-eval: Evaluating the tool utilization capability step by step. *arXiv preprint arXiv:2312.14033*, 2023.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*, 2024.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

Yuzhe Gu, Ziwei Ji, Wenwei Zhang, Chengqi Lyu, Dahua Lin, and Kai Chen. Anah-v2: Scaling analytical hallucination annotation of large language models. *arXiv preprint arXiv:2407.04693*, 2024.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*, 2023.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Ziwei Ji, Yuzhe Gu, Wenwei Zhang, Chengqi Lyu, Dahua Lin, and Kai Chen. Anah: Analytical annotation of hallucinations in large language models. *arXiv preprint arXiv:2405.20315*, 2024.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. Autowebglm: Bootstrap and reinforce a large language model-based web navigating agent. *arXiv preprint arXiv:2404.03648*, 2024.

Xiangyu Lei, Guilin Zhang, Shuaijun Li, Huihuan Qian, and Yangsheng Xu. Dual-spring agv shock absorption system design: Dynamic analysis and simulations. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1068–1074. IEEE, 2017.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

Shuaijun Li, Guilin Zhang, Xiangyu Lei, Xiao Yu, Huihuan Qian, and Yangsheng Xu. Trajectory tracking control of a unicycle-type mobile robot with a new planning algorithm. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 780–786. IEEE, 2017.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*, 2023.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. Webglm: Towards an efficient web-enhanced question answering system with human preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4549–4560, 2023.

Hongyin Luo, Yung-Sung Chuang, Yuan Gong, Tianhua Zhang, Yoon Kim, Xixin Wu, Danny Fox, Helen Meng, and James Glass. Sail: Search-augmented instruction learning. *arXiv preprint arXiv:2305.15225*, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks. *arXiv preprint arXiv:2405.20309*, 2024.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Cheng Qian, Chi Han, Yi R Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. *arXiv preprint arXiv:2305.14318*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*, 2020.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. *arXiv preprint arXiv:2311.18760*, 2023.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Qwen Team. Generalizing an llm from 8k to 1m context using qwen-agent, May 2024. URL https://qwenlm.github.io/blog/qwen-agent-2405/.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *arXiv preprint arXiv:2110.04330*, 2021.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. *arXiv preprint arXiv:2309.17428*, 2023.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Shuyan Zhou, Uri Alon, Frank F Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. Docprompting: Generating code by retrieving the docs. *arXiv preprint arXiv:2207.05987*, 2022.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36, 2024.