Agentic Information Retrieval

Weinan Zhang, Junwei Liao, Ning Li, Kounianhua Du
Shanghai Jiao Tong University
wnzhang@sjtu.edu.cn

Abstract

What will information entry look like in the next generation of digital products? Since the 1970s, user access to relevant information has relied on domain-specific architectures of information retrieval (IR). Over the past two decades, the advent of modern IR systems, including web search engines and personalized recommender systems, has greatly improved the efficiency of retrieving relevant information from vast data corpora. However, the core paradigm of these IR systems remains largely unchanged, relying on filtering a predefined set of candidate items. Since 2022, breakthroughs in large language models (LLMs) have begun transforming how information is accessed, establishing a new technical paradigm. In this position paper, we introduce Agentic Information Retrieval (Agentic IR), a novel IR paradigm shaped by the capabilities of LLM agents. Agentic IR expands the scope of accessible tasks and leverages a suite of new techniques to redefine information retrieval. We discuss three types of cutting-edge applications of agentic IR and the challenges faced. We propose that agentic IR holds promise for generating innovative applications, potentially becoming a central information entry point in future digital ecosystems.

1 The Trends of IR

Information retrieval (IR) refers to the tasks or techniques of finding information items to match the user's needs from a large corpus. Broadly speaking, there exists a wide range of real-world IR applications, including web search, item recommendation, online advertising, online travel agency, online shopping, online food delivery, etc. [Singhal et al., 2001, Wang et al., 2017].

As an automated information filtering system, the traditional IR generally employs a specialized architecture to retrieve, rank, and select the information item according to the query. Web search engines, a remarkable example of IR, employ an inverted index system to maintain the posting list of documents for each term (or word). Given a query, the candidate documents containing the query terms are retrieved using the inverted index, and then ranked using a refined or learned scoring function. Finally, the top-ranked documents are presented on the search engine result page (SERP) [Baeza-Yates et al., 1999]. Personalized recommender systems, another major IR example, typically involve retrieval, pre-ranking (optional), ranking, and re-ranking stages to perform a funnel-like filtering of the items and finally present the re-ranked top items to the user [Qin et al., 2022].

Despite their technical and business success, the above IR architectures need to be predefined from the very beginning of the application, and, once built, remain unchanged throughout the lifetime of the IR systems, so as to the information flow during each IR process. Based on the fixed information flow of the predefined architectures, it is difficult to perform interactive or complex IR tasks (with multiple-step reasoning and actions). For example, on a search engine, the user needs to carefully refine the search keywords to iteratively get the updated SERPs to find the webpage he is seeking; on an e-commerce recommender system, the user has no effective way to change the recommended item list for him rapidly. Moreover, the returned items are as they are — there is no way to manipulate the information items across the IR process — which keeps the IR scenarios simple and limited.

Since the beginning of 2023, with the success of large language models (LLMs) such as ChatGPT, Claude, and GPT4, the generative question-answering applications become much popular. Furthermore, by wrapping an LLM as an AI agent to interact with the environment, it becomes feasible to let the agent to perform multiple (or/and multi-round) reasoning-action steps to accomplish completed tasks. In addition, a variety of tools, including search engines, calculators, weather forecasters, databases, can be accessed by the agents via APIs, which largely enhance the task-solving abilities of the AI agents. With such a background, it is good timing to think about the next-generation IR architectures in the era of LLM-driven AI agents.

In this position paper, we introduce the concept of agentic information retrieval (Agentic IR), a novel paradigm of IR techniques that could serve as the key architecture form of the next-generation information retrieval. In general, agentic IR differs from traditional IR in the following aspects.

- **Task scope.** Agentic IR deals with a much wider scope of tasks. For agentic IR, the user shows an expected information state, while the agent takes actions to reach the user to that information state. As such, the traditional IR is a special case of agentic IR, i.e., to present the relevant information items to the user.

- **Architecture.** Unlike the fixed domain-specific architecture for the served scenario in traditional IR, agentic IR generically employs a unified architecture, i.e., the AI agent, to different scenarios, as illustrated in Figure 1. The key difference between the architectures of agentic IR and traditional IR is that the agent solves the problem with the recurrence architecture of observation, reasoning, and action across multiple steps, while the traditional IR tries to solve the problem in one interaction step with a large architecture.

- **Key methods.** The key methods of agentic IR include prompt engineering, retrieval-augmented generation, fine-tuning with supervised and reinforcement learning, multi-agent systems, which are essentially different from those of traditional IR, such as indexing, retrieval methods, scoring function, learning to rank, and pseudo relevance feedback.

The remaining part of this paper is organized as follows. In Section 2, we formally present agentic IR including task formulation, architecture form, and some key methods. Then, in Section 3, we describe several emergent representative applications of agentic IR, namely life assistant, business assistant, and coding assistant. Later we discuss the current challenges of agentic IR in Section 4. Finally we conclude this paper in Section 5.
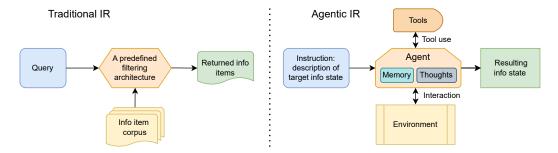

Figure 1: The paradigms of traditional IR vs. agentic IR.

2 Agentic IR

2.1 Task Formulation

Let s_* denote the user's target information state, which can be a desired document in a document retrieval task, a satisfying answer in a QA task, or an accomplished accurate order for an online shopping task, etc. Let $x(s_*)$ denote the instruction text provided by the user describing his target information state s_*.

Let $\pi(a_t|x(s_t))$ denote the policy of the agent, where s_t denotes the information state at step t, $x(s_t)$ denotes the corresponding transformed text input to the policy, and a_t denotes the corresponding action a_t taken by the agent. The action is then delivered to the environment and environment

accordingly transits to the next state s_{t+1} conditioned on the current state s_t and the action taken a_t based on its dynamics $p(s_{t+1}|s_t, a_t)$.

When the environment reaches a resulting information state s_T at a certain step T, the agentic IR process terminates. Then, the corresponding success or failure result can be calculated by a predefined rule (or verifier) $r(s_*, s_T)$.

In such a framework, the target of an agentic IR algorithm is to acquire an optimal agent policy that maximizes the expectation of success as

$$\max_\pi \ \mathbb{E}_{s_*}[r(s_*, s_T)]$$
$$\text{such that } s_{t+1} \sim p(\cdot|s_t, a_t), a_t \sim \pi(\cdot|x(s_t)), \text{ for } t = 1 \dots T - 1. \tag{1}$$

2.2 Architecture

The agent policy $\pi(a_t|x(s_t))$, as shown in the central module of the agentic IR subplot in Figure 1, takes the user's language instruction as input, then interacts with the environment with single or multiple turns, and finally reaches the resulting information state.

The inner-architecture modules of the agent generally include memory and thought. Generally, the memory means the the log history, experience that can be stored in the disk, while the thought is the information stored in the context window of the LLM. Additionally, there is a pool of external tools for the agent to call [Patil et al., 2023, Lin et al., 2024]. A tool can be regarded as a function (with input arguments) that cannot be replaced by a neural net model, such as web search engine, relational DB, real-time weather app, calculator, etc.

As such, the textual description of the information state of the agent at step t can be written as

$$x(s_t) = g(s_t, h_t, \text{MEM}, \text{THT}, \text{TOOL}), \tag{2}$$

where MEM, THT, and TOOL denote three functions that update memory, manipulate thoughts, and call tools, respectively. g denotes a composite function based on the above three functions, takes the current state s_t and the memory h_t as the raw input, and outputs the intermediate representation of the state s_t, which is normally the language prompt $x(s_t)$ to further feed into the LLM agent. Note that the specific design of g directly determines the agent, along with the used LLM, which is still underexplored.

With such a framework, the specific architecture of the agent can be instantiated by creating a direct acyclic graph over the three functions, which can be implemented by changing the prompts of the LLMs. A previous study of similar architectures is provided by Christianos et al. [2023]. As a result, the architecture of the agentic IR can be built in a unified way.

2.3 Key Methods

Given the above task formulation and agent architecture, the key methods to improve the performance of agentic IR, namely Eq. (1), would include but not be limited to prompt engineering, retrieval-augmented generation, reflection, supervised fine-tuning, preference learning, reinforcement (learning) fine-tuning, complex reasoning, reward modeling, multi-agent systems, etc.

- **Prompt engineering.** Prompts are the task-based language token input to the LLM to enable its ability for the task [Liu et al., 2023]. For an LLM, the prompt is a human-controllable way to set its hidden state in comparison to the model parameters, including the chain-of-thought prompting.

- **Retrieval-augmented generation (RAG).** The task-specifically retrieved demonstrations play a crucial role in LLM-based applications. In agentic IR, the retrieved demonstrations can be on the action level or the thought level [Zhou et al., 2024].

- **Reflection.** The agent may use the failure or suboptimal results of its interactions with the environment to update its thoughts, so as to make further attempts to refine its actions and thus the resulting information state [Shinn et al., 2024].

- **Supervised fine-tuning (SFT).** As a basic methods for fine-tuning LLMs, SFT can be seamlessly adapted to agentic IR tasks, where the successful historic trajectories are used as the training data with each step of action or the output of each inner function as the label to fit. SFT corresponds to

the behavioral cloning imitation learning methods in reinforcement learning. Despite simplicity, SFT does not directly optimize the objective (1).

- **Preference learning.** One step further based on SFT, fine-tuning LLMs based on a preference objective over a pair of outputs may improve the performance of the agentic IR models [Rafailov et al., 2024]. Note that such methods are to some extent similar to the pairwise learning to rank techniques in traditional IR [Burges et al., 2005].

- **Reinforcement fine-tuning (RFT).** Regarding the environment as a Markov decision process, reinforcement learning methods, including PPO [Schulman et al., 2017] and AlphaZero [Silver et al., 2018], directly optimize the objective (1), given the reward signal from the environment or human feedbacks (RLHF) [Ouyang et al., 2022]. Compared with SFT and preference tuning, RFT usually requests larger computational resources to explore the environment, accumulate experience data, and update the model parameters [Christianos et al., 2023].

- **Complex reasoning.** For non-trivial tasks, the agent needs to perform task planning and complex reasoning before taking actions. The recent success of OpenAI o1 [OpenAI, 2014] indicates the great potential of a strong reasoner for improving the agent's task-solving performance. In contrast, RAG can be regarded as a case-based reasoning [Guo et al., 2024].

- **Reward modeling.** As a judge of the resulting information state or intermediate states in the process, the reward function modeling is crucial to enable RFT or search-based decoding techniques in complex agentic IR tasks. Referring to recent advances in solving math reasoning problems, the outcome reward models and process reward models are essential modules to yield high-performance math agents [Uesato et al., 2022, Luo et al., 2024].

- **Multi-agent systems (MAS).** A MAS contains multiple homogeneous or heterogeneous agents, each of which could be equipped with a special role or resources. With proper mechanisms, the team of agents manages to coordinate to achieve remarkable collective intelligence [Chen et al., 2023, Li et al., 2024a].

3 Application Scenarios and Case Studies

In this section, we briefly discuss three types of applications with case studies of agent IR, i.e., life assistant, business assistant, and coding assistant. As their names are, the agent IR would play more like an assistant for users with a certain level of autonomy. The traditional IR, by contrast, is like a tool to call in agent IR, which is non-autonomous.

3.1 Life Assistant

In recent years, life assistants have evolved from simple voice-activated tools into sophisticated systems capable of supporting users across a wide array of daily tasks. At the core of this transformation is a significant advancement in IR technologies. Agentic IR empowers these assistants not only to gather and deliver information but also to proactively support planning and decision-making with a deep understanding of the user's needs, context, and preferences. This shift enables life assistants to act as active, autonomous agents that adapt seamlessly to a user's lifestyle, offering guidance and taking actions in real time.

Agentic IR capabilities are already present in major products like Apple's ecosystem, where Apple Intelligence[1] powers advanced assistant features across devices such as iPhone, iPad, and Mac [Apple, 2024]. Apple Intelligence enhances user experience by seamlessly integrating with apps, services, and smart devices, embodying the proactive and contextual characteristics of agentic IR. Beyond Apple, other life assistants, such as Google Assistant[2], Oppo Breeno, and Huawei Celia[3], operate across diverse platforms, including smartphones, smart home devices, and wearables. These assistants empower users with convenient control over both digital and physical environments, enabling them to make informed plans and adjustments anytime, anywhere [Li et al., 2024b].

Consider the following scenario: Jane is a busy professional who uses a life assistant integrated into her smartphone and other devices. Agentic IR allows her assistant to anticipate her needs, gather

---

[1]https://www.apple.com/apple-intelligence/
[2]https://assistant.google.com/
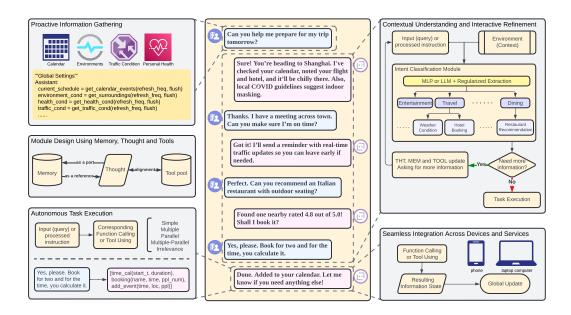[3]https://consumer.huawei.com/en/emui/celia/

Figure 2: Illustration of agentic IR in life assistant scenarios.

relevant information, and autonomously perform tasks without constant user intervention. Figure 2 illustrate how the various features of agentic IR play out in Jane's daily life.

Proactive information gathering and state transition. Each step taken by the assistant is a transition from one intermediate information state to another, achieved through actions (a_t) that are dynamically informed by Jane's requests and surrounding context. If Jane's meeting is across town during rush hour, her assistant identifies the information state s_t that includes traffic conditions and suggests an earlier departure time. This suggestion is the next intermediate state for that interaction, precisely capturing the assistant's proactive adaptation to Jane's immediate needs. Each conversation round refines the assistant's understanding, moving incrementally closer to the target information state s_*.

Modular design using memory, thought, and tools. In reaching each intermediate information state, the assistant employs a modular structure with memory (MEM) for context, manipulate thought (THT) for processing Jane's preferences, and tools (TOOL) for external information sources like real-time traffic or weather data. For example, before Jane's business trip, the assistant integrates data from her calendar (via MEM) with travel conditions (accessed via TOOL). Each round of action culminates in a specific intermediate information state s_t, gradually advancing toward the target information state and bringing Jane closer to her goal, such as having "all travel preparations confirmed".

Adaptation through contextual understanding and interactive refinement. Agentic IR enables the assistant to adapt seamlessly by refining its actions based on both explicit queries and passive contextual cues, allowing continuous updates to the information state s_t without needing repeated user input. For example, when Jane arrives at a grocery store, the assistant references her shopping list based on location; if a previous item is missing, it suggests adding it without requiring her to ask. Similarly, if Jane requests a restaurant recommendation, the assistant interactively clarifies her preferences, such as cuisine and view, to better align with her needs. By adapting to both explicit responses and situational context, the assistant effectively progresses toward the target information state, underscoring agentic IR's capacity for flexible, accurate assistance with minimal input.

Autonomous task execution and final information states. Beyond simply gathering information, agentic IR enables the assistant to autonomously execute tasks, such as booking a dinner reservation or setting reminders. When Jane's assistant books a restaurant, it concludes that conversation round in an information state where the booking is completed and confirmed in her calendar. This autonomous capability frees Jane from cognitive overhead, allowing her to focus on higher-priority tasks as the assistant seamlessly progresses from one target information state to the next.
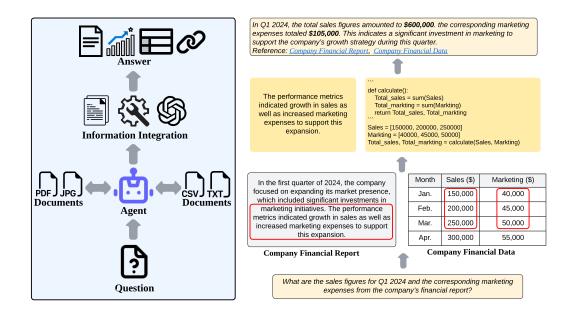
5

Figure 3: Illustration of agentic IR in business assistant scenarios.

Seamless integration across devices and services. The assistant's ability to unify various devices and services allows it to operate as a fully integrated agentic IR system. By combining memory and real-time inputs across her smart home and calendar applications, Jane's assistant ensures that her arrival time and thermostat settings are in harmony. This synchronization allows the assistant to present a resulting information state where her physical environment aligns with her personal schedule, streamlining her routines.

Agentic IR represents a fundamental shift in how life assistants interact with users. By anticipating needs, understanding context, and performing tasks autonomously, agentic IR makes life assistants not just more useful, but indispensable. The proactive nature of these systems — along with their ability to integrate multiple sources of information, learn from interactions, and act independently — enables them to provide a uniquely tailored and efficient user experience.

3.2 Business Assistant

Business assistant is designed to support enterprise users by providing relevant business knowledge and insights based on various documents and data sources. With agentic IR capabilities, the business assistant goes beyond passive information retrieval to actively participate in intention recognition and response generation. Leveraging powerful information retrieval and generation capabilities, business assistants can address a wide range of business-related queries, from financial analysis to marketing strategies, helping users make better decisions.

Currently, there are already several business assistants powered by agentic IR in use, such as Microsoft 365 Copilot[4], Notion AI [5], and IBM watsonx[6]. Typically, the workflow of business assistant consists of four stages: query understanding, document retrieval, information integration, and response generation, as illustrated in Figure 3. Each stage is described in detail below.

Query understanding. Given a business-related query, the agent, core of the business assistant, first attempts to understand and analyze the user's intention. For complex queries, the agent can generate thoughts (THT) with CoT to break down the problem into smaller, manageable steps, allowing for multi-step reasoning. In business assistants, conversations are continuous, allowing the historical

---

[4]https://www.microsoft.com/en-us/microsoft-365/copilot/copilot-for-work

[5]https://www.notion.so/product/ai

[6]https://www.ibm.com/watsonx

dialogues to serve as memory (MEM), helping the agent better understand the context and user intention.

Document retrieval. Based on the query, the agent retrieves relevant information from external and internal documents to extract the most pertinent data. Given the diverse formats of documents (e.g., PDFs, figures, tables), the agent may utilize tools (TOOL) such as OCR for scanned text or SQL for structured data. In addition, the agent can leverage semantic search capabilities to go beyond simple keyword matching, ensuring that the retrieved information aligns more closely with the intent of the query.

Information integration. In many cases, the retrieved information is scattered across multiple sections or even different documents. To construct a comprehensive response, the agent must combine and distill the information, which often requires functions like THT and TOOL. By generating internal thoughts (THT), the agent can establish logical connections between disparate pieces of information and perform complex reasoning, gradually working toward the resulting information state. Additionally, specialized tools (TOOL) assist in this process by enabling capabilities such as executing mathematical calculations and filtering out redundant information.

Response generation. Finally, the agent generates a response, reaching the resulting information state. Depending on the query and the retrieved data, the resulting information state can take various forms. The response may be presented in multiple formats, including plain text, tables, visualized charts, etc. Additionally, the assistant can complete tasks and return an action state. For transparency, responses can also be linked back to their original source documents, allowing users to trace how the information was derived.

The application of business assistant is continuously evolving with advancements in agentic IR and growing market demand. Key trends include enhanced contextual understanding and multi-step reasoning, enabling business assistants to comprehend and execute more complex instructions. Additionally, in business scenarios where data is generated continuously, business assistants will need to retrieve and integrate information from ever-updating sources. Security is also a critical concern, including the protection of internal enterprise data and ensuring the safety of the responses generated by the agent.

3.3 Coding Assistant

Interactive programming assistance and automatic program synthesis play important roles in liberating productivity and improving development efficiency. Industrial programming assistant products like Copilot [7] emerge, offering an interactive environment for developers to gather information from open world and reach their programming needs. Agentic information retrieval in the context of coding assistants refers to systems designed to autonomously retrieve and provide relevant information based on developer queries and contextual needs. This approach emphasizes the ability of the assistant not just to respond to requests but to proactively understand developer intent, code context, and potential challenges the user may face. The main stages of Developer - Coding Assistant interaction process can be summarized into information need diagnosis, knowledge content generation, and information state update. An illustration is provided in Figure 4.

Information need diagnosis. At a certain state s_t, the developer's information need can be conscious and unconscious. On the one hand, developers can consciously input their requirements like "Generate docs" or call '*/doc*' to retrieve documentary knowledge from the coding assistant to suit the need. On the other hand, the information need can be unconscious. For example, during the programming procedure, a developer may write a function declaration, while the functionality to be achieved can be automatically identified by the coding assistant and then used to query itself for corresponding assistance. This unconscious information need diagnosis makes agentic IR distinct and advanced compared to the traditional IR, offering timely and tailored knowledge assistance. Another characteristic of agentic IR is the memory module (MEM), which allows the coding assistant to remember previous interactions, including developer preferences, past queries, debugging histories, and specific coding projects. This enables the assistant to maintain context over time, providing more tailored information need diagnosis.

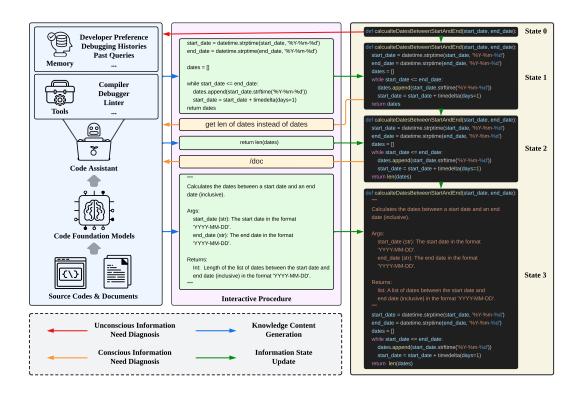[7]https://github.com/features/copilot

Figure 4: Illustration of agentic IR in coding assistant scenarios.

Knowledge content generation. After the information need is identified, it is then used to query the code assistant for corresponding knowledge content. Powered by the intelligent large language model like OpenAI CodeX [8], the coding assistant enables the developer' access to public documents and a professional programming assistant. As a high reasoning-demand task, code generation and debugging often requires an intermediate thinking process (THT) [Li et al., 2024c], seeking an optimal knowledge content. In addition, integrated with various coding tools (TOOL), like debuggers, compilers, and linters, the coding assistant can provide reliable and non-parametric knowledge to suit the developer's information need. For example, the coding assistant can synthesize the codes to be completed enhanced by THT, call test generation to generate tests and provide compiler feedback enhanced by TOOL for debugging purpose, retrieve and complement documents for codes refinement, etc. The generated knowledge content is then presented to the developer.

Information state update. After the knowledge content is generated, the developer can then perceive the knowledge and proceed to refine its work. This refinement updates the information state of the developer to reach s_{t+1}, where a new round of interaction is then activated.

During the interaction process, the developer gathers timely, tailored, and evolving information from the coding assistant, proceeding to the resulting information state s_T where a qualified code or project is accomplished.

4 Challenges

As a branding-new paradigm of IR, most of the techniques and engineering modules of agentic IR are still in their infancy and facing challenges in different aspects.

- **Data acquisition.** As a decision-making task, the logged data of agentic IR largely comes from the agent's interaction with the environment, which is determined by the users' instructions, the agent policy, and the environment dynamics. The exploration-exploitation tradeoff will be crucial to

---

[8]https://openai.com/index/openai-codex

collecting high-quality and wide coverage data. Directly labeling the correct trajectories to achieve the target state is still possible, but highly expensive.

- **Model training.** As the agent policy would consist of a DAG of functions, i.e., memory update, thought manipulation, and tool use, it is highly challenging to effectively update the parameters of these functions and the total composite policy function. Some recent attempts to tackle this challenge via RFT [Christianos et al., 2023] and action decomposition [Wen et al., 2024] have been performed.

- **Inference cost.** Due to the large parameter size and the autoregressive nature, the inference of LLMs is both GPU-heavy and time-consuming. Thus, the system optimization of agentic IR is crucial for practical service deployment.

- **Safety.** As the agent directly interacts with the real environment, its decision of action will change the environment and carry the user to different resulting information states. Thus, it is even more important than the chat applications to guarantee safety across the user journey. Alignment techniques [Ji et al., 2023] can be helpful but the safety is not guaranteed. A recent proposal of "world model + verifier" framework [Dalrymple et al., 2024] can be a way to explore safety for agentic IR.

- **Interacting with users.** Finally, given the differences from traditional IR in almost all aspects, including inference latency, data manipulation, information state representation, etc., the product form of agentic IR is still under-explored. There is still a long way to go to find the product-market fit for agentic IR.

5 Conclusions

In this position paper, we conceptualize a new technical paradigm of information retrieval in the era of LLMs, named *Agentic IR*. Unlike the traditional IR that filters the item corpus and returns the relevant items to the user, in agentic IR, the agent automatically interacts with the environment to reach the user's target information state. As such, the agentic IR serves a broad task scope, employs a unified agent architecture, and involves different key methods compared to traditional IR. Although facing challenges in multiple aspects, it can be expected that agentic IR will be highly developed and promoted in the coming couple of years.

Acknowledgments

References

Amit Singhal et al. Modern information retrieval: A brief overview. IEEE Data Eng. Bull., 24(4): 35–43, 2001.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 515–524, 2017.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. Modern information retrieval, volume 463. ACM press New York, 1999.

Jiarui Qin, Jiachen Zhu, Bo Chen, Zhirong Liu, Weiwen Liu, Ruiming Tang, Rui Zhang, Yong Yu, and Weinan Zhang. Rankflow: Joint optimization of multi-stage cascade ranking systems as flows. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 814–824, 2022.

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.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. Hammer: Robust function-calling for on-device language models via function masking. arXiv preprint arXiv:2410.04587, 2024.

Filippos Christianos, Georgios Papoudakis, Matthieu Zimmer, Thomas Coste, Zhihao Wu, Jingxuan Chen, Khyati Khandelwal, James Doran, Xidong Feng, Jiacheng Liu, et al. Pangu-agent: A fine-tunable generalist agent with structured reasoning. arXiv preprint arXiv:2312.14878, 2023.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 55(9):1–35, 2023.

Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. Trad: Enhancing llm agents with step-wise thought retrieval and aligned decision. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 3–13, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems, 36, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems, 36, 2024.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In Proceedings of the 22nd international conference on Machine learning, pages 89–96, 2005.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419): 1140–1144, 2018.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35:27730–27744, 2022.

OpenAI. Learning to reason with llms. `https://openai.com/index/learning-to-reason-with-llms/`, 2014.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In Forty-first International Conference on Machine Learning, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. arXiv preprint arXiv:2211.14275, 2022.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated process supervision. arXiv preprint arXiv:2406.06592, 2024.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In The Twelfth International Conference on Learning Representations, 2023.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. arXiv preprint arXiv:2402.05120, 2024a.

Apple. Apple intelligence foundation language models, 2024. URL `https://arxiv.org/abs/2407.21075`.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal llm agents: Insights and survey about the capability, efficiency and security, 2024b. URL `https://arxiv.org/abs/2401.05459`.

Qingyao Li, Wei Xia, Kounianhua Du, Xinyi Dai, Ruiming Tang, Yasheng Wang, Yong Yu, and Weinan Zhang. Rethinkmcts: Refining erroneous thoughts in monte carlo tree search for code generation, 2024c. URL `https://arxiv.org/abs/2409.09584`.

Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition. arXiv preprint arXiv:2405.15821, 2024.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. Ai alignment: A comprehensive survey. arXiv preprint arXiv:2310.19852, 2023.

David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, et al. Towards guaranteed safe ai: A framework for ensuring robust and reliable ai systems. arXiv preprint arXiv:2405.06624, 2024.