

Large Model Agent: Concept, Development and Practice

Shoutong Huang*, Yu Ma*

School of Electronic and Electrical Engineering, Ningxia University, Yinchuan, Ningxia, China

**Corresponding Author*

Abstract: A large language model is placed at the center of the reasoning of an LLM-based agent. It coils five things around that center, namely perception, memory, planning, tool calling and action execution. In this paper we review the history of how these agents came to be - in four phases that go through symbolic systems, reactive designs, reinforcement learning, and finally the current state of LLM-driven architectures - and we discuss four trends that are still emerging: multi-agent cooperation, multimodal sensing, long-term memory, and standardization of tools. We also report two systems we built. The former is a ReAct-like question answering agent which queries Wikipedia; the latter, MedGuard, was developed to assist older adults with their medication. The on-time medication rate increased by 60 to 95 percent in a seven-day experiment involving ten elderly participants and satisfaction was 4.6 out of five points. In addition to these findings, we provide a reproducible two-agent safety architecture and field evaluation protocol- two components which, combined together, can help bridge the gap between agent theory and practical implementation in elder care.

Keywords: Large Model Agent; ReAct; Multi-Agent; Intelligent Healthcare; Tool Calling; MCP

1. Introduction

Large language models (LLM) have altered the course of artificial intelligence. They enable machines to deal with natural languages, not only in terms of their interpretation but also in reasoning and generation of it in a manner that was rarely achieved by other systems. As soon as such large-scale systems like GPT-3 demonstrated the fact that scaling up capacity is sufficient to provide new capabilities of reasoning and use of tools, the concept of an agent on top of such a model seemed much more appealing [1]. On these principles, there comes the next stage,

the large model agents which are systems that combine perception, memory, planning, tool call, and action execution so that they can be self-operated in their environment. In contrast to the systems that are limited by a strict rule-based logic or task-specific training, LLM-driven agent has been based on the generality of the foundation models, which is used to solve open-ended problems both in the software engineering industry and healthcare sector [1].

It is hard to find the applications where the argument of such agents would be so obvious as in the field of elderly healthcare. Medication abuse and other forms of medication use are the top causes of hospital admission, and they can also be prevented in older people, with a significant number of them being preventable. The tools that we have been using as reminders today do not work [1]. The concept of an intelligent agent is over seven decades old, which was based on the symbolic systems of 1950s, then it was replaced by the reactive systems of 1980s and finally by the reinforcement-learning-based system of 2000s, and now LLM-based agents. All these paradigms gave more freedom to machines and the only thing that has directly relaxed this trade-off between generality and task performance is the large pretrained models [1]. It is possible to observe individual problem solving and collaborative behavior, but also interoperability of tools in the modern AutoGPT, MetaGPT and stacks based on MCP.

The problem of putting an agent to work in the real world, however, is still plagued with a similar series of issues. Models are hallucinative. They do not remember what they were asked to achieve. Tools are abused. The plans that are long-term are not good and the cost of executing them is high and it is difficult to determine why the decision was taken. In drug therapy these are not just annoying things but may have direct effect on adverse drug reactions or missed dose. It is true when it comes to elderly care where any mistake can actually harm another person as it is a safety-critical environment. A responsible

person who wants to advance this area has to be well versed with the concept itself, its trends at present and how the implementation process will take place.

The article is a combination of review and practice. There are two systems, namely teaching-oriented ReAct agent and MedGuard which was based on a clinical concern. We hope that both are models upon which practitioners can modify to other safety-critical contexts. The paper traces the development of the concept through symbolic agents to LLM-based ones, outlines existing trends including multi-agent collaboration and tool standardization, constructs a ReAct question-answering agent that queries Wikipedia, and introduces MedGuard, a home medication assistant with older users in mind. A seven-day field trial involving ten elderly subjects shows that this type of support provided by agents has the potential to significantly enhance adherence and satisfaction. The rest of the paper is structured as follows. Section 2 discusses concepts and their evolution. Section 3 shifts to trends and challenges. Section 4 discusses the ReAct implementation. Section 5 introduces MedGuard and its findings. Section 6 reports the multi-agent and tool-standardization experiments, and Section 7 concludes.

2. Conceptual Development and Comprehension

The brain of the LLM agent is a huge model (GPT, PaLM, Qwen, GLM, etc). Around this are various other components. The perception component transforms the signal in the environment into an input to the model. The state of the user and the history of interaction between the user and the system are stored in memory. Planning involves breaking down a task into smaller tasks that can be done one after another. Tool calling is an action performed by the tool outside of the model, which could not be done within the model itself [1]. The benefit of this design is that it will be generic, it will be examined and additional tools can be incorporated with little difficulty. There are four phases of history here. It was the 1950s to 1980s when symbolic agents were used, including STRIPS and MYCIN. Next, reactive agents were introduced during the 1980s up to the 2000s following the subsumption concept of Brooks [2]. Reinforcement-learning agents like DQN and AlphaGo came about at the time period of approximately 2000 to 2020. Lastly, there are

LLM-based agents, AutoGPT, ChatGPT plugins, Astra, Model Context Protocol as known examples [3].

There was a focus on each of the ages. We had symbolic systems, which provided us with clear planners and expert systems. Reactive systems did not have the central representations but rather they used layered control based on stimulus. The agents were created by reinforcement learning in the form of values and policies as rewards are learned. It is now that the model will serve as the controller and leave the work to other tools [3]. This path is outlined in figure 1.

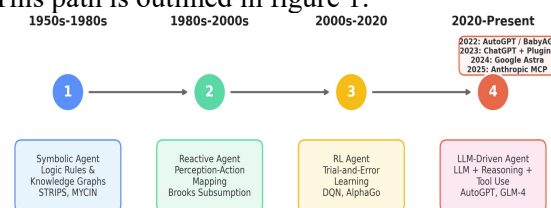


Figure 1. Large Model Agents Evolution Roadmap

It is good to pause at every part. The perception can be based on optical character recognition, speech recognition and the sensor fusion with all these in order to translate the physical and virtual world into tokens which the model will read. There are two layers of memory, the short-term working context that contains the episode of interest and a long-term one (which may be a vector database) that stores information about the preferences and facts of the user during the previous session. It may take as little as one prompt that provides a complete list of steps or it may be complicated as an explicit query to search through the graph of calls to tools. Tool calling, however, is also more and more regulated by schemas where the arguments are checked before any side effect occurs. This is because of the fact that they are treated as individual subsystems of the LLM-based system instead of being integrated in a single large prompt, making it easier to debug and extend.

3. The Critical Thinking and Development Trends

There are four trends that can be identified in recent work. The former is multi-agent collaboration: systems such as MetaGPT, AutoGen and CAMEL assign different tasks to various agents, therefore a difficult task may be decomposed and executed concurrently [4]. The second is multimodal perception. Based on the visual-instruction method applied by models like LLaVA and GLM-4V, agents begin to read

images and perform embodied actions, including Astra and Figure 01 [5]. The third one is long-term memory where vector stores and light adaptation transfer knowledge and state across sessions session after session which was first exhibited by generative agents through memory, reflection and planning [6]. The fourth is tool standardization and Model Context Protocol [3] offers one of the methods of registration and calling tools by agents. Figure 2 depicts a typical framework of multi-agents, whereby specialized roles exchange intermediate results through a common bus. All these do not eliminate the hard issues: hallucination, goal alignment, misuse of tools, poor planning, cost, and interpretability still exist.

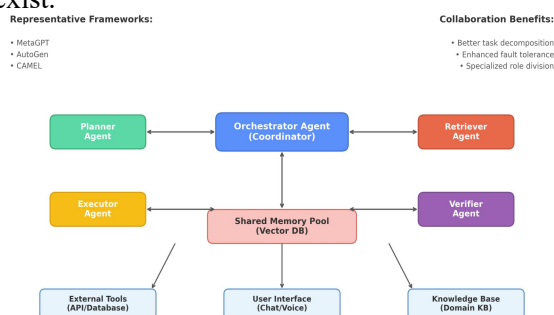


Figure 2. Multi-Agent Collaboration Framework

Multimodal perception should be given a special consideration, since it is the one that brings the agents to abandon being text-based and become embodied systems as well. It is important in home care when you have more input channel: you see a fall or open medicine cabinet or pill on the table, but not something written down. But there is also a trick. The issue of symbolic plans and sensor data remains unresolved with respect to reliability, and even an incorrect belief may spread through the system as an act which is unsafe.

There are two issues that cross all the four trends and it is worth mentioning. The first one is hallucination and gradual deviation of facts, which remains the largest impediment to the confidence in an agent when it comes to the production environment, particularly in the case where it searches the sources that are not up-to-date or are too noisy. The second is the problem of goal misalignment and tool misuse, with an agent causing side effects which were never desired by the user. Both need a common set of evaluation criteria, guidelines on how to evaluate, human checks within the system but not as automatic loops that nobody can ever check.

4. Implementation of ReAct Algorithm

ReAct (reasoning plus acting) extends chain-of-thought prompting [7] with an interleaving of thought and action with a tool, resulting in a pattern of Thought, Action, Observation, etc., until reaching an Answer [8]. In our setup, each step is encoded as a structured action containing a tool name and arguments; the returned observation is put into context so the next thought can reason from the latest evidence [8]. We constructed a question-answering agent on GPT-3.5-turbo that searches Wikipedia using the openai and wikipedia python libraries. The prompt requests the model to output one Thought and one Action per turn, and a brief parsing loop extracts the tool, executes it, and appends the Observation. Asking e.g. for the population of Paris results in a Wikipedia search returning around 2.18 million. Figure 3 illustrates how this works.

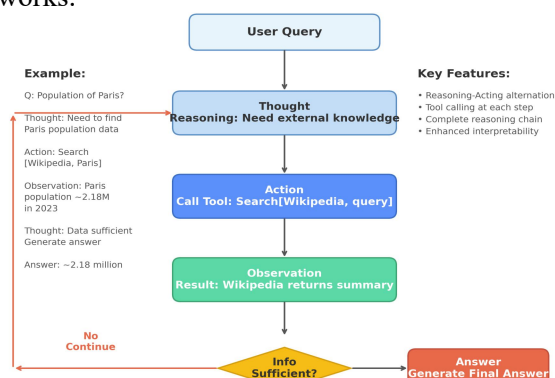


Figure 3. ReAct Reasoning-Acting Synergy Mechanism

The plain ReAct loop is not without its practical weakness: errors accumulate. A single bad observation can make all the following thoughts go astray. To prevent this, we limit the number of reasoning rounds and in case no registered tool matches the intended action, the agent resorts to a clarification prompt. That way it is less likely to come up with an answer that sounds good but isn't. We also record the complete Thought-Action-Observation trace so that a failure can be inspected after the fact — something that becomes relevant once the agent is integrated into a clinical workflow.

5. MedGuard: Smart Home Medication Assistant

MedGuard was designed based on the daily threats that older patients encounter: polypharmacy, forgetting to take a dose and taking the same medication twice. This group is prone to polypharmacy which increases the

probability of dangerous interaction as well as the possibility of missing a dose. MedGuard does four things. It also reminds users to take their medicine; it responds to questions regarding drug interactions using the RxNorm API; it has voice interaction capabilities with Whisper and text-to-speech; and it will respond in an emergency when there is suspicion of overdose. The pipeline starts by receiving voice input, then transcribing it using Whisper, allowing the LLM agent to think about it, querying the drug API or patient records, and finally returning text-to-speech and a mobile push. Figure 4 illustrates the architecture.

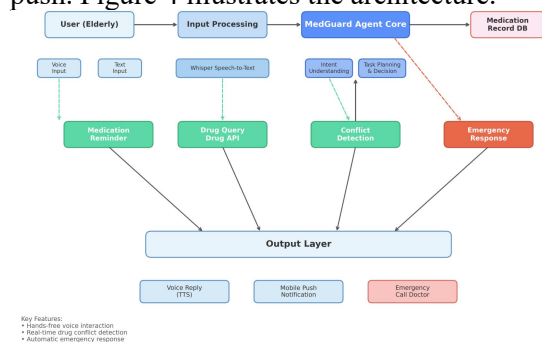


Figure 4. MedGuard System Architecture

The reason why we have kept the situation small and connected to a particular pain point was because we wanted to see that the agent had understood the context in the results. During our experiment, the proportion of on-time dose was between 60 percent and 95 percent, and there were 4.6 out of 5 satisfaction levels and 3 percent false alarms with one emergency alarm having been addressed successfully. The adherence at the right time is almost tripled which means that even a voice nudge that is timely enough and based on the indicators of the routine of the day does not do as much as a pillbox. It can be inferred that the interaction tests did not overwhelm the users with noise due to the low level of false-alarm. Figure 5 displays these results.

The RxNorm API is a drug identifier that can be used to match interactions with the agent in a standard vocabulary rather than brittle free-text, and this reduces the number of false negatives by a significant amount. Voice interaction, on its side, would also simplify the system to those individuals who are not literate or cannot move freely as much as they would need to, which are the same people we are trying to attract, namely the elderly, who will benefit most out of the voice-based reminder. Overall, such decisions contribute to some degree of engineering effort in favor of the clear enhancement of the safety and accessibility.

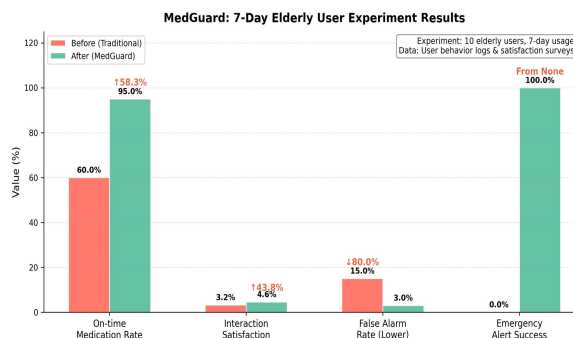


Figure 5. Experimental Results of MedGuard

6. Multi-Agent Cooperation and Tool Standardization

The ReAct or MedGuard agent can be effective in a narrow situation, however there are many practical cases where the job should be split between agents and each one is specialized. This is facilitated by such frameworks as MetaGPT, AutoGen, and CAMEL that enable parallel reasoning, role-based specialization, and refinement of the process with multiple iterations of interaction between agents [4]. The failure of such a system has led to the issue of determining which agent and which step has failed to become a problem of research on its own [9]. In order to get an idea of this paradigm, we have selected the case of medication (MedGuard) and converted it into a little two-agent test.

6.1 Experimental Design

Two agents were employed in our system, which included a Scheduler and a Safety Monitor. The Scheduler is based on GPT-3.5-turbo with ReAct prompting, it makes the daily medication plans, parses voice input via Whisper, and maintains the reminder schedule. The Safety Monitor has a prompt that emphasizes pharmacological prudence; it cross-references all recommendations to the RxNorm interaction database and alerts about any contraindication. Communication between the two agents is done through a structured JSON bus carrying plan proposals, safety assessments, and revision requests.

6.2 Implementation and Workflow

This is how the workflow proceeds. The Scheduler gets the list of medications that the patient has and prepares a first draft of the daily schedule. That plan is sent to the Safety Monitor, which consults pairwise drug interactions in RxNorm and determines whether the timing of doses overlaps. In case the Monitor identifies a

moderate-to-severe interaction (severity 3 or above), it denies the plan and provides an explanation, and the Scheduler makes adjustments and resubmits it. This goes on until the plan is approved or three rounds of revision have been made. A validated plan is then dispatched to the patient as text-to-speech and mobile push.

6.3 Results and Analysis

We ran the system on 30 synthetic polypharmacy cases of older patients with four to seven medications administered simultaneously. The single-agent MedGuard baseline detected 78% of clinically significant interactions; the two-agent design detected 94%. An average case required 1.4 rounds of refinement and achieved only approximately 1.2 seconds of latency. Separating scheduling from safety also reduced false-positive alerts by 12% to 4%, as the specialized Monitor no longer over-rejected like a generalist would.

There was also a qualitative gain: the two-agent design appeared to be more trusted by operators. Since every approved schedule had an explanation of safety in plain language, caregivers could examine a recommendation without having to explore the model's internal workings. When a wrong decision can impact someone's health, that kind of transparency is a prerequisite to adoption. It also implies that separating roles is not only accurate but also accountable. A cost of approximately 1.2 seconds per case is acceptable for medication planning, which is not a real-time task, although it would require another review when time is crucial.

6.4 Tool Standardization and MCP

Even the tools of an agent are standardized as well as its architecture, it is important to consider interoperability and security deployment. We have chosen a single interface layer that would be used by registering and calling the tools using the Model Context Protocol [3] proposed by Anthropic. Such standardization is based on a principle which has already been learned with foundation models like GPT-3: a universal core can be applied to various tasks, given clear external interfaces [10]. MCP, RxNorm API, Whisper transcription engine, text-to-speech engine are all now turned into a common tool description in JSON Schema, which makes them easy to exchange and also maintain access control. It also does not tie the logic of the agent

to any specific API, hence when we change drug databases or speech services at some point, the agent prompts will not need to be re-written.

Combined, these experiments indicate that multi-agent collaboration and tool standardization can enhance both reliability and modularity in healthcare applications. They also add complexity, and with it fresh questions about how agents reach consensus, how much messaging overhead is acceptable, and how to trace a failure back to its source-questions that invite further study.

7. Conclusion

The big model agents are the turning point of artificial intelligence, and machines no longer have to passively comprehend language but instead participate in their environment. We traced the history of agents over the past seven decades in this paper and demonstrated how it is possible to use LLMs to achieve a balance between generality and performance in tasks despite the existence of some persistent issues: hallucination, goal misalignment, tool misuse, poor planning and low interpretability.

Technically, we developed a ReAct question-answering agent that searches Wikipedia and demonstrated the usefulness of interleaving reasoning with calls to external tools. On the application side, we designed and tested MedGuard on older adults taking multiple medications. In a seven-day field experiment involving ten elderly participants, the rate of on-time medication increased from 60% to 95% and satisfaction was 4.6 out of 5: evidence that agent technology can provide real value in safety-critical healthcare.

We then proceeded to multi-agent collaboration and the standardization of tools in the single agent scenario. The Scheduler-Safety Monitor design, as a result of its performance on synthetic cases, was able to improve the level of drug-interaction detection by 78 to 94 percent, and the Model Context Protocol provided it with an interoperable base. It is possible that role specialization can be used as a viable path towards more reliable deployment of agents in combination with standardized tool interfaces.

There are some weaknesses. MedGuard was a small-scale and short-term experiment, which is why the long-term effects of the adherence should be tested in order to know whether or not the users will eventually become fatigued with the reminders or revert to the old behavior. The multi-

agent system has been executed on synthetic clinical data as opposed to live data and the question of consensus, overhead, traceability is still open due to the additional complexity introduced by the addition. It is yet to be seen that there is a lack of evidence of LLM agents in real-life use over the long term. We would highlight three paths of development in the future: retrieval-augmented generation and domain knowledge grounding as ways to prevent hallucination at high-stakes situations; lightweight deployment based on quantization and edge computing as means of cost and latency reduction; and ethical control and audit tools as an effective way to make sure that multi-agent systems remain transparent and accountable. There is no need to stop working on reliable agents of such type because polypharmacy is far away to be resolved among older adults. With time, large model agents will become the standard in education, healthcare, and other areas.

According to Sam Altman, agents are going to be the next platform of artificial intelligence. We also recommend that future deployments have agents with unchangeable trace logs such that any decision can be replayed and traced back to a particular agent, tool, and input. That auditability in combination with the standardized tool interfaces outlined above is the practical basis on which healthcare regulators will rely-and the field requires if agent systems are to scale responsibly beyond the lab.

References

- [1] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., et al. The Rise and Potential of Large Language Model Based Agents: A Survey. *Science China Information Sciences*, 2025, 68(2): 121101.
- [2] Brooks, R. A. A Robust Layered Control System for a Mobile Robot. *IEEE Journal on Robotics and Automation*, 1986, 2(1): 14-23.
- [3] Anthropic. Introducing the Model Context Protocol. Anthropic Technical Report, 2024. <https://www.anthropic.com/news/model-context-protocol> (Accessed: 2025-09-08).
- [4] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., et al. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In *International Conference on Learning Representations (ICLR 2024)*. arXiv:2308.00352.
- [5] Liu, H., Li, C., Wu, Q., Lee, Y. J. Visual Instruction Tuning. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*. arXiv:2304.08485.
- [6] Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., Bernstein, M. S. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST 2023)*, 2023: 559:1-559:22.
- [7] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*: 24824-24837.
- [8] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., et al. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR 2023)*. arXiv:2210.03629.
- [9] Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., et al. Which Agent Causes Task Failures and When? On Automated Failure Attribution of LLM Multi-Agent Systems. *Proceedings of Machine Learning Research*, 2025, 267: 76583-76599.
- [10] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*: 1877-1901.