ISSN NO: 9726-001X
Volume 13 Issue 02 2025

JOURNAL OF CURRENT SCIENCE

REACT-POWERED AI AGENT FOR INTELLIGENT WEB SEARCHING AND CONTEXTUAL DECISION-MAKING

Nadia Shaikh,
UG Student,
Department of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India
nadiashaikh2024@gmail.com

Ms. P. Swetha,
Assistant Professor,
Department of CSE,
St. Martin's Engineering College,
Secunderabad, Telangana, India
gshwethacse@smec.ac.in

Abstract - In recent years, humankind has witnessed a surge in revolutionary technologies, with Artificial Intelligence (AI) emerging as a prominent force shaping industries and daily life. Organizations are increasingly seeking scalable solutions that optimize efficiency, enhance accuracy, and minimize resource consumption. This rising demand for automation has led to the development of AI agents. AI agents are autonomous, perceptive software systems capable of performing tasks on behalf of users. These intelligent agents possess key functionalities such as reasoning, planning, decision-making, learning, adaptation, and action execution. They find applications across various domains, such as research, automation, sales, marketing, search engine optimization (SEO), coding, customer support, and beyond. The AI agent industry is projected to become a multi-billion-dollar market by 2030, revolutionizing the way businesses and individuals interact with technology. A crucial aspect of AI agent design is the ReAct (Reasoning + Action) framework, which enables agents to dynamically think, act, observe, and refine their responses in an iterative manner. The ReAct framework operates through a structured cycle comprising three key components: Thought, Action, and Observation. This loop mimics human cognitive processes by allowing the agent to generate thoughts, determine the best course of action based on available resources, execute actions, and assess observations to decide the next steps. By continuously learning from its observations, the agent refines its decision-making process, making it more adaptive and intelligent over time. As the field of AI agents gains prominence, numerous no-code and low-code platforms have emerged, allowing businesses and developers to build AI-driven assistants that perform specialized tasks independently or collaborate with other agents. We explore the architectural foundations of ReAct agents by constructing one from the ground up and evaluating its efficiency in retrieving and processing information. To accomplish this, we integrate the Groq API for leveraging a Large Language Model (LLM) and utilize SERPApi to facilitate real-time web searches via Google. The Groq API enables us to integrate a large language model, serving as the core of the ReAct implementation. The SERPApi gives Google Search Results. This project illustrates the practical implementation of the thought-action-observation cycle, showcasing how an AI agent autonomously navigates search queries, processes retrieved data, and returns relevant, contextual, and optimized results. By analyzing the effectiveness of the ReAct approach in structured decision-making and information retrieval, this research highlights the potential of AI agents in enhancing knowledge discovery, automating workflows, and optimizing human-computer interactions. Through this work, we aim to contribute to the growing field of AI-driven automation by demonstrating the real-world applicability, efficiency, and scalability of ReAct-based intelligent agents.

Keywords: AI Agents, ReAct Framework, Large Language Models, APIs, Web Search, Thought-Action-Observation loop.

I. INTRODUCTION

An AI Agent is a software system capable of autonomously performing tasks without human intervention. These systems collect data from their environment and leverage it to achieve specific objectives. Their core functionalities include reasoning, acting, decision-making, learning, and task execution. Unlike traditional chatbots, such as ChatGPT, which engage in conversational interactions, AI agents operate based on their perception of an organization's needs and execute tasks accordingly. AI agents can be classified in multiple ways, depending on their application. Some define them as systems capable of independently performing tasks for extended periods while utilizing different tools for handling complex workflows. Others describe AI agents as implementations that follow predefined workflows [1]. Many companies have already integrated AI agents into their teams, with approximately 51% of organizations adopting them in development environments. Mid-sized companies (ranging from 100 to 2,000 employees) have shown a particularly high adoption rate, with 63% deploying AI agents in production [2, 3].

To better understand the ReAct framework, let's consider a simple analogy: making tea. The process begins with a thought, the person decides to boil water. To achieve this, they perform an action that is lighting the stove using a tool, such as a lighter. After observing that the water has started boiling, they articulate the next thought which is to add tea leaves and sugar. Once they observe the tea brewing, they proceed to add the required amount of milk. Each thought leads to an action, and based on the resulting observation, the next thought is triggered. This cyclical

process of thought, action, and observation is central to the ReAct framework. Similarly, an AI agent implementing ReAct operates by reasoning through a task, executing an action, observing the outcome, and iterating based on new information [4, 5]. To execute specific actions, the AI agent utilizes tools from a predefined set. These tools enable the agent to interact with external systems, fetch data, or perform computations, just as a human would use utensils and ingredients while making tea.

The system prompt plays a crucial role in guiding the AI agent, instructing it on how to think, what tools to use, and how to articulate its reasoning to invoke subsequent thoughts. For this implementation, we use the Groq API, which allows us to integrate a Large Language Model (LLM) as the agent's core reasoning engine. Using an API key, the client sends requests to the server, processes responses, and returns meaningful results. Users can specify their preferred LLM model, in this case the Llama-3b-70-8192 model has been used as it is better suited for the implementation of AI agents. To incorporate web searching capabilities, we utilize SERPApi, an efficient API that fetches real-time web search results from Google. The retrieved results are then processed by the LLM, which analyzes and synthesizes the most relevant answer. This enables the AI agent to perform live web searches and return accurate information dynamically.

II. RELATED WORK

In [6] Yan et al, proposed a general AI agent framework for smart buildings based on LLMs and ReAct Strategy. In the construction industry, the concept of smart buildings is gaining traction. It can be achieved with effective interaction algorithms which current human-computer interaction algorithms lack. So, the paper introduces the concept of large language models and Ai agents into smart building, using ReAct strategy. The LLM acts as the core of the agent and has reasoning abilities to make the required decisions. LLM will interpret user intent and accordingly perform the necessary actions. The experiment was conducted on a virtual building and demonstrated completion of 91% of simulated tasks. The agent was deployed on a single-board computer.

In [7] T. Masterman et al, examined the advancements in AI agent implementations, and how they can achieve complex goals using reasoning, tool execution and planning. In this paper, the authors have mentioned the pros and cons of existing implementations of AI Agents. They have also provided insights gained from their observations by providing an explanation on single-agent and multi-agent architectures and assessing their impact on accomplishing goals. In this paper, the authors have outlined the key themes when selecting an agentic architecture, the impact of leadership on agent systems, agent communication styles, and key phases for planning, execution, and reflection that enable robust AI agent systems.

J. Blümel and G Jha [8] designed a conservational AI Agent. They have explained how the existing methods of conversational AI agents do not ensure customer satisfaction. So, in this paper the authors have developed a framework to design conversational AI agents. They have proposed a six-stage iterative design that can sense the customer intent, adapt to the context, assign tone to the conversation, training of agents and adaptively improve. The aim of the project is to enhance CX (Customer Experience).

Dennis, Alan R, et al [9], performed a research which compared the performance of a virtual team of AI agents with a regular team of humans. They evaluated the coordination between the virtual and real time. Upon conducting an experiment, the researchers found out that the conflict between the agent and person was less when performance was good but the perceived conflict was more when there was lower performance. It was also observed that the process satisfaction was lower. Their research suggested that AI team members are likely to be accepted into teams, meaning that many old collaboration research questions may need to be reexamined to consider AI team members.

In [10] Hamada et al, have discussed about AI agents facilitating social interaction and wellbeing. It has become a trend in individual's mental wellbeing, organizational health, and has enhanced societies. There are diverse applications of wellbeing AI. In the paper, the authors have proposed a brief on the mediative role of AI-augmented agents for social interactions. A two-dimensional framework is developed which classifies wellbeing AI to individual or group and analysis or intervention. This agent also assists in human-human interaction and improving social behaviours. But this approach is unethical.

III. PROPOSED WORK

This research aims to build a ReAct Pattern from the ground up, to gain a understand of the foundations of this framework. Utilization of the framework in an AI Agent has many applications ranging from automation to research. AI Agents are efficient and intelligent software systems that can perform tasks autonomously on behalf of the user. The ReAct pattern, which stands for Reasoning and Action, is a structured methodology that defines how the AI agent processes information, makes decisions, responds and observes in an iterative manner much like the human mind thus making these systems more adaptable and intelligent.\

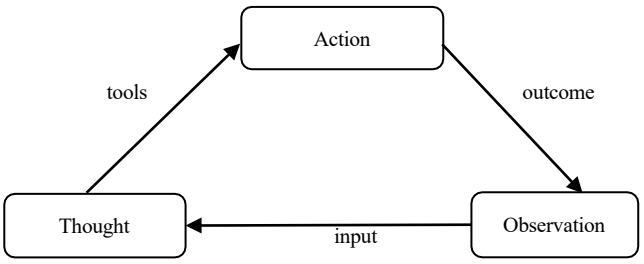

Fig 3: ReAct Framework Loop

The ReAct framework operates through a structured cycle consisting of Thought, Action and Observation. The first phase of the AI Agent is Thought in which it receives input and generates an internal thought process based on the context. This is where

reasoning takes place, allowing the agent to decide what action to take next. After reasoning, the next phase is Action. Here, the AI Agent makes an action by selecting a tool from the given list of available tools. The action's output is not the final output but rather it is a step that helps in refining the response. After performing an action, the AI agent observes the outcome and processes the result. If the retrieved information is insufficient, the agent might refine its approach and repeat the cycle. If the observation provides enough clarity, the agent finalizes its response. The below figure 3 illustrates the working of the AI agent built for web searching.

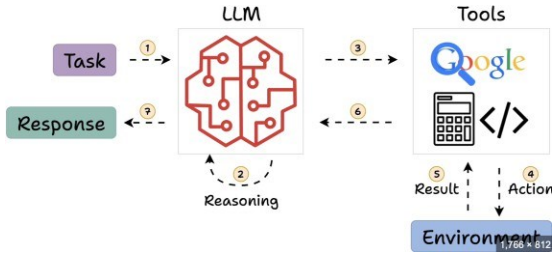

Fig 4: Design of the ReAct Framework for Enhanced Web Searching

3.1 Groq API

The API used in this research is provided by Groq, which allows the creation of a client that communicates via the API.
The Llama-3B-70-8192 LLM is being used, as research suggests it is effective in performing AI agent tasks. Groq's platform offers a suite of large language models (LLMs), allowing users to choose a model based on their specific use case and computational needs.

After obtaining the API key and configuring the environment, the user must perform a chat completion by specifying key parameters. These parameters define how the AI model processes and generates responses. Some essential parameters include:

- **Model**: Specifies the LLM to be used (e.g., "llama-3b-70-8192").
- **Messages**: A structured list containing user prompts and system responses. Typically, it includes roles such as "system", "user", and "assistant" to guide the conversation.

3.2 Agent Code

To enable AI-driven interactions using the ReAct Pattern, an Agent class is defined that encapsulates key functionalities required for reasoning and action execution. This class contains a parameterized constructor and three essential methods: __init__, __call__, and execute.

Initialization (__init__Method): The __init__method serves as the constructor for the Agent class. It initializes the following:

- **Client:** The API client used to communicate with the LLM.
- **System:** A predefined system prompt that establishes the agent's behavior and constraints.
- **Messages:** A list that stores the history of interactions, including user inputs, system messages, and AI-generated responses.

Handling User Input (__call__Method): The __call__method allows the agent instance to be invoked directly as a function. This method handles:

- Appending user messages to the interaction history.
- Calling the execute method to generate a response.
- Appending the AI-generated response to the message list.
- Returning the AI's response to the user.

Generating AI Responses (execute Method): The execute method is responsible for communicating with the **Groq API** and retrieving responses from the **Llama-3B-70-8192** model. It constructs a chat completion request using the message history and returns the AI-generated output.

3.3 System Prompt

The system prompt guides the AI Agent. It demonstrates the manner in which the AI Agent should work. This specifies how the agent will think, what tools to use, what should be done with the outcome of an action, and when to give the final response.
In this research, the system prompt is specified in a system prompt variable which contains a multi-line string. It mentions that the AI Agent works in a loop of Thought, Action, PAUSE, Observation. At the end of the loop the agent has to provide an answer. Thought is to be used to describe the agent's thoughts about the question asked. Action is used to run the available tools that the agent has, followed by this, the agent has to return PAUSE. Observation will be the result of the actions.

The next line describes the actions available to the agent. In this research the actions are:

- **calculate**: it runs a calculation and returns the number.
- **get_planet_mass:** returns weight of planet in kg.
- **search_web:** Searches the internet and returns relevant information.

Along with the name of the tools, examples are also provided to offer better guidance to the agent.

Two example sessions are specified which demonstrate how the agent, when given a question by the user, has to get a thought, which action it should perform, the outcome of the action leading

to an observation, which in turn acts as input for the thought in the next iteration of the loop.

For example, sample session 2: Question: What is the latest AI trend? Thought: I should search the web for AI trends. Action: search_web: Latest AI trends PAUSE You will be called again with this: Observation: AI is shifting towards multi-modal models in 2024. Answer: The latest AI trend is multi-modal models in 2024.

Finally we end with the line, "Now it's your turn."

3.4 Tools

To enhance the AI agent's functionality, several tools are integrated to perform external tasks such as web searching, mathematical calculations, and retrieving planetary data. Each tool provides specific capabilities that assist the agent in executing reasoning-based actions efficiently.

SerpAPI, Web Search Tool: SerpAPI is a Google Search API that allows the agent to retrieve search results programmatically. Before using this API, the google-search-results package must be installed. The API enables the agent to query Google, fetch relevant results, and extract information from organic search listings. This functionality is useful for retrieving up-to-date information that may not be present in the AI's pre-trained knowledge base.

Mathematical Computation Tool: The agent includes a simple evaluation tool to process mathematical expressions dynamically. This allows users to input arithmetic operations, which the agent computes and returns. By leveraging Python's built-in evaluation mechanisms, the tool supports basic operations like addition, subtraction, multiplication, and division.

Planet Mass Retrieval Tool: It provides the mass of various planets in the Solar System based on predefined values. When a user asks for a planet's mass, the tool matches the input with the corresponding celestial body and returns its standard mass in kilograms. This ensures accurate and quick retrieval of astronomical data.

3.5 Running the AI Agent

Once the AI agent and its tools are set up, it is executed using a loop-based mechanism that enables interaction, reasoning, and tool usage. The agent loop serves as the core logic of how the agent processes queries and responds iteratively.

Defining Tool Functions: A dictionary named tool_functions maps function names to their corresponding implementations. This ensures that the agent can dynamically call specific tools when required. The available tools include:

- calculate – Evaluates mathematical expressions.
- get_planet_mass – Retrieves the mass of a given planet.
- search_web – Searches the web using the SerpAPI.

Agent Loop Mechanism: The agent_loop function specifies how the agent will operate. The loop operates based on the following logic:

- **Initialization**:
  - An instance of the Agent class is created with a system prompt.
  - A list of available tools is defined.
  - The user query is set as the first input.
- **Iteration & Processing**:
  - The loop runs for a predefined maximum number of iterations.
  - The agent processes the query and generates a response.
  - The response is printed for monitoring.
- **Action Handling**:
  - If the agent detects an action request (e.g., calling a tool), it extracts the tool name and argument using regular expressions.
  - If the specified tool is available, the function is executed with the provided argument.
  - The observation from the tool is appended as feedback to the next cycle of reasoning.
- **Answer Generation**:
  - If the agent determines that an answer has been reached, it exits the loop.
  - Otherwise, the cycle continues with the updated prompt.

Execution Example: The agent is initiated with a sample query: agent_loop(max_iterations=10, system=system_prompt, query="What is the capital city of India?")

- The loop ensures that the agent can think, act, and observe repeatedly until a final response is produced.
- If external information is required, the agent utilizes one of the defined tools before generating a final answer.

3.6 Calculating Response Time and Accuracy

To evaluate the accuracy and time taken, five queries were given. User input was taken to determine the relevance of the given output. To calculate the response time, the time module in Python was used. A list 'response_time' stored the time taken for the query's answer to be returned. Followed by that, the average response time was calculated. Finally, the accuracy for the queries was determined by using the responses that the user gave 'yes' to by the total number of queries.

IV. RESULTS AND DISCUSSIONS

The below code snippet shows how the AI Agent using ReAct framework functions. The user queries, "What is the capital city of Delhi?" Then the Agent goes through the following:

```
Thought: I need to find the capital city of
India.

Action: search_web: Capital city of India
PAUSE
Observation: Delhi, officially the National
Capital Territory (NCT) of Delhi, is a city
and a union territory of India containing
New Delhi, the capital of India.
Thought: I have found the capital city of
India, which is New Delhi.

Answer: The capital city of India is New
Delhi.
```

Figure 4 shows how the ReAct framework used by the AI Agent for performing web searching using the given tools. The LLM acts as the core of the AI Agent and it utilizes the tools to perform the required tasks. The SERP API performs real-time searching which assists the model in providing the most relevant information related to the query. The AI-powered web search agent was tested using SERP API to determine the efficiency.

To get an estimate of the time taken, 5 queries were given to the agent. The relevance of the statements was also determined by calculating accuracy. The system demonstrated an average response time of 1.44 seconds, indicating efficient processing and retrieval speed. This performance is crucial for real-time web search applications where minimal latency is required. To assess accuracy, we manually reviewed the top search result returned by our AI for each query. Based on a binary user feedback system (relevant/non-relevant), the AI agent achieved an accuracy of 80%. This suggests that while the system is effective in retrieving relevant results, there is room for improvement in refining the query processing and filtering mechanisms.

V. CONCLUSION

In this paper, we developed and evaluated an AI-powered web search assistant capable of retrieving and filtering relevant information in real-time. By leveraging the SERP API, our system efficiently extracts query-based search results and assesses their relevance based on user feedback. The experimental results indicate that the AI agent achieves an accuracy of 80%, with an average response time of 1.44 seconds, demonstrating its effectiveness in providing quick and relevant search results.

While the model performs well in retrieving factual data, some limitations remain, particularly in handling ambiguous queries and ranking information based on context. Future work could enhance the system by integrating natural language understanding (NLU) techniques, improving contextual relevance, and incorporating machine learning-based ranking models. Overall, this research contributes to the development of AI-powered search assistants, bridging the gap between automated information retrieval and user intent.

REFERENCES

1. Anthropic, "Building Effective Agents," *Anthropic*, Feb. 2024. [Online]. Available: https://www.anthropic.com/engineering/building-effective-agents. [Accessed: 10-Mar-2025].
2. MarketsandMarkets, "AI Agents Market - Global Forecast to 2028," Sept. 2024. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/ai-agents- market-15761548.html
3. Roots Analysis, "AI Agents Market." Available: https://www.rootsanalysis.com/AI-Agents-Market. [Accessed: 9-Mar-2025].
4. S. Willison, "Python and the ReAct pattern for LLMs." Available: https://til.simonwillison.net/llms/python-react-pattern. [Accessed: 9-Mar-2025].
5. Yao, Shunyu, Zhao, Jeffrey, Yu, Dian, Du, Nan, Shafran, Izhak, Narasimhan, Karthik, and Cao, Yuan. *ReAct: Synergizing Reasoning and Acting in Language Models*. Retrieved from https://par.nsf.gov/biblio/10451467. *International Conference on Learning Representations (ICLR)*
6. X. Yan *et al.*, "A general AI agent framework for smart buildings based on large language models and ReAct strategy," *Smart Computing*, early online, 2025. [Online]. Available: https://pdf.elspublishing.com/paper/journal/open/SC/earlyOnline/2025/SC-20250004.pdf.
7. T. Masterman *et al.*, "The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey," *arXiv preprint arXiv:2404.11584*, 2024.
8. J. Blümel and G. Jha, "Designing a conversational AI agent: Framework combining customer experience management, personalization, and AI in service techniques," 2023.
9. Dennis, Alan R., Akshat Lakhiwal, and Agrim Sachdeva. "AI agents as team members: Effects on satisfaction, conflict, trustworthiness, and willingness to work with." *Journal of Management Information Systems* 40.2 (2023): 307-337.
10. Hamada, Hiro Taiyo, and Ryota Kanai. "AI agents for facilitating social interactions and wellbeing." *arXiv preprint arXiv:2203.06244* (2022).