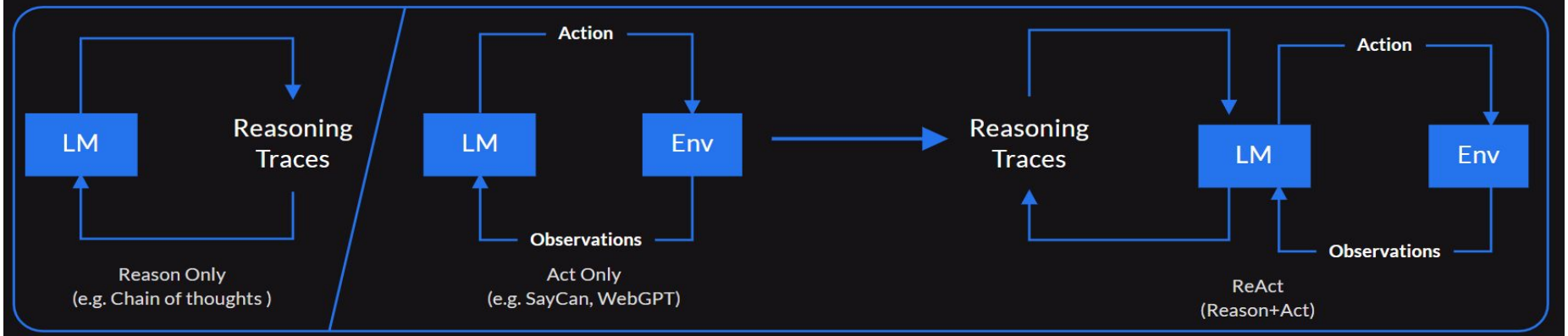
Comprehensive Guide to Build AI Agents from Scratch

Dipanjan (DJ) Sarkar

Analytics Vidhya

ReAct AI Agents

What is ReAct?

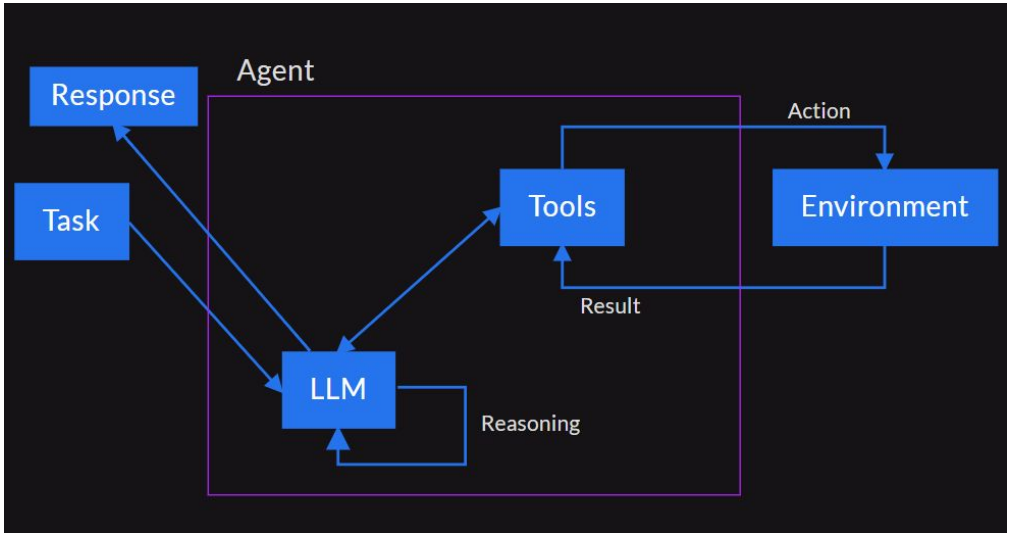


- Prompting patterns like Chain-of-thought enables LLMs to reason better
- Tools and techniques like SayCan, and WebGPT enable LLMs to take better actions
- ReAct aims to combine both of these elements to empower LLMs to reason and take action

First mentioned in the famous paper, 'ReAct: Synergizing Reasoning and Acting in Language Models' in March, 2023

Analytics Vidhya

ReAct AI Agents

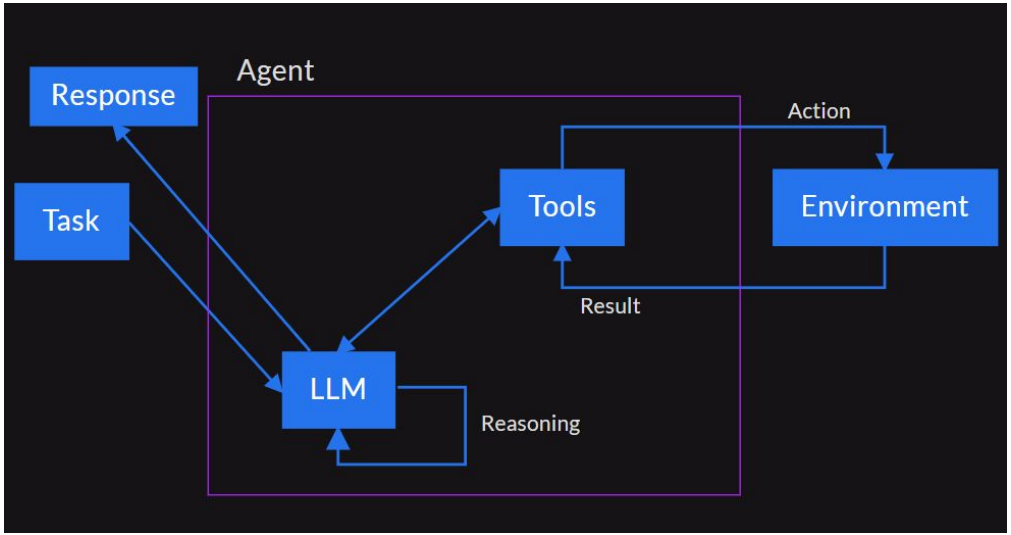


- The ReAct pattern operates in a cyclic loop consisting of the following steps:
- **Thought:** The AI agent processes the input and reasons about what needs to be done.
- **Action:** Based on the reasoning, the agent performs a predefined action.
- **Pause:** The agent waits for the action to be completed.
- **Observation:** The agent observes the results of the action. It analyzes the output received from the action.
- **Answer:** The agent uses the observed results to generate a response. This response is then provided to the user, completing the loop.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Tools and Libraries Needed



- For building AI agents, several Python libraries are essential:
- **OpenAI API:** This library allows you to interact with OpenAI's language models, such as GPT-3 and GPT-4. It provides the necessary functions to generate text, answer questions, and perform various language-related tasks.
- **httpx:** This is a powerful HTTP client for Python that supports asynchronous requests. It is used to interact with external APIs, fetch data, and perform web searches.
- **re (Regular Expressions):** This module provides support for regular expressions in Python. It is used to parse and match patterns in strings, which is useful for processing the AI agent's responses.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Setting Up the Environment

Step1: Installation of Required Libraries

To get started with building your AI agent, you need to install the necessary libraries. Here are the steps to set up your environment:

- **Install Python:** Ensure you have Python installed on your system. You can download it from the official Python website:
- **Set Up a Virtual Environment:** It's good practice to create a virtual environment for your project to manage dependencies. Run the following commands to set up a virtual environment:

```
python -m venv ai_agent_env
source ai_agent_env/bin/activate  # On Windows, use `ai_agent_env\Scripts\activate`
```

- **Install OpenAI API and httpx:** Use pip to install the required libraries:

```
pip install openai httpx
```

- **Install Additional Libraries:** You may also need other libraries like **re** for regular expressions, which is included in the Python Standard Library, so no separate installation is required.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Setting Up the Environment

Step2: Setting Up API Keys and Environment Variables

To use the OpenAI API, you need an API key. Follow these steps to set up your API key:

- **Obtain an API Key:** Sign up for an account on the OpenAI website and obtain your API key from the API section.
- **Set Up Environment Variables:** Store your API key in an environment variable to keep it secure. Add the following line to your .bashrc or .zshrc file (or use the appropriate method for your operating system):

```
export OPENAI_API_KEY='your_openai_api_key_here'
```

- **Access the API Key in Your Code:** In your Python code, you can access the API key using the os module:

```
import os
openai.api_key = os.getenv('OPENAI_API_KEY')
```

With the environment set up, you are now ready to start building your AI agent.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Creating the Basic Structure of the AI Agent

To build the AI agent, we will create a class that handles interactions with the OpenAI API and manages the reasoning and actions. Here's a basic structure to get started:

```python
import openai
import re
import httpx

class ChatBot:
    def __init__(self, system=""):
        self.system = system
        self.messages = []
        if self.system:
            self.messages.append({"role": "system", "content": system})

    def __call__(self, message):
        self.messages.append({"role": "user", "content": message})
        result = self.execute()
        self.messages.append({"role": "assistant", "content": result})
        return result

    def execute(self):
        completion = openai.ChatCompletion.create(model="gpt-3.5-turbo", messages=self.messages)
        return completion.choices[0].message.content
```

This class initializes the AI agent with an optional system message and handles user interactions. The __call__ method takes user messages and generates responses using the OpenAI API.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Implementing the ReAct Pattern

To implement the ReAct pattern, we need to define the loop of Thought, Action, Pause, Observation, and Answer. Here's how we can incorporate this into our AI agent:

Define the Prompt

```
prompt = """
You run in a loop of Thought, Action, PAUSE, Observation.
At the end of the loop you output an Answer.
Use Thought to describe your thoughts about the question you have been asked.
Use Action to run one of the actions available to you - then return PAUSE.
Observation will be the result of running those actions.

Your available actions are:
calculate:
e.g. calculate: 4 * 7 / 3
Runs a calculation and returns the number - uses Python so be sure to use floating point
syntax if necessary

wikipedia:
e.g. wikipedia: Django
Returns a summary from searching Wikipedia

simon_blog_search:
e.g. simon_blog_search: Django
Search Simon's blog for that term

Example session:
Question: What is the capital of France?
Thought: I should look up France on Wikipedia
Action: wikipedia: France
PAUSE

You will be called again with this:
Observation: France is a country. The capital is Paris.

You then output:
Answer: The capital of France is Paris
""".strip()
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Implementing the ReAct Pattern

To implement the ReAct pattern, we need to define the loop of Thought, Action, Pause, Observation, and Answer. Here's how we can incorporate this into our AI agent:

Define the query Function

```
action_re = re.compile('^Action: (\w+): (.*)
```

The query function runs the ReAct loop by sending the question to the AI agent, parsing the actions, executing them, and feeding the observations back into the loop.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Implementing Actions

Let us now look into the implementing actions.

Action: Wikipedia Search

The Wikipedia search action allows the AI agent to search for information on Wikipedia. Here's how to implement it:

```
def wikipedia(q):
    response = httpx.get("https://en.wikipedia.org/w/api.php", params={
        "action": "query",
        "list": "search",
        "srsearch": q,
        "format": "json"
    })
    return response.json()["query"]["search"][0]["snippet"]
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Action: Blog Search

The blog search action allows the AI agent to search for information on a specific blog. Here's how to implement it:

```
def simon_blog_search(q):
    response = httpx.get("https://datasette.simonwillison.net/simonwillisonblog.json", params={
        "sql": """
        select
          blog_entry.title || ': ' || substr(html_strip_tags(blog_entry.body), 0, 1000) as text,
          blog_entry.created
        from
          blog_entry join blog_entry_fts on blog_entry.rowid = blog_entry_fts.rowid
        where
          blog_entry_fts match escape_fts(:q)
        order by
          blog_entry_fts.rank
        limit
          1
        """.strip(),
        "_shape": "array",
        "q": q,
    })
    return response.json()[0]["text"]
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Action: Calculation

The calculation action allows the AI agent to perform mathematical calculations. Here's how to implement it:

```
def calculate(what):
    return eval(what)
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Adding Actions to the AI Agent

Next, we need to register these actions in a dictionary so the AI agent can use them:

```
known_actions = {
    "wikipedia": wikipedia,
    "calculate": calculate,
    "simon_blog_search": simon_blog_search
}
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Building the AI Agent

Integrating Actions with the AI Agent

To integrate the actions with the AI agent, we need to ensure that the query function can handle the different actions and feed the observations back into the reasoning loop. Here's how to complete the integration:

```python
def query(question, max_turns=5):
    i = 0
    bot = ChatBot(prompt)
    next_prompt = question
    while i < max_turns:
        i += 1
        result = bot(next_prompt)
        print(result)
        actions = [action_re.match(a) for a in result.split('\n') if action_re.match(a)]
        if actions:
            action, action_input = actions[0].groups()
            if action not in known_actions:
                raise Exception(f"Unknown action: {action}: {action_input}")
            print(" -- running {} {}".format(action, action_input))
            observation = known_actions[action](action_input)
            print("Observation:", observation)
            next_prompt = f"Observation: {observation}"
        else:
            return result
```

With this setup, the AI agent can reason about the input, perform actions, observe the results, and generate responses.

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Testing the AI Agent

To test the AI agent, you can run sample queries and observe the results. Here are a few examples:

```
print(query("What does England share borders with?"))
```

```
In [25]: print(query("What does England share borders with?"))

Thought: I should look up England on Wikipedia to find out which countries or regions it shares borders with.
Action: wikipedia: England
PAUSE
 -- running wikipedia England
Observation: <span class="searchmatch">England</span> is a country that is part of the United Kingdom. It is located on t
he island of Great Britain, of which it covers approximately 62%, and over
Thought: The initial summary does not provide information about the borders of England. I need to look further into the d
etails to find out which countries or regions it shares borders with.
Action: wikipedia: Geography of England
PAUSE
 -- running wikipedia Geography of England
Observation: <span class="searchmatch">Geography</span> <span class="searchmatch">of</span> Scotland <span class="searchm
atch">Geography</span> <span class="searchmatch">of</span> Wales Geology <span class="searchmatch">of</span> <span class
="searchmatch">England</span> Climate <span class="searchmatch">of</span> <span class="searchmatch">England</span> List <
span class="searchmatch">of</span> islands <span class="searchmatch">of</span> <span class="searchmatch">England</span> L
ist <span class="searchmatch">of</span> rivers <span class="searchmatch">of</span> <span class="searchmatch">England</spa
n> List <span class="searchmatch">of</span> lakes <span class="searchmatch">of</span> England
Thought: The search results are not providing a clear answer about the borders of England. I need to refine my search to
get more specific information about the borders of England.

Action: wikipedia: Borders of England
PAUSE
 -- running wikipedia Borders of England
Observation: Anglo-Scottish <span class="searchmatch">border</span> (Scottish Gaelic: Crìochan Anglo-Albannach) is an int
ernal <span class="searchmatch">border</span> <span class="searchmatch">of</span> the United Kingdom separating Scotland
and <span class="searchmatch">England</span> which runs for
Answer: England shares a border with Scotland to the north. It is also bordered by Wales to the west.
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Testing the AI Agent

```
print(query("Has Simon been to Madagascar?"))
```

```
In [27]: print(query("Has Simon been to Madagascar?"))

Thought: I should search Simon's blog to see if there are any posts about Madagascar.
Action: simon_blog_search: Madagascar
PAUSE
 -- running simon_blog_search Madagascar
Observation: Weeknotes: More releases, more museums: Lots of small releases this week.
Datasette
I released two bug fix releases for Datasette - 0.30.1 and 0.30.2. Changelog here. My Dogsheep personal analytics project
means I&#39;m using Datasette for my own data analysis every day, which inspires me to fix small but annoying bugs much m
ore aggressively.
I&#39;ve also set myself a Streak goal to land a commit to Datasette every day.
I landed a tiny new feature to master yesterday: a ?column__notin=x,y,z filter, working as an inverse of the existing ?co
lumn__in=x,y,z filter. See issue #614 for details.
More Niche Museums
I&#39;ve been keeping up my streak of adding at least one new museum to www.niche-museums.com every day. This week I adde
d the Pirates Museum in Antananarivo, Madagascar, the David Rumsey Map Center at Stanford, Galerie de Paléontologie et
d'Anatomie comparée in Paris, DEVIL-ish Little Things in Vancouver, Washington, Mardi Gras World in New Orleans, Environm
ental Volunteers EcoCenter in Palo Alto, the Evergreen A
Answer: Based on the information from Simon's blog, it appears that Simon has added the Pirates Museum in Antananarivo, M
adagascar to his niche museums website. However, there is no explicit mention of him personally visiting Madagascar.
```

```
print(query("Fifteen * twenty five"))
```

```
In [28]: print(query("Fifteen * twenty five"))

Thought: I need to perform the multiplication of fifteen and twenty-five.
Action: calculate: 15 * 25
PAUSE
 -- running calculate 15 * 25
Observation: 375
Answer: Fifteen multiplied by twenty-five is 375.
```

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya

Code and More Details at Original Source

https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Source: https://www.analyticsvidhya.com/blog/2024/07/build-ai-agents-from-scratch/

Analytics Vidhya