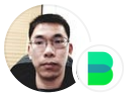
Make Langchain Agent Actually Work With Local LLMs (Vicuna, WizardLM)



gArtist · Follow

Published in Better Programming
10 min read · May 17, 2023

Listen Share

There are a ton of articles to help you build your first agent with Langchain. Those have shown good performance with OpenAI API, which is a powerful model. Then, I tried many of them and I realize that it does not actually work well with local LLMs like Vicuna or Alpaca. So, I decide to modify and optimize the Langchain agent with local LLMs. In the end, my LLM agent can do a much better job with several customizes. So here are my experiences. You can find the source code here.

But first, what are LLM Agent and Langchain?

The ChatGPT and other LLMs are really powerful and we all know that. When using ChatGPT or any LLMs, you just simply pass a text phase as input and it will respond with an answer. Depending on how good the model is, you can get a good response or a non-sense text output.

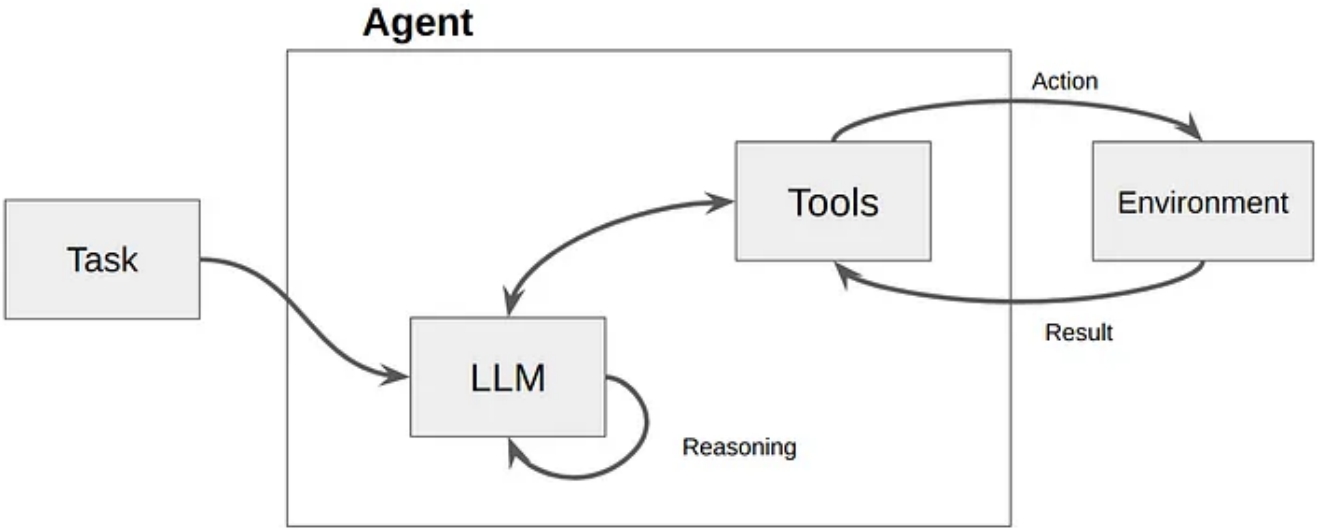
Of course, many studies show that they have a generalization ability, where they might work well even on questions or tasks that don't appear in training data. However, even if you have a really well-trained, the model itself is limited by their training data. What I mean is that you can not provide enough knowledge for them in some cases. The recent LLMs can do math pretty well, but they are not built for calculating tasks. Therefore, they will struggle with complex math equations.

Another example is the limitation of common knowledge. You take years for collecting data to train an LLM. But if you ask your LLM about the result of the game between Arsenal and Brighton yesterday, they just simply can't answer correctly. The model can't access the internet to find out, and the training data doesn't have

that information. That's the reason we should connect LLMs with other tools to expand their power. And the agent is the one we want.

An agent is a program that can "think" and interact with "the world". Given a task, it will think about how to solve it, make a plan, do some actions to the environment, receive feedback, and repeat these steps until the task is done. For example, we can see a cleaning robot as an agent. We assign it the task of cleaning the bedroom. The robot received the requirement from the human. Then, it starts to think and make a plan for cleaning. It does some actions: go to the bedroom, start to collect garbage, etc. The sensors may help it to check its location, and avoid hitting stuff. The cleaning robot can interact with and receive feedback from the environment.

An LLM agent is also the same. In this case, we exploit the LLM to help the agent can "think" since LLMs are quite good at reasoning tasks. To interact with "the world", we add some tools that can act and receive results from "the world".



This is how the LLM agent works.

Writing a custom LLM agent with Langchain

In this post, I will build an agent that receives a question, then search through the internet to find relevant information, and finally give an answer.

Okay, let's start by running your LLM. I used https://github.com/oobabooga/text-generation-webui since it provides a nice UI with many cool features. The installation is quite simple, you can check in the GitHub link. Then, download model weights from Huggingface and start the server with this command:

```
python server.py --model your_model_name --listen --api
```

For choosing the model, the most important is its size. The bigger model will provide a better answer. In my case, the 30B model can provide a decent result. The one I used in this experiment is the GPT4-X-Alpasta-30b-4bit.

You may want to apply quantization options (like llama.cpp or GPTQ) to speed up your model. Just remember to add the `--api` flag to enable the API extension, the Langchain agent will use API to interact with our LLM.

Let's build your custom LLM instance:

```
HOST = 'localhost:5000'
URI = f'http://{HOST}/api/v1/generate'

class AlpacaLLM(LLM):
    @property
    def _llm_type(self) -> str:
        return "custom"
```



```
        response = requests.post(
            URI,
            json={
                "prompt": prompt,
                "temperature": 0.7,
                "max_new_tokens": 256,
                "early_stopping": True,
                "stopping_strings": stop,
                'do_sample': True,
                'top_p': 0.1,
                'typical_p': 1,
                'repetition_penalty': 1.18,
                'top_k': 40,
                'min_length': 0,
                'no_repeat_ngram_size': 0,
                'num_beams': 1,
                'penalty_alpha': 0,
                'length_penalty': 1,
                'seed': -1,
                'add_bos_token': True,
```

```
                'truncation_length': 2048,
                'ban_eos_token': False,
                'skip_special_tokens': True,
            },
        )
        response.raise_for_status()
        return response.json()['results'][0]['text']

    @property
    def _identifying_params(self) -> Mapping[str, Any]:
        """Get the identifying parameters."""
        return {}

llm = AlpacaLLM()
```

The code is pretty clear. The `_call` function makes an API request and returns the output text from your local LLM. Only two parameters you should are `prompt` and `stop` . The `prompt` is the input text of your LLM. The `stop` is the list of stopping strings, whenever the LLM predicts a stopping string, it will stop generating text.

Now, we will do the main task: make an LLM agent. Before that, I **highly recommend** you take a look at the Langchain official document for customizing an agent. We will start to modify based on that code.

So first, we will define a tool so we can look up information from the internet.

```
from langchain.utilities import GoogleSerperAPIWrapper
import os
os.environ["SERPER_API_KEY"] = 'YOUR_API_KEY'

search = GoogleSerperAPIWrapper()
tool_google = Tool(
        name="Google Search",
        func=search.run,
        description="useful for when you need to ask with search"
    )

tool_names = [tool_google]
```

In order to do this, you may need a Serper API key. Just go to https://serper.dev/ and make an account, then you can have the API key for free. Now, you have a tool `tool_google` to search, let's try it.

```
>>>search.run('What is Langchain?')
'LangChain: Software. LangChain is a software development framework designed tc
```

Okay, it looks good.

```
template = """Below is an instruction that describes a task, paired with an inp

### Instruction:
Answer the following questions as best you can. You have access to the followir

Google Search: A wrapper around Google Search. Useful for when you need to answ

Strictly use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [Google Search]
Action Input: the input to the action, should be a question.
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

For examples:
Question: How old is CEO of Microsoft wife?
Thought: First, I need to find who is the CEO of Microsoft.
Action: Google Search
Action Input: Who is the CEO of Microsoft?
Observation: Satya Nadella is the CEO of Microsoft.
Thought: Now, I should find out Satya Nadella's wife.
Action: Google Search
Action Input: Who is Satya Nadella's wife?
Observation: Satya Nadella's wife's name is Anupama Nadella.
Thought: Then, I need to check Anupama Nadella's age.
Action: Google Search
Action Input: How old is Anupama Nadella?
Observation: Anupama Nadella's age is 50.
Thought: I now know the final answer.
Final Answer: Anupama Nadella is 50 years old.

### Input:
{input}

### Response:
{agent_scratchpad}"""
```

```
temp_Ins = """Below is an instruction that describes a task, paired with an inp

### Instruction:
Question: {thought}
Query: {query}
Observation: {observation}

### Input:
Make a short summary of useful information from the result observation that is

### Response:"""

prompt_Ins = PromptTemplate(
    input_variables=["thought", "query", "observation"],
    template=temp_Ins,
)

class CustomPromptTemplate(StringPromptTemplate):

    input_variables: List[str]
    """A list of the names of the variables the prompt template expects."""

    template: str
    """The prompt template."""

    template_format: str = "f-string"
    """The format of the prompt template. Options are: 'f-string', 'jinja2'."""

    validate_template: bool = False
    """Whether or not to try validating the template."""

    def format(self, **kwargs) -> str:
        # Get the intermediate steps (AgentAction, Observation tuples)
        # Format them in a particular way
        intermediate_steps = kwargs.pop("intermediate_steps")
        thoughts = ""
        # Refine the observation
        if len(intermediate_steps) > 0:
            regex = r"Thought\s*\d*\s*:(.*?)\nAction\s*\d*\s*:(.*?)\nAction\s*\
            text_match = intermediate_steps[-1][0].log
            if len(intermediate_steps) > 1:
                text_match = 'Thought: ' + text_match
            match = re.search(regex, text_match, re.DOTALL)
            my_list = list(intermediate_steps[-1])
            p_INS_temp = prompt_Ins.format(thought=match.group(1).strip(), quer
            my_list[1] = llm(p_INS_temp)
            my_tuple = tuple(my_list)
            intermediate_steps[-1] = my_tuple

        for action, observation in intermediate_steps:
            thoughts += action.log
            thoughts += f" {observation}\nThought:"
        # Set the agent_scratchpad variable to that value
```

```
            kwargs["agent_scratchpad"] = thoughts
            return self.template.format(**kwargs)


    prompt = CustomPromptTemplate(input_variables=["input", "intermediate_steps"],
                                  template=template,validate_template=False)
```

It looks similar to the original guideline. I modified several things to make it works well with local LLM. Here are my changes:

First, I change the prompt a bit so that can work better. An example is added so it will be a few-shot learner rather than a zero-shot learner. And it works much better with this example. And one important thing is the **instruction prompt**. Since the Alpaca models are trained with a specific format. Making a correct format with your LLM can boost performance easily.

```
Below is an instruction that describes a task, paired with an input that provic

### Instruction:
{instruction}

### Input:
{question}

### Response:
{answer}
```

However, I notice the agent is not stable with the default setting. Sometimes, the observation contains noisy text, which hurts the performance. Another problem is that observation text is quite long, it will exceed the context length quickly after several loops. So I decided to refine the observation text with `prompt_Ins` . Basically, we ask the LLM to summarize the observation related to our question. The refined short output will be returned to the agent. This trick actually **helps a lot**, especially when dealing with complicated questions.

Now, let's start to test:

```
agent = LLMSingleActionAgent(
    llm_chain=llm_chain,
    output_parser=output_parser,
    stop=["\nObservation:"],
    allowed_tools=tool_names
)
agent_executor = AgentExecutor.from_agent_and_tools(agent=agent, tools=tool_nam

agent_executor.run("Is Eminem an actor?")
```

```
**Output:**
> Entering new AgentExecutor chain...
Question: Is Eminem an actor?
Thought: First, I need to confirm if Eminem has acted in any movies or TV shows
Action: Google Search
Action Input: Has Eminem acted in any movies or TV shows?
Observation: Observation:
Yes, Eminem has acted in some movies and TV shows. He appeared in "The Wash" (2
Thought: I now know the final answer.
Final Answer: Yes, Eminem is an actor.
```

Okay, looks fine for me. Let's try with a smaller model — wizard-mega-13B-GPTQ.

```
agent_executor.run("Is Eminem an actor?")
```

```
**Output:**
> Entering new AgentExecutor chain...
Thought: To confirm if Eminem is an actor or not, we will first need to define
Action: Google Search
Action Input: What does it mean to be an actor in Hollywood?
Observation: The term "actor" refers to individuals who bring characters to lif
Thought: Based on this definition, Eminem is not considered an actor by traditi
Final Answer: No, Eminem is not an actor.
```

It works well also! But interestingly, The answer is different. How about a more complicated question?

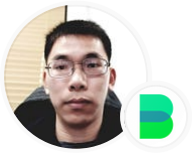
```
agent_executor.run("How much is the salary of number 8 of Manchester United?")
```

```
**Output:**
> Entering new AgentExecutor chain...
Thought: To start, we need to identify which player on Manchester United has th
Action: Google Search
Action Input: Which Manchester United player has jersey number 8?
Observations: The article discusses the history of the number 8 jersey at Manch
Thought: We still don't know how much Bruno Fernandes earns per year. Let's loc
Action: Google Search
Action Input: What is Bruno Fernandes' salary at Manchester United?
Observations: Bruno Fernandes currently plays for Manchester United Football Cl
Thought: That settles it!
Final Answer: Bruno Fernandes' annual salary at Manchester United is approximat
```

Again, it gives good reasoning and finds the answer.

P/s: I made another ReAct version with Guidance. It works better with local LLMs. Check it out at this link.

Large Language Models | NLP | Langchain | Python | AI



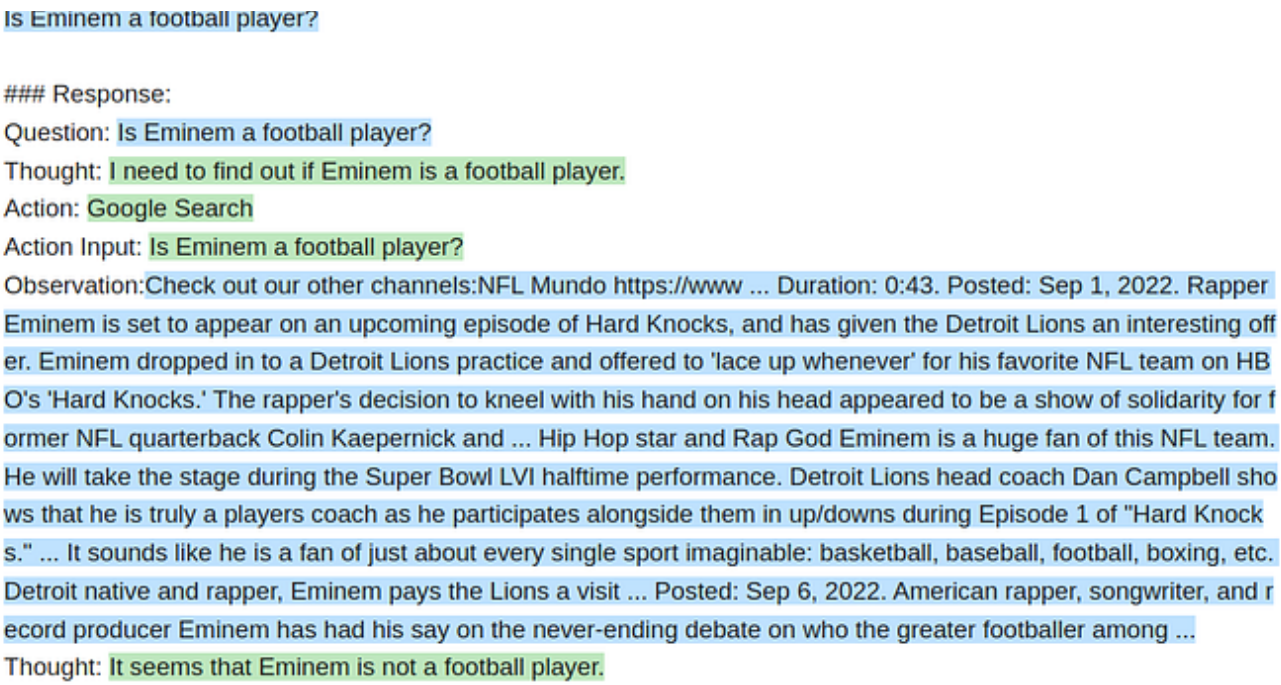
Follow

Written by gArtist

161 Followers · Writer for Better Programming

A machine who can code by himself

More from gArtist and Better Programming



 gArtist in Better Programming

A Simple Agent With Guidance and Local LLM

8 min read · Jun 5, 2023

 356  5 

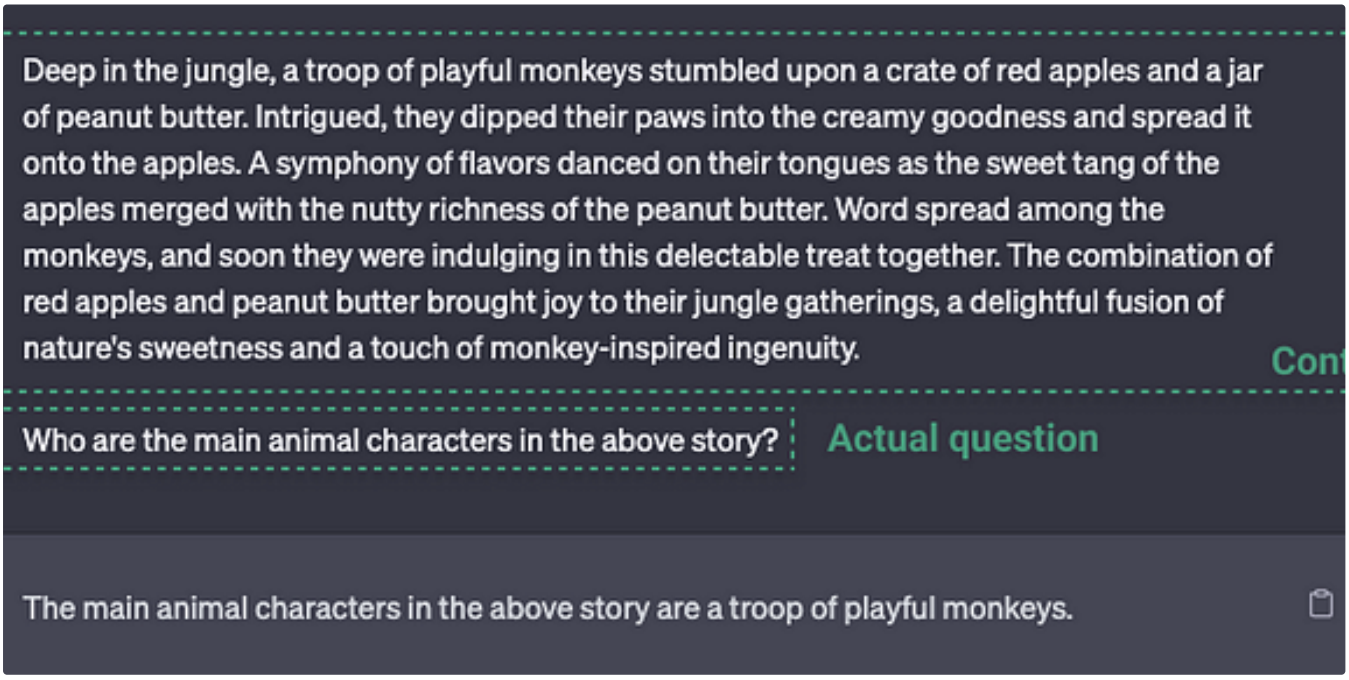
Benoit Ruiz in Better Programming

Advice From a Software Engineer With 8 Years of Experience

Practical tips for those who want to advance in their careers

22 min read · Mar 20, 2023

14.6K 271



Sami Maameri in Better Programming

Building a Multi-document Reader and Chatbot With LangChain and ChatGPT

The best part? The chatbot will remember your chat history

17 min read · May 20, 2023

1.8K 16

gArtist in Better Programming

Implementing Generative Agent With Local LLM, Guidance, and Langchain

Background is generated by [Stable Diffusion 2.1].

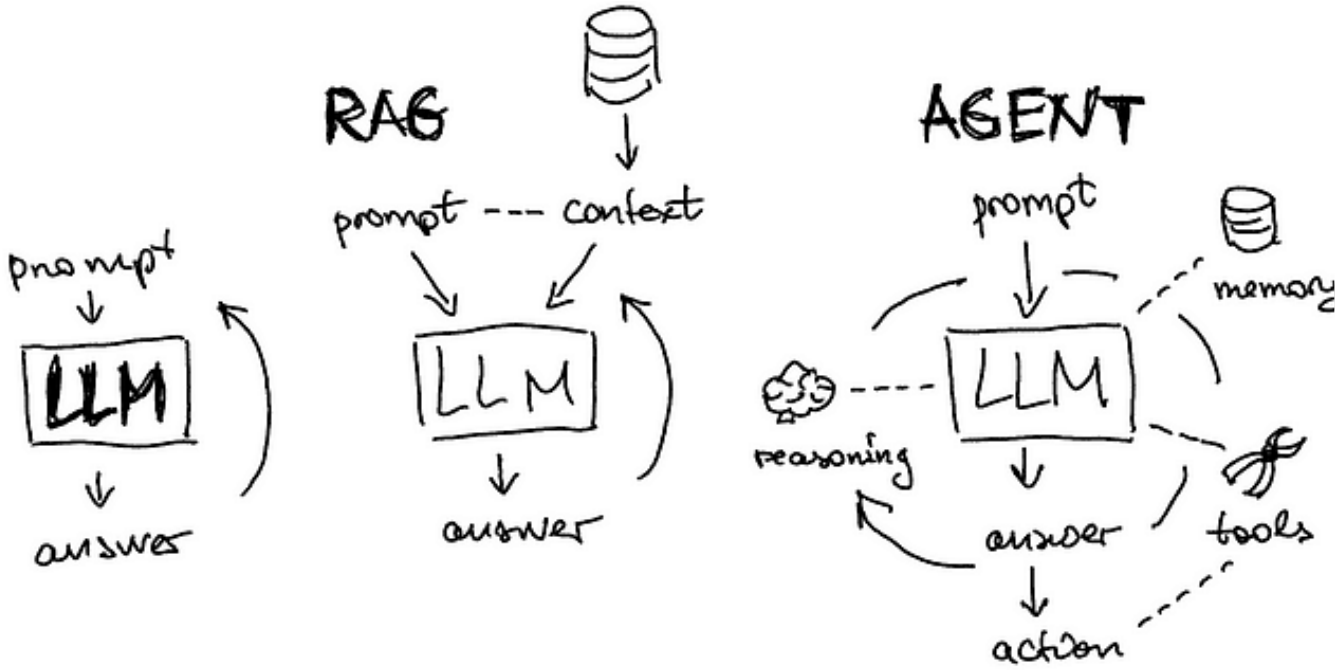
7 min read · Jun 2, 2023

66 3

See all from gArtist

See all from Better Programming

Recommended from Medium



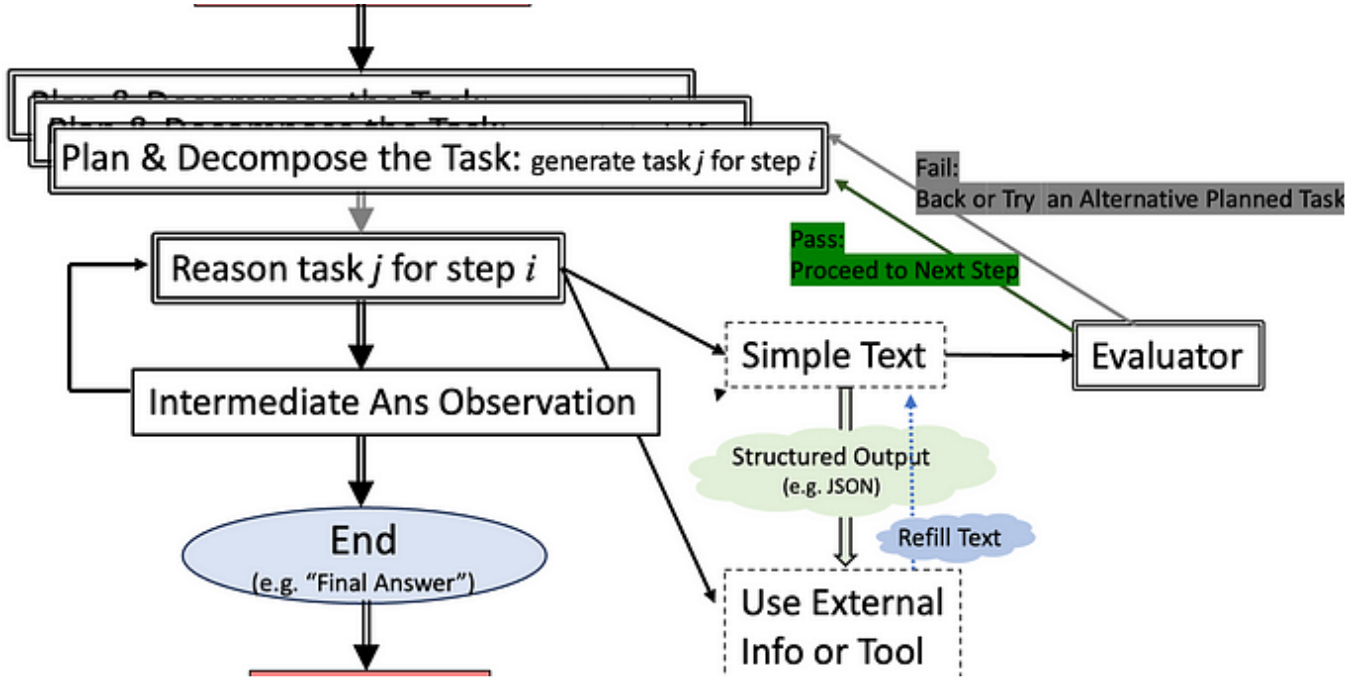
Alex Honchar in Towards Data Science

Intro to LLM Agents with Langchain: When RAG is Not Enough

First-order principles of brain structure for AI assistants

7 min read · 4 days ago

806 5



Yule Wang, PhD in The Modern Scientist

A Complete Guide to LLMs-based Autonomous Agents (Part I):

——Chain of Thought, Plan and Solve/Execute, Self-Ask, ReAct, Reflexion, Self-Consistency, Tree of Thoughts and Graph of Thoughts

20 min read · Oct 10, 2023

357 6

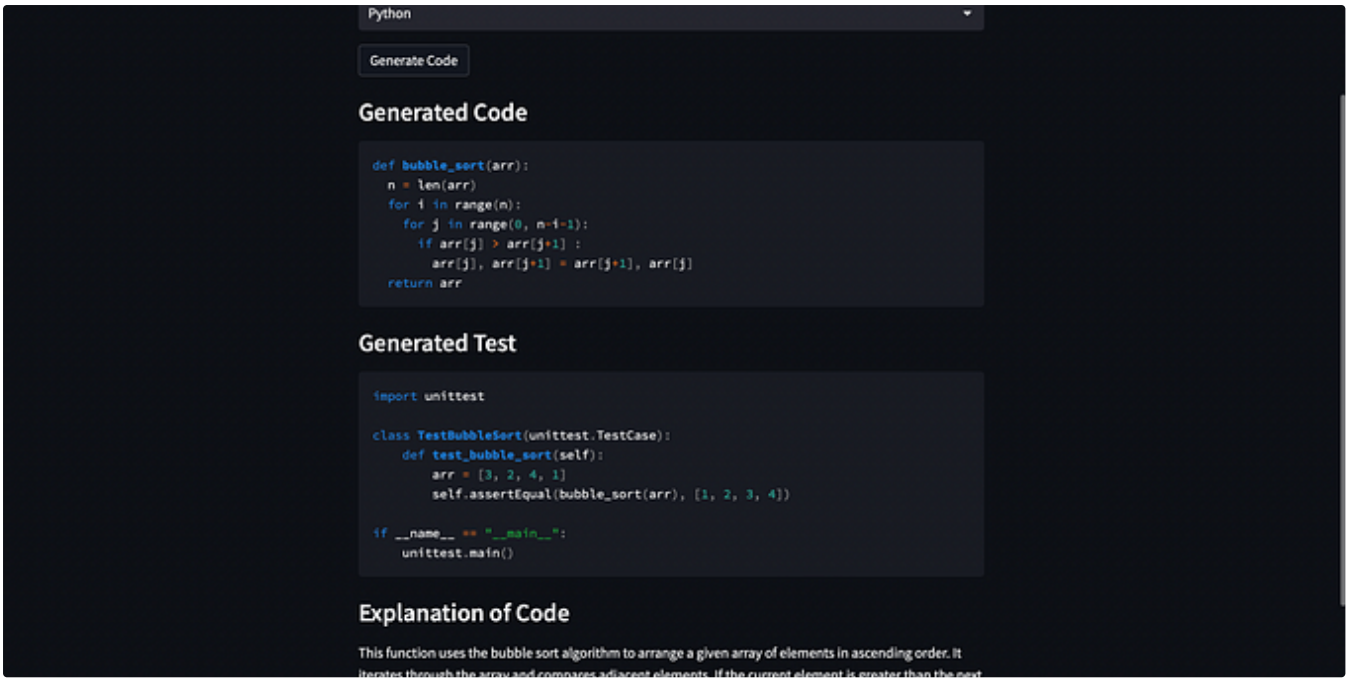
Lists

Natural Language Processing
1300 stories · 789 saves

Coding & Development
11 stories · 514 saves

The New Chatbots: ChatGPT, Bard, and Beyond
12 stories · 338 saves

ChatGPT prompts
47 stories · 1288 saves

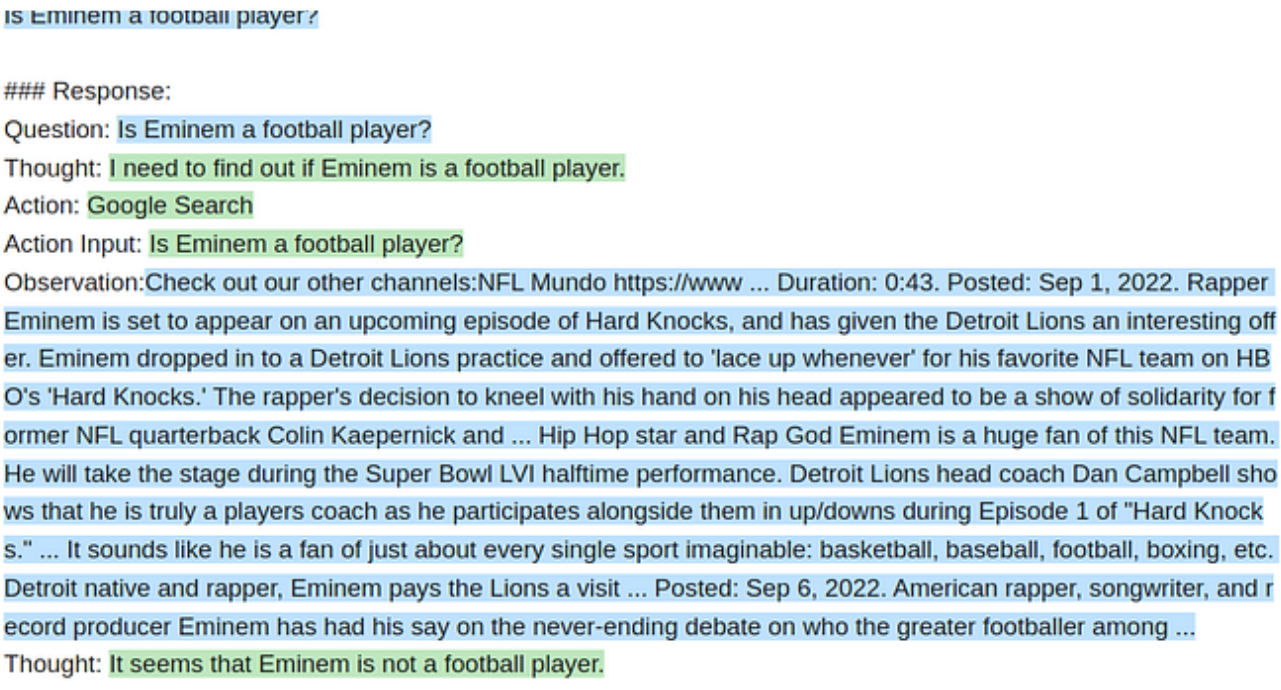


Charaka Abeywickrama

Introducing LangChain 🦜🔗: Building an App for Automated Code Generation

LangChain is a powerful tool for building language model applications, and in this blog, we'll explore how you can use LangChain to create...

5 min read · Nov 17, 2023


Is Eminem a football player?

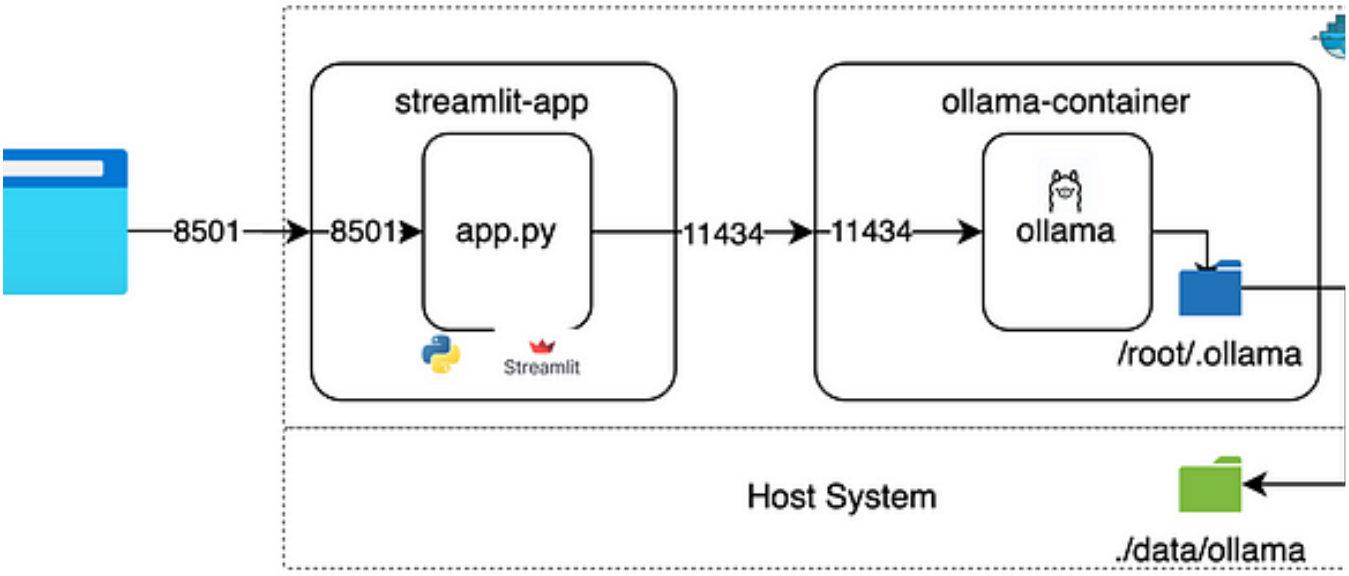
Response:
Question: Is Eminem a football player?
Thought: I need to find out if Eminem is a football player.
Action: Google Search
Action Input: Is Eminem a football player?
Observation:Check out our other channels:NFL Mundo https://www ... Duration: 0:43. Posted: Sep 1, 2022. Rapper Eminem is set to appear on an upcoming episode of Hard Knocks, and has given the Detroit Lions an interesting off er. Eminem dropped in to a Detroit Lions practice and offered to 'lace up whenever' for his favorite NFL team on HB O's 'Hard Knocks.' The rapper's decision to kneel with his hand on his head appeared to be a show of solidarity for f ormer NFL quarterback Colin Kaepernick and ... Hip Hop star and Rap God Eminem is a huge fan of this NFL team. He will take the stage during the Super Bowl LVI halftime performance. Detroit Lions head coach Dan Campbell sho ws that he is truly a players coach as he participates alongside them in up/downs during Episode 1 of "Hard Knock s." ... It sounds like he is a fan of just about every single sport imaginable: basketball, baseball, football, boxing, etc. Detroit native and rapper, Eminem pays the Lions a visit ... Posted: Sep 6, 2022. American rapper, songwriter, and r ecord producer Eminem has had his say on the never-ending debate on who the greater footballer among ...
Thought: It seems that Eminem is not a football player.

gArtist in Better Programming

A Simple Agent With Guidance and Local LLM

8 min read · Jun 5, 2023

356 5



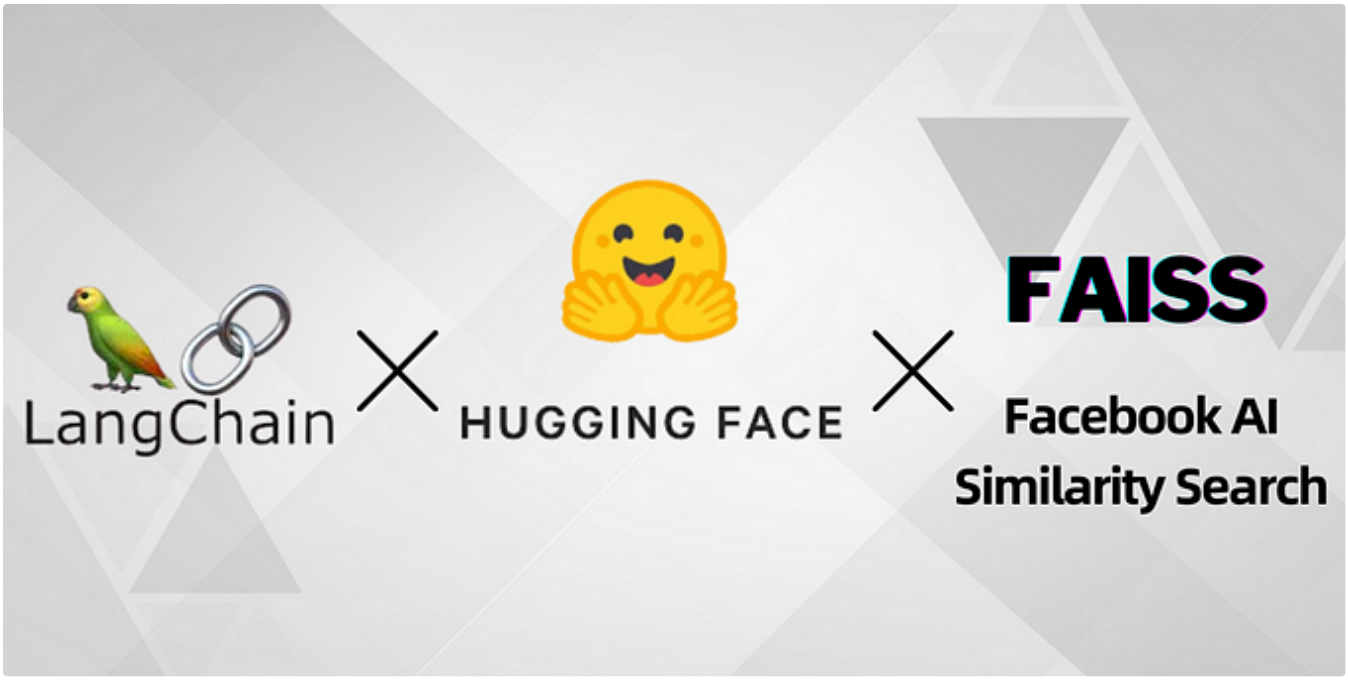
A B Vijay Kumar

Ollama—Build a ChatBot with Langchain, Ollama & Deploy on Docker

Working with Ollama to run models locally, build LLM applications that can be deployed as docker containers.

5 min read · Feb 21, 2024

254



Charu Makhijani in Towards AI

LangChain: Develop LLM-powered applications using LangChain, Hugging Face, and Facebook AI...

LangChain Tutorial to build powerful applications using Large Language Models

14 min read · Jan 3, 2024