Report on Card 10 - Pratica: Agentes com LangChain (III)

Marcus Vinicius Oliveira Nunes

1. Activity Description

1.1 Como usar a lib LangChain para criar um agente

In this card it was taught how to build a simple LangChain agent, integrated with a LLM model and some tools created by the community. Differently from a chain, an agent doesn't work with static inputs, I don't need to specify a specific parameter like in the chain from Card 9, which expects an input of the type of animal and its color. The agent takes an input and it decides on what to do with taking the tools I gave to it.

```python
from dotenv import load_dotenv
from langchain import hub
from langchain_community.chat_models import ChatOllama
from langchain_community.tools.youtube.search import YouTubeSearchTool
from langchain_community.tools.google_trends import GoogleTrendsQueryRun
from langchain_community.utilities.google_trends import GoogleTrendsAPIWrapper
from langchain.agents import create_openai_functions_agent, AgentExecutor

load_dotenv()

# Tools
youtubeTool = YouTubeSearchTool()
google_trends = GoogleTrendsQueryRun(api_wrapper=GoogleTrendsAPIWrapper())
tools = [youtubeTool, google_trends]


# Prompt

prompt = hub.pull('hwchase17/openai-functions-agent')

# LLM

llm = ChatOllama(model ="mistral", temperature = 0)

# Juntar Tools, Prompt, LLM, Agent

meu_agente = create_openai_functions_agent(llm,tools, prompt)

# Executor

agent_executor = AgentExecutor(agent= meu_agente, tools= tools, verbose= True)

agent_executor.invoke({'input': 'Qual o valor medio do interesse mais recente pelo termo llms'})
```

An agent is composed by these main components:

- **Tools:** They are a set of tools as the name implies that are used by the agent to perform actions and help with a better answer, in this case I'm using a

youtube tool that can help me retrieve information from youtube, I'm also using google trends.

- **LLM:** It's the brain of any agent, it's the llm model that gives me the answers.
- **Prompt:** prompt is the one who defines how the model needs to act and how to behave.
- **Executor:** Is the one who will execute the agent that was built.

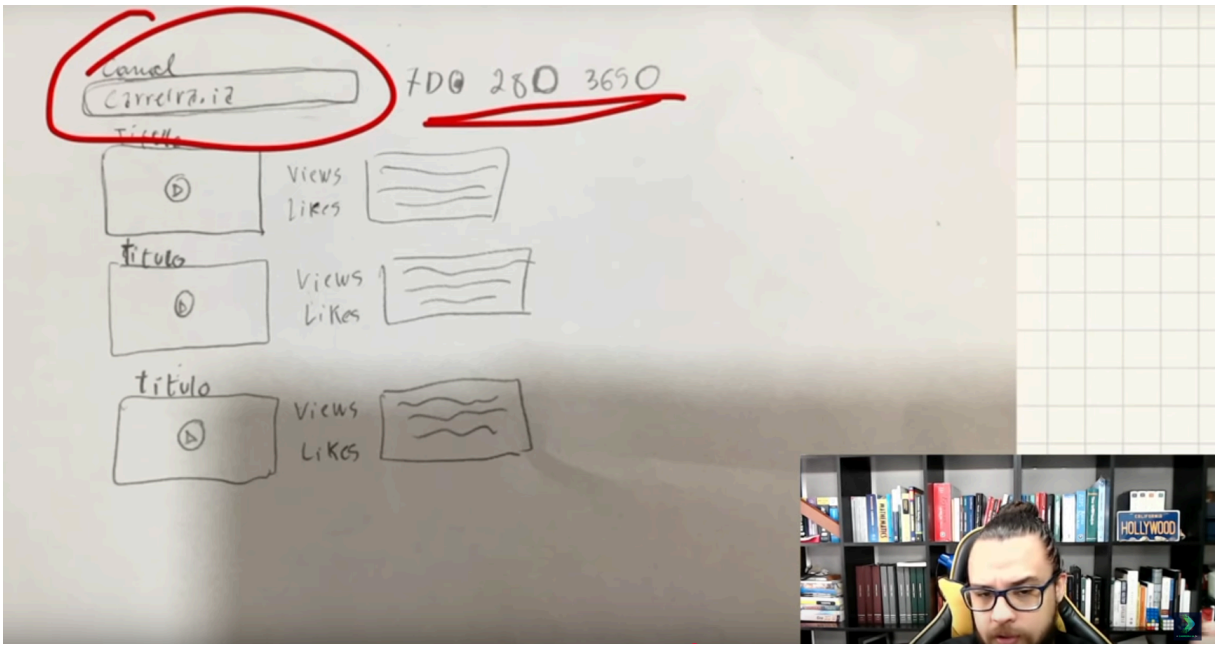
If I desire, I can pass more tools that the model can work with, in this case I passed the wikipedia:

```python
from dotenv import load_dotenv
from langchain import hub
from langchain_community.chat_models import ChatOllama
from langchain_community.tools.youtube.search import YouTubeSearchTool
from langchain_community.tools.google_trends import GoogleTrendsQueryRun
from langchain_community.utilities.google_trends import GoogleTrendsAPIWrapper
from langchain_community.tools import WikipediaQueryRun
from langchain_community.utilities import WikipediaAPIWrapper
from langchain.agents import create_openai_functions_agent, AgentExecutor

load_dotenv()

# Tools
youtubeTool = YouTubeSearchTool()
google_trends = GoogleTrendsQueryRun(api_wrapper=GoogleTrendsAPIWrapper())
wikipediaTool = WikipediaQueryRun(api_wrapper= WikipediaAPIWrapper())
tools = [youtubeTool, google_trends, wikipediaTool]


# Prompt

prompt = hub.pull('hwchase17/openai-functions-agent')

# LLM

llm = ChatOllama(model ="mistral", temperature = 0)

# Juntar Tools, Prompt, LLM, Agent

meu_agente = create_openai_functions_agent(llm,tools, prompt)

# Executor

agent_executor = AgentExecutor(agent= meu_agente, tools= tools, verbose= True)

agent_executor.invoke({'input': 'me de uma pesquisa da wikipedia sobre bethoven'})
```

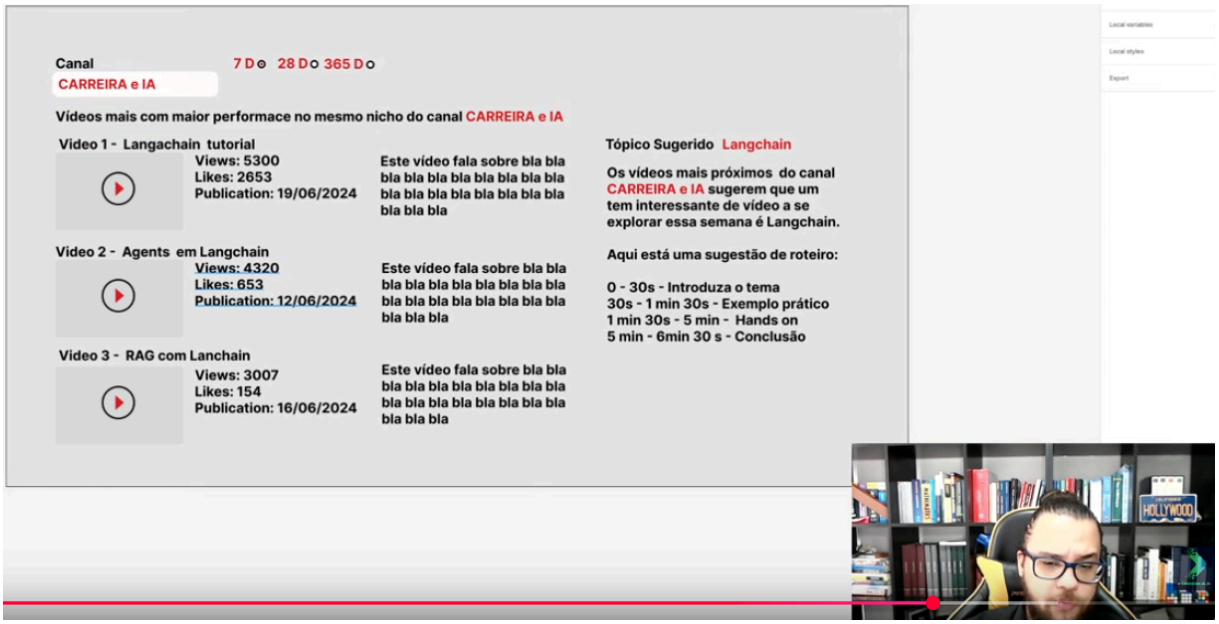
1.2 Como fazer um projeto prático com Langchain, Chatgpt, Python, RAG

In the second video, it's shown a set of ideas and knowledge that we must have before building an application that integrates LLM models. It's a self experience video that the teacher has based on what he has learned throughout the years as a

developer. When developing an application we must have in mind what kind of problem we are working with and what we need to do to solve it, it's also nice to have some kind of planning as well, not only in mind but described on some paper:



After putting the idea on paper we can make our project on some kind of screen or blue paper



2. Conclusion

Both videos are pretty useful since they teach how to use langchain to build an agent that does work for us and how to apply this knowledge into some kind of project in a good way.

3. References

Como fazer um projeto prático com Langchain, Chatgpt, Python, RAG

Como usar a lib langchain para cria um agente - criando o seu chatgpt person…