# Prompt 102
# Intro to LangChain

By Rod Soto
@rodsoto

02-24-2024

# $whoami

Rod Soto

@rodsoto

rodsoto.net

https://github.com/rsfl/

# RECAP

## LLMs

A Large Language Model (LLM) like the one you're interacting with now (ChatGPT, based on GPT-4) is an advanced type of artificial intelligence model designed to understand, generate, and respond to human language. These models are "large" in the sense that they are trained on vast amounts of text data and have a significant number of parameters, which are the aspects of the model that have been learned from training data.

## Generative AI

Generative AI refers to a subset of artificial intelligence technologies that are capable of generating new content, be it text, images, music, or other forms of media. This is distinct from other AI technologies that are primarily focused on understanding or interpreting existing content.

# Key aspects of generative AI

Content Creation: It can generate novel content based on the patterns and structures it has learned from its training data.
.
Deep Learning Models: Generative AI typically relies on advanced deep learning models, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformer models like GPT (Generative Pretrained Transformer).
Generative Adversarial Networks (GANs): This is a popular type of model used in generative AI. It involves two neural networks, a generator and a discriminator, which are trained simultaneously. The generator learns to produce increasingly realistic outputs, while the discriminator learns to better distinguish between real and generated outputs.

Applications: The applications of generative AI are diverse and growing. They include creating art, designing products, generating realistic human-like text for chatbots, developing new video game environments, aiding in film and video production, and more.

Customization and Personalization: Generative AI can be used to tailor content to individual preferences or specific requirements, making it valuable in marketing, personalized education, and user experience enhancement.

Ethical and Societal Implications: As with other AI technologies, generative AI raises important questions about ethics, including concerns about authenticity, the potential for misuse (such as deepfakes), copyright issues, and the impact on creative industries.
Data Dependency

# Prompt Engineering

Prompt Engineering is a comprehensive process of designing and optimizing specific and unambiguous prompts for a Large Language Model to ensure that it generates relevant, accurate and coherent responses.

Emiliano Viotti

Prompts usually contain:

- Instructions
- Examples
- Inputs

# Prompt Engineering Recap - Techniques

| Technique | Description | Key Idea | Performance Considerations |
|---|---|---|---|
| Zero-Shot Prompting | No examples provided; rely on the model's training | Leverages the model's pre-training | Works for simple tasks, but struggles with complex reasoning |
| Few-Shot Prompting | Provides a few demos of input and desired output | Shows desired reasoning format | Tripled accuracy on grade-school math |
| CoT | Prefix responses with intermediate reasoning steps | Gives the model space to reason before answering | Quadrupled accuracy on a math dataset |
| Least-to-Most Prompting | Prompts the model for simpler subtasks first | Decomposes a problem into smaller pieces | Boosted accuracy from 16% to 99.7% on some tasks |
| Self-Consistency | Picks the most frequent answer from multiple samples | Increases redundancy | Gained 1–24 percentage points across benchmarks |
| Chain-of-Density | Iteratively creates dense summaries by adding entities | Generates rich, concise summaries | Improves information density in summaries |
| Chain-of-Verification (CoV) | Verifies an initial response by generating and answering questions | Mimics human verification | Enhances robustness and confidence |
| Active Prompting | Picks uncertain samples for human labeling as examples | Finds effective few-shot examples | Improves few-shot performance |
| Tree-of-Thought | Generates and automatically evaluates multiple responses | Allows backtracking through reasoning paths | Finds an optimal reasoning route |
| Verifiers | Trains a separate model to evaluate responses | Filters out incorrect responses | Lifted grade-school math accuracy by ~20 percentage points |
| Fine-Tuning | Fine-tunes on an explanation dataset generated via prompting | Improves the model's reasoning abilities | 73% accuracy on a commonsense QA dataset |

From Generative AI with Langchain
https://amazon.com/Generative-AI-LangChain-language-ChatGPT-ebook/dp%20/B0CB%20BL55PQ/ref=sr_1_1?crid=1EFOS0W3YU3SL&keywords=generative+ai+with+langchain&qid=1708037000&sprefix=generative+ai+with+langchai%2Caps%2C358&sr=8-1

# Current Limitations of LLMs as of 02-2024

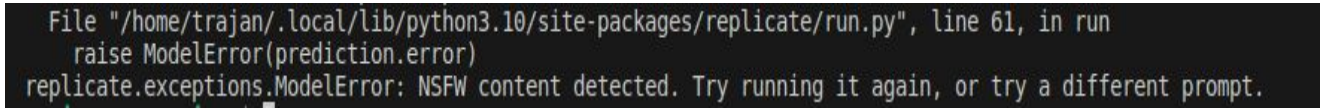
**Outdated knowledge**: most of these LLMs rely on training data without external integration they cannot produce recent real world information.

**Inability to perform certain actions:** In many instances LLMs cannot perform, searches, calculations or lookups.

**Limited Context:**

**Hallucination Risks:** with no context or external integration -> MSU

**Biases & Discrimination:**




**Lack of transparency:** No idea how evaluating criteria is set.

·**Generative AI with Langchain**

ISBN-13

**978-1835083468**

# Stochastic Parrot

LLMs can produce convincing language but lack any true comprehension of the meaning behind words.

Coined by Emily Bender, T Gebru, Margaret Mitchell, A MacMillan 2021

# What is Langchain

LangChain is a framework for developing applications powered by language models. It enables applications that:

- Are context-aware: connect a language model to sources of context (prompt instructions, few shot examples, content to ground its response in, etc.)
- Reason: rely on a language model to reason (about how to answer based on provided context, what actions to take, etc.)

This framework consists of several parts.

- LangChain Libraries: The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents.
- LangChain Templates: A collection of easily deployable reference architectures for a wide variety of tasks.
- LangServe: A library for deploying LangChain chains as a REST API.
- LangSmith: A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.

https://python.langchain.com/docs/get_started/introduction

# Langchain

Advantages of LangChain

- Integration to LLMs
- Modular Design
- Chain of Thought Prompting
- Ease of Use
- Community & Collaboration
- Application Development
- Flexibility

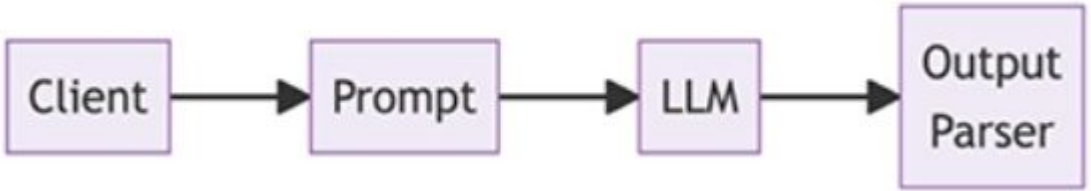
# Our goal today is to interact with LLMs via Langchain

We are going to use OpenAI and HuggingFace . You will need either vs code with python3 or jupyter also you will need API keys of the LLM services with are going to use. We will walk through on how to get them

***Generative AI with Langchain**

ISBN-13

978-1835083468



# LangChain Tools

- Machine Translator
- Calculator
- Maps
- Weather
- Stocks
- Slides
- Table Processing
- Knowledge Graphs
- Search Engine
- Wikipedia
- Online Shopping

# Langchain can also load documents

Langchain also help load documents for processing, and subsequent prompting.

We will do that by importing a pdf (2 are provided in the github repository) and then asking questions about it.  Langchain can load other type files such as webpages, videos.

# Other Frameworks like Langchain

- Microsoft AutoGen: AutoGen is a framework that enables the development of LLM applications using multiple agents that can converse with each other to solve tasks
- Optimus SuperAGI: an open-source platform that provides infrastructure for building autonomous AI agents.
- Llama (Meta LLM)

There is also a GUI version of Langchain called LangFlow which i found very frustrating trying to use it so it will not be shown today.

# Let’s get started with Langchain

https://python.langchain.com/docs/get_started/installation

git clone https://github.com/rsfl/022024AI

pip install langchain

```
pip install langchain-core
```

```
pip install langchain-community
```

```
pip install langchain-cli
```

```
pip install "langserve[all]"
```

```
pip install langchain-openai
```

# Get your OpenAI API Key

https://www.howtogeek.com/885918/how-to-get-an-openai-api-key/

Then in your ide (vs code or jupyter)

```
Obtain your API keys for OpenAI, HuggingFace, Replicate, Tavilly Search,

The scripts were written in a way to make it easy to input such keys, but  you should not put them openly in code and should use other mechanisms to protect your keys such as dotenv or create a config.py file then import it. This is out of the scope of this presentation.
```

# What we are going to do today

Clone the repo https://github.com/rsfl/022024AI

pip install requirements.txt

open the .py files in vs code or jupyter and add your apis

We will have multiple hands on examples

1)Simple LLM prompts (OpenAI, HuggingFace,Replicate, Tavilly Search)

2)Use PromptTemplate against LLM

3)Langchain Chains

4) Agent & Internet

5) Memory Conversation

6) Import a pdf book and ask questions about it

7) Execute Prompt Techniques via langchain in Python

8) Use Text2image prompts

9) DuckDuckGo & Wikipedia

# Langchain LLMs

We will prompt directly to LLMs

You can modify the prompt to ask a different question or try different models by Hugging Face HUB

https://huggingface.co/models #take a look at the hugging face models

# LangChain PromptTemplates

You can use Prompt Templates to get an answer from an LLM

Play with the template file

Play with the different templates

https://python.langchain.com/docs/templates/

# Langchain Chains

Play with the Chains Langchain model

Chains refer to sequences of calls - whether to an LLM, a tool, or a data preprocessing step.  Play with the Chains file in the repo.

Here is a link to experiment with the Chains module.

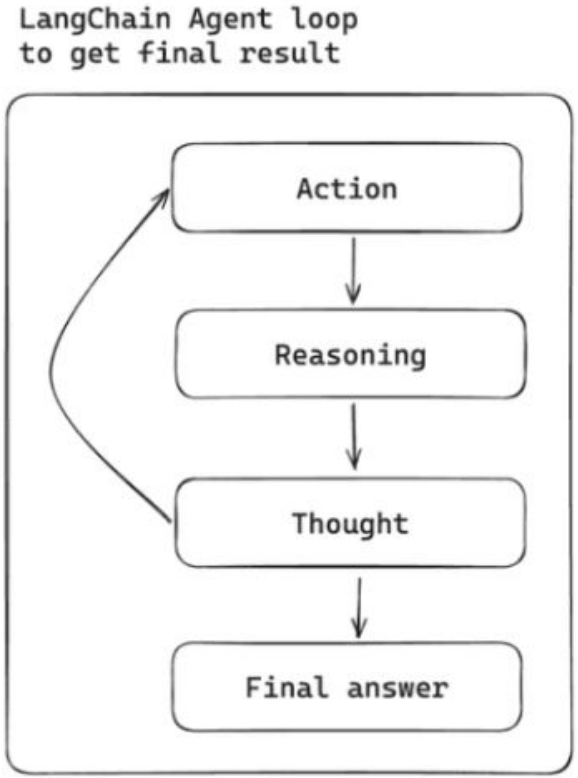
Here is jupyter notebook for additional info https://github.com/pinecone-io/examples/blob/master/learn/generation/langchain/handbook/02-langchain-chains.ipynb (you will need pinecone libs)

# Agent & Internet

We can enhance the responses from LLMs by connecting to the internet using Tavily search module, play with the python file and modify some of the sequences and observe how the agents  have “thoughts” as it tries to respond your prompt.

An agent is an entity integrating a chain that helps accomplish a task by executing actions

https://brightinventions.pl/blog/introducing-langchain-agents-tutorial-with-example/



# Memory Conversation

Conversational memory is how a chatbot can respond to multiple queries in a chat-like manner. It enables a coherent conversation, and without it, every query would be treated as an entirely independent input without considering past interactions.

Play with the memory example file and adjust it at your will.

Here is a good blog about langchain memory conversation

https://medium.com/@michael.j.hamilton/conversational-memory-with-langchain-82c25e23ec60

# Tree of Thoughts

The Tree-of-Thought (ToT) technique takes inspiration from the way human minds solve complex reasoning tasks by trial and error. In this approach, the mind explores the solution space through a thought process resembling a tree, enabling backtracking when needed.

You have a file with a great example of Tree-of-Thought technique. Play with it modify it and add more as you get familiar with Langchain.

Here is a blog explaining the rationale behind this prompt technique

https://medium.com/@astropomeai/implementing-the-tree-of-thoughts-in-langchains-chain-f2ebc5864fac

# Other Prompt Techniques

- Zero shot: Prompt directly without examples or context.
- Few-Shot: Provide LLM with a few demonstrations of desired output
- Chain of Thought: Provide LLM with intermediate steps
- Self Consistency: Pick the most frequent answer from multiple responses
- Tree of Thought: Evaluates multiple responses and generates an answer

# begin.py

```python
from langchain_community.llms import OpenAI

# Initialize the language model
llm = OpenAI(temperature=0.9)

text = "What is Hackmiami?"

# Generate the response using the language model
try:
    response = llm.generate([text])  # Wrap the text in a list
    print(response)
except Exception as e:
    print(f"An error occurred: {e}")
```

```
An error occurred: Argument `prompts` is expected to be of type List[str], received argument of type <class 'str'>.
trajan@docker-ptf:~$ /usr/bin/python3 /home/trajan/Desktop/022024AI/begin.py
/home/trajan/.local/lib/python3.10/site-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The clas
s `langchain_community.llms.openai.OpenAI` was deprecated in langchain-community 0.0.10 and will be removed in 0.2.0. An updat
ed version of the class exists in the langchain-openai package and should be used instead. To use it run `pip install -U langc
hain-openai` and import as `from langchain_openai import OpenAI`.
  warn_deprecated(
generations=[[Generation(text='\n\nHackmiami is a non-profit organization dedicated to promoting ethical hacking and informati
on security education. It organizes conferences, workshops, and other events to bring together hackers, security professionals
, and enthusiasts for networking, learning, and collaboration. The organization also provides resources and support for those
interested in learning about cybersecurity and ethical hacking.', generation_info={'finish_reason': 'stop', 'logprobs': None})
]] llm_output={'token_usage': {'completion_tokens': 65, 'prompt_tokens': 6, 'total_tokens': 71}, 'model_name': 'gpt-3.5-turbo-
instruct'} run=[RunInfo(run_id=UUID('ba0b5b54-b58a-42c3-a9da-2382389211eb'))]
```

# beginhuggingface.py

```python
from langchain_community.llms import OpenAI
from langchain_openai import OpenAI
from langchain import HuggingFaceHub

llm = llm = HuggingFaceHub(repo_id="google/flan-t5-base", model_kwargs={"temperature":1, "max_length":64})

text = "What is the capital of Germany?"      You, 1 second ago • Uncommitted changes
print(llm(text))
```

```
/home/trajan/.local/lib/python3.10/site-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The func
tion `__call__` was deprecated in LangChain 0.1.7 and will be removed in 0.2.0. Use invoke instead.
  warn_deprecated(
berlin
```

# basicprompttemplate.py

```python
from langchain_community.llms import OpenAI
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
from langchain.memory import ConversationBufferWindowMemory

# Initialize the language model
llm = OpenAI()

prompt_template = PromptTemplate.from_template(
    template="Write a {length} story about: {content}"
)      trajan, last week • pyfiles

prompt = prompt_template.format(
    length="2-sentence",
    content="The hometown of the legendary nuclear scientist, J. Robert Oppenheimer"
)

response = llm.predict(text=prompt)

print(response)
```

```
J. Robert Oppenheimer's hometown was filled with pride as they watched the renowned scientist make groundbreaking discoveries
in the field of nuclear physics, solidifying his place in history.
```

# agentsinternet.py

```python
# libraries
import os

from langchain.agents import AgentType, initialize_agent
from langchain.chat_models import ChatOpenAI
from langchain.tools.tavily_search import TavilySearchResults
from langchain.utilities.tavily_search import TavilySearchAPIWrapper
from langchain import FewShotPromptTemplate


# set up API key
os.environ["TAVILY_API_KEY"] =
os.environ['OPENAI_API_KEY'] =       You, 1 second ago • Uncommitted change

# set up the agent
llm = ChatOpenAI(model_name="gpt-4", temperature=0.5)
search = TavilySearchAPIWrapper()
tavily_tool = TavilySearchResults(api_wrapper=search)

# initialize the agent
agent_chain = initialize_agent(
    [tavily_tool],
    llm,
    agent=AgentType.STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
)

# run the agent
agent_chain.run(
    "What is the JellyFish UAP?",
```

```
'}, {'url': 'https://www.newsnationnow.com/space/ufo/marine-speaks-jellyfish-uap/', 'content': '( NewsNation) — A Marine offic
er spoke with NewsNation about footage of what\'s been dubbed a "jellyfish" UAP (unknown aerial phenomenon) spotted in Iraq ar
ound 2018. Investigative journalist Jeremy Corbell released the video earlier this week of a jellyfish-looking object over a m
ilitary base.'}, {'url': 'https://www.newsnationnow.com/space/ufo/pentagon-jellyfish-uap-video/', 'content': 'Tom Palmer. ( Ne
wsNation) — The U.S. Department of Defense (DOD) released a statement Wednesday in response to questions about a newly surface
d video of an alleged unidentified anomalous phenomenon ( UAP ). What\'s now been dubbed a "jellyfish" UAP was allegedly recor
ded by the U.S. military over a U.S. joint operations base in Iraq ...'}, {'url': 'https://www.youtube.com/watch?v=ncGu2CRqlmU
', 'content': '1.14M subscribers Subscribe Subscribed 7.9K 431K views 11 days ago #VargasReports #UAP #Jellyfish The U.S. Depa
rtment of Defense released a statement Wednesday in response to questions about a...'}]
Thought:The JellyFish UAP refers to an Unidentified Aerial Phenomenon that was reportedly seen over a U.S. operations base in
Iraq. Named "the jellyfish" UAP due to its appearance, the video of this phenomenon was released by investigative journalist J
eremy Corbell. The U.S. Department of Defense has been asked about this, but they have not verified the video's authenticity.
Action:
```
{
  "action": "Final Answer",
  "action_input": "The JellyFish UAP refers to an Unidentified Aerial Phenomenon that was reportedly seen over a U.S. operatio
ns base in Iraq. Named \"the jellyfish\" UAP due to its appearance, the video of this phenomenon was released by investigative
 journalist Jeremy Corbell. The U.S. Department of Defense has been asked about this, but they have not verified the video's a
uthenticity."
}
```


> Finished chain.
```

# langchainchains.py

```python
from langchain_community.llms import OpenAI
#from langchain import HuggingFaceHub
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
from langchain.memory import ConversationBufferWindowMemory

llm =  OpenAI()

prompt = """Question: Can we time travel?

Let's think step by step.

Answer: """
llm(prompt)


template = """Question: {question}

Let's think step by step.

Answer: """

prompt = PromptTemplate(template=template, input_variables=["question"])


llm_chain = LLMChain(prompt=prompt, llm=llm)

question = "Can we time travel?"

print(llm_chain.run(question))
```

```
In theory, time travel is possible through the manipulation of space-time, which is the fabric of the universe that combines t
he three dimensions of space and one dimension of time. According to Einstein's theory of relativity, space and time are inter
connected and can be warped by massive objects, such as planets and stars.

One way to potentially time travel is through the use of wormholes, which are hypothetical tunnels through space-time that con
nect two distant points in the universe. If a person were to enter a wormhole and emerge at the other end, they could potentia
lly travel to a different time in the past or future.

Another concept that has been explored is time dilation, which is the idea that time moves at different rates for objects in d
ifferent gravitational fields. This has been observed in experiments with high-speed particles and in the strong gravitational
 fields near black holes.

However, these concepts are still theoretical and have not been proven or tested on a large scale. The technology and knowledg
e required for time travel are currently beyond our capabilities. It is also important
trajan@docker-ptf:~$
```

# LLMemory.py

```python
import os
os.environ['OPENAI_API_KEY'] = '
from langchain_community.chat_models import ChatOpenAI
from langchain.chains import ConversationChain
from langchain.memory import ConversationBufferMemory

llm = ChatOpenAI(temperature=0.2)
memory = ConversationBufferMemory()
conversation = ConversationChain(llm=llm, memory=memory, verbose=False)

print(conversation.predict(input="Is Start Wars a good movie?"))

print(conversation.predict(input="Who is the main bad guy?"))

print(conversation.predict(input="I heard Han Solo died is that true?"))
```

```
warn_deprecated(
an AI, I don't have personal opinions or experiences, but I can provide you with information. "Star Wars" is a highly popul
and influential science fiction franchise created by George Lucas. The original film, released in 1977, was a groundbreakin
success and is widely regarded as a classic. It has since spawned numerous sequels, prequels, spin-offs, and an expanded uni
rse of books, comics, and TV shows. The franchise has a large and dedicated fan base, and many people consider the original
ilogy, consisting of "Star Wars: Episode IV - A New Hope," "Star Wars: Episode V - The Empire Strikes Back," and "Star Wars:
isode VI - Return of the Jedi," to be excellent movies. However, opinions on the newer films and spin-offs may vary. Ultima
ly, whether "Star Wars" is a good movie or not is subjective and depends on individual tastes and preferences.
the "Star Wars" franchise, the main bad guy is Darth Vader. He is a Sith Lord and a central character in the original trilo
. Darth Vader is known for his iconic black armor, deep voice, and use of the Force. He serves as the primary antagonist, in
ially working for the Galactic Empire and later revealed to be the father of the main protagonist, Luke Skywalker. Darth Vad
's presence and actions throughout the franchise make him one of the most recognizable and memorable villains in cinematic h
tory.
s, that is true. In the "Star Wars" sequel trilogy, specifically in "Star Wars: Episode VII - The Force Awakens," Han Solo,
rtrayed by Harrison Ford, dies. Han Solo is a beloved character and one of the main protagonists in the original trilogy. Hi
death was a significant moment in the storyline and had a profound impact on the other characters and the overall narrative.
```

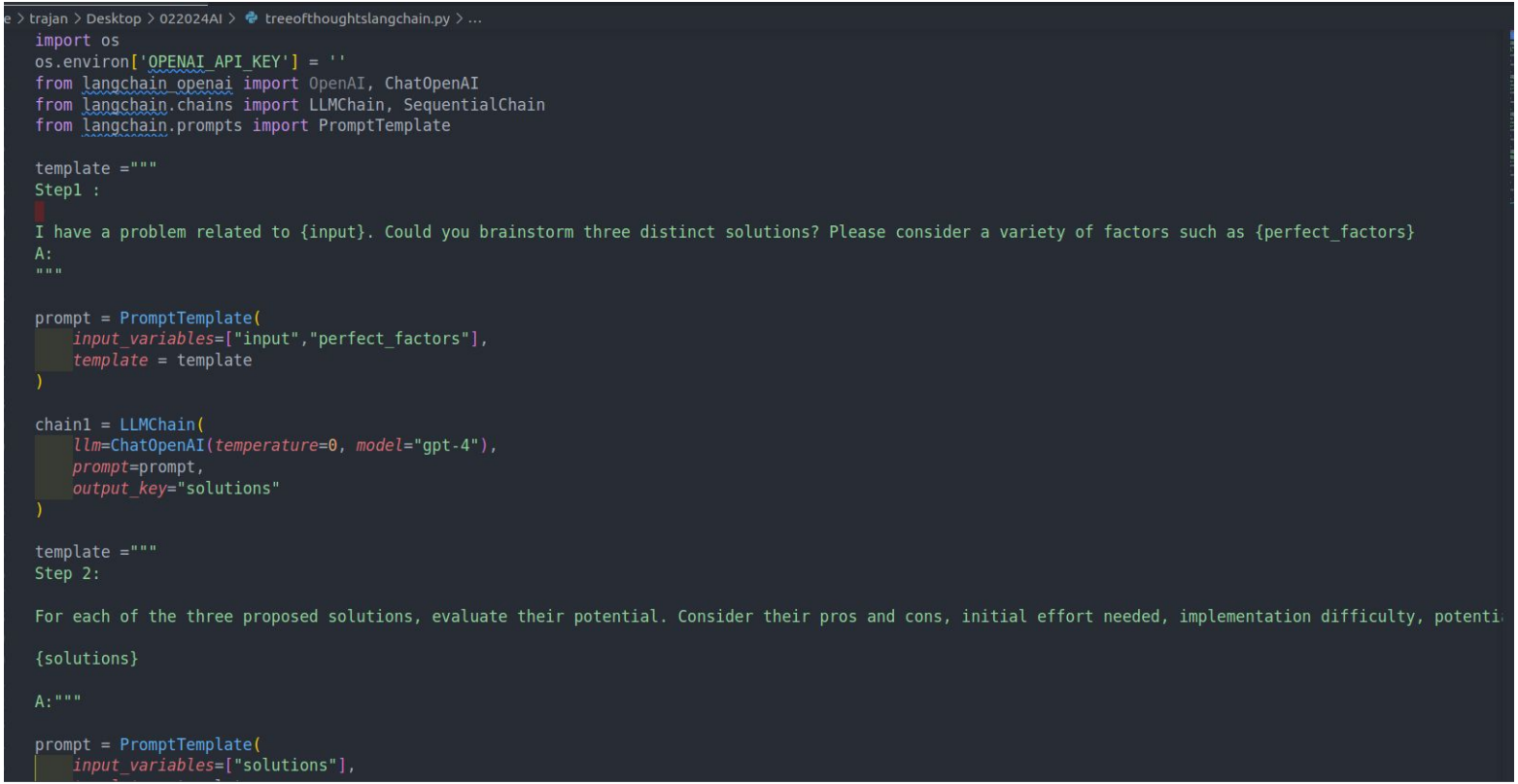
# Replicate27.py “Not LangChain in this example”

```python
import os
os.environ["REPLICATE_API_TOKEN"] = ""
import replicate
from PIL import Image
from urllib.request import urlretrieve

out = replicate.run(
    "stability-ai/stable-diffusion:27b93a2413e7f36cd83da926f3656280b2931564ff050bf9575f1fdf9bcd7478",
    input={"prompt": "Miami Beach"}
    )

urlretrieve(out[0], "/tmp/out.png")
background = Image.open("/tmp/out.png")
```

# treeofthoughtslangchain.py



```
e > trajan > Desktop > 022024AI > treeofthoughtslangchain.py > ...
import os
os.environ['OPENAI_API_KEY'] = ''
from langchain_openai import OpenAI, ChatOpenAI
from langchain.chains import LLMChain, SequentialChain
from langchain.prompts import PromptTemplate

template ="""
Step1 :

I have a problem related to {input}. Could you brainstorm three distinct solutions? Please consider a variety of factors such as {perfect_factors}
A:
"""

prompt = PromptTemplate(
    input_variables=["input","perfect_factors"],
    template = template
)

chain1 = LLMChain(
    llm=ChatOpenAI(temperature=0, model="gpt-4"),
    prompt=prompt,
    output_key="solutions"
)

template ="""
Step 2:

For each of the three proposed solutions, evaluate their potential. Consider their pros and cons, initial effort needed, implementation difficulty, potenti

{solutions}

A:"""

prompt = PromptTemplate(
    input_variables=["solutions"],
```

# Threeofthoughtslangchain.py - continued

```
> Entering new SequentialChain chain...

> Finished chain.
{'input': 'human colonization of Mars', 'perfect_factors': 'The distance between Earth and Mars is very large, making regular resupply difficult', 'ranked_so
lutions': 'Building a Dome Colony:\nPros: This is a more feasible solution with our current technology and knowledge. It would provide a controlled environme
nt for humans to live in.\nCons: It would still require significant resources and effort to build and maintain. It would also not be self-sustaining and woul
d rely on supplies from Earth.\nInitial Effort: High. This would require significant research and development, as well as the construction of the dome and li
fe-support systems.\nImplementation Difficulty: High. This would involve overcoming technical challenges and ensuring the safety and well-being of the coloni
sts.\nPotential Challenges: Developing the necessary technology, securing the necessary funding and resources, and ensuring the colony can be maintained and
supplied.\nExpected Outcomes: If successful, this could result in a functioning colony on Mars, but it would rely on ongoing support from Earth.\nProbability
 of Success: 50%\nConfidence Level: Medium\n\nB: \n\nSending Robots to Prepare Mars:\nPros: This would reduce the risk to human life and could be done with c
urrent technology. Robots could prepare the environment and build infrastructure before humans arrive.\nCons: It would still require significant resources an
d effort. The robots would also need to be able to operate autonomously and handle unexpected situations.\nInitial Effort: Medium. This would require researc
h and development, as well as the construction and launch of the robots.\nImplementation Difficulty: Medium. This would involve overcoming technical challeng
es and ensuring the robots can operate effectively on Mars.\nPotential Challenges: Developing the necessary technology, securing the necessary funding and re
sources, and dealing with any issues or failures that arise.\nExpected Outcomes: If successful, this could result in a prepared environment and infrastructur
e on Mars, reducing the risk and effort required for human colonists.\nProbability of Success: 70%\nConfidence Level: High\n\nRanking:\n1. Sending Robots to
Prepare Mars\n2. Building a Dome Colony\n3. Terraforming Mars\n\nJustification:\nSending robots to prepare Mars is the most promising solution due to its fea
sibility with current technology, reduced risk to human life, and potential to prepare the environment and infrastructure for human colonists. Building a dom
e colony is the second most promising solution, as it is also feasible with current technology and would provide a controlled environment for humans. However
, it would require significant resources and effort, and would not be self-sustaining. Terraforming Mars is the least promising solution due to the enormous
technical challenges, resource requirements, and time scale involved.'}
trajan@docker-ptf:~$
```

# pdfconvolangchain.py

```python
import os
from langchain_community.chat_models import ChatOpenAI
from langchain.document_loaders import PyMuPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import Chroma
from langchain.embeddings import OpenAIEmbeddings
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA

os.environ["OPENAI_API_KEY"] = ""

persist_directory = "/home/trajan/Desktop/022024AI/store"
pdf_path = "/home/trajan/Desktop/022024AI/divinecomedy.pdf"

loader = PyMuPDFLoader(pdf_path)
documents = loader.load()

text_splitter = RecursiveCharacterTextSplitter(chunk_size=512, chunk_overlap=10)
texts = text_splitter.split_documents(documents)

embeddings = OpenAIEmbeddings()
vectordb = Chroma.from_documents(documents=texts,
                                 embedding=embeddings,
                                 persist_directory=persist_directory)
vectordb.persist()

retriever = vectordb.as_retriever(search_kwargs={"k": 3})
llm = ChatOpenAI(model_name='gpt-4')

qa = RetrievalQA.from_chain_type(llm=llm, chain_type="stuff", retriever=retriever)

while True:
        user_input = input("Enter a query: ")
        if user_input == "exit":
            break

        query = f"###Prompt {user_input}"
        try:
            llm_response = qa(query)
```

```
This document appears to be a piece of classical literature, likely a poem or an excerpt from an epic. The language and style suggest it could be from works
such as Dante's "Divine Comedy". The narrator is being guided by a master through a mysterious place, possibly symbolic of a spiritual or intellectual journe
y. The lines contain various themes including courage, intellect, sorrow, and secrets. The last part seems to indicate a conversation with a spirit, which fu
rther suggests a context of the afterlife or spiritual realm. The phrase "vested was with the great mantle" could potentially refer to a person of importance
 or with a significant role.
Enter a query: what can you tell me about this document?
```

# fewshot.py

```python
prompt = FewShotPromptTemplate(
    examples=examples,
    example_prompt=example_prompt,
    suffix="Question: {input}",
    input_variables=["input"]
)
print((prompt | model).invoke({"input": "This is an excellent book with high quality explanations."}))

selector = SemanticSimilarityExampleSelector.from_examples(
    examples=examples,
    embeddings=OpenAIEmbeddings(),
    vectorstore_cls=Chroma,
    k=4,
)
prompt = FewShotPromptTemplate(
    example_selector=selector,
    example_prompt=example_prompt,
    suffix="Question: {input}",
    input_variables=["input"]
)
print((prompt | model).invoke({"input": "What is floor division?"}))
```

```
Number of requested results 4 is greater than number of elements in index 3, updating n_results = 3
content='Floor division is a mathematical operation that returns the largest integer that is less than or equal to the division of two numbers. In Python, th
e floor division operator is denoted by "//".'
```

# chain_of_thought.py

```python
from langchain.chat_models import ChatOpenAI
from langchain.prompts import PromptTemplate


cot_instruction = "Let's think step by step!"
cot_instruction2 = "Explain your reasoning step-by-step. Finally, state the answer."
reasoning_prompt = "{question}\n" + cot_instruction
prompt = PromptTemplate(
    template=reasoning_prompt,
    input_variables=["question"]
)

model = ChatOpenAI()
chain = prompt | model
print(chain.invoke({
    "question": "I had two Flipper Zeros. I lost 1. Then my friend gave me 2. How many Flipper Zeros do I have now?",
}))


if __name__ == "__main__":
    pass
```

```
content='Step 1: You had two Flipper Zeros.\nStep 2: You lost one Flipper Zero.\nStep 3: Your friend gave you two Flipper Zeros.\nStep 4: To determine the to
tal number of Flipper Zeros you have now, we add the remaining Flipper Zero from Step 2 (1) to the two given by your friend in Step 3 (2).\nStep 5: 1 + 2 = 3
\nTherefore, you now have three Flipper Zeros.'
```

# tree_of_thought2.py

```
evalutation_chain = LLMChain(
    llm=ChatOpenAI(),
    prompt=evaluation_prompt,
    output_key="evaluations"
)
reasoning_chain = LLMChain(
    llm=ChatOpenAI(),
    prompt=reasoning_prompt,
    output_key="enhanced_reasoning"
)
ranking_chain = LLMChain(
    llm=ChatOpenAI(),
    prompt=ranking_prompt,
    output_key="ranked_solutions"
)
tot_chain = SequentialChain(
    chains=[solutions_chain, evalutation_chain, reasoning_chain, ranking_chain],
    input_variables=["problem", "factors", "num_solutions"],
    output_variables=["ranked_solutions"]
)
print(tot_chain.run(
    problem="Time travel",
    factors="Give 3 most possible current ways to achieve it and explain why",
    num_solutions=3
))
```

```
1. Implementing strategies for time travel research: This solution is ranked first because it involves a multi-disciplinary approach and collaboration across
 different fields, which is essential for making progress in understanding time travel.
2. Partnerships: Establishing partnerships with academic institutions, research centers, and government agencies is ranked second because collaborating with
experts in relevant fields would provide valuable insights and resources for time travel research.
3. Scenarios: Creating controlled environments with intense gravitational forces and studying quantum entanglement are ranked third because they offer potent
ial avenues for exploring time travel, but they may be more challenging to implement and require significant advancements in technology and theoretical under
standing.
4. Potential Obstacles: This solution is ranked fourth because it highlights the challenges and limitations currently associated with time travel research. O
vercoming these obstacles is necessary for progress, but it is a more long-term aspect of the overall endeavor.
5. Feasibility and probability of achieving practical time travel: This is ranked last because it acknowledges the current low likelihood of achieving practi
cal time travel in the near future. However, it also emphasizes the importance of continued scientific research and exploration for potential breakthroughs i
n the future.
```

# self_consistency.py

```
    For each answer in {solutions}, count the number of times it occurs. Finally, choose the answer that occurs most.

    Most frequent solution:
    """
    consistency_prompt = PromptTemplate(
        template=consistency_template,
        input_variables=["solutions"]
    )
    consistency_chain = LLMChain(
        llm=ChatOpenAI(),
        prompt=consistency_prompt,
        output_key="best_solution"
    )
    answer_chain = SequentialChain(
        chains=[solutions_chain, consistency_chain],
        input_variables=["question", "num_solutions"],
        output_variables=["best_solution"]
    )

    print(answer_chain.run(
        question="What is the holographic principle?",
        num_solutions="5"
    ))


    if __name__ == "__main__":
        pass
```

```
The answer that occurs most frequently is the holographic principle is a concept in theoretical physics that suggests that the information contained within a
 three-dimensional volume can be represented on a two-dimensional surface.
```

# Wikipedia & DuckDuckGo jupyter notebooks

```
from langchain.tools import WikipediaQueryRun
from langchain_community.utilities import WikipediaAPIWrapper
wikipedia = WikipediaQueryRun(api_wrapper=WikipediaAPIWrapper())
wikipedia.run("Hackmiami")
```

```
"Page: HackMiami\nSummary: HackMiami is a formal organization of information security professionals who host the a
nnual international hacker conference that takes place in Miami Beach, FL known as the 'HackMiami Conference.'The
organization has been involved in research primarily in the fields of malware analysis, botnets, web application s
ecurity research, cybercrime research, and network security research.\nThe organization also organizes bi-weekly l
ocal events themed around ethical hacking topics throughout South Florida, and offer information security consulta
tion services to both individuals and businesses.\n\n\n\nPage: Alexander Heid\nSummary: Alexander Heid is an Ameri
can computer security consultant, white hat hacker, and business executive.\nHeid is a co-founder of the South Flo
rida hacker conference and hacker group known as HackMiami, and currently serves as the chief research officer of
the New York City information security firm SecurityScorecard.\n\nPage: Computer security conference\nSummary: A c
omputer security conference is a convention for individuals involved in computer security. They generally serve as
meeting places for system and network administrators, hackers, and computer security experts.\n\n"
```

gs @ info@hackmiami.com...'

[1]:
```
from langchain.tools import DuckDuckGoSearchRun
search = DuckDuckGoSearchRun()
search.run("What is Hackmiami")
```

```
/home/trajan/.local/lib/python3.10/site-packages/langchain_community/utilities/duckduckgo_search.py:47: UserWarnin
g: DDGS running in an async loop. This may cause errors. Use AsyncDDGS instead.
  with DDGS() as ddgs:
```

[1]:
```
'HackMiami is a dynamic and innovative community of cybersecurity enthusiasts and experts, dedicated to exploring
the cutting edge of information security. --== HackMiami XI ==--Training Sessions: May 15 - 17, 2024 Conference Da
te: May 18, 2024 CFP Open Date: August 1, 2023 Pre-sale Starts: September 1, 2023 Location: Marenas Beach Resort @
Sunny Isles Beach Address: 18683 Collins Avenue, Sunny Isles Beach, FL 33160 %27. Follow @HackMiami on Twitter for
latest updates. HackMiami 2023 HackMiami X 2023 Conference will consist of training classes on May 19, 2023 and sp
eaking tracks on Saturday May 20, 2023. Training Sessions - We are accepting proposals for paid training courses t
aking place on May 19, 2023. Cyber Security Hackmiami X By HackMiami Overall Rating In-Person Sunny Isles Beach, F
L, United States May 19, 2023 - May 20, 2023 Event Link https://hackmiami.com/ About Event HackMiami X 2023 Confer
ence will consist of training classes on May 19, 2023 and speaking tracks on Saturday May 20, 2023. Hackmiami XI C
onference early bird is now OPEN, including general admission and training. Also CFP is open for talks and trainin
gs @ info@hackmiami.com...'
```

# Langchain and cybersecurity

As you have seen in this presentation, langchain framework is an excellent way of extending and interacting with LLMs, this technology is just starting and will be advancing as the technology of LLMs expands and develops. Here is an example of an use of Langchain with a Red Team Tool called Cobalt Strike presented at Hackmiami 2023. Take time to watch it and think about the possibilities of this framework in combination with LLMs and other new technologies down the road that are still based on Generative AI.  I hope you enjoyed this presentation.

https://www.youtube.com/watch?v=AcG4jDVO-UI

# References


- https://python.langchain.com/docs/additional_resources/tutorials
- https://huggingface.co/blog/Andyrasika/agent-helper-langchain-hf
- https://python.langchain.com/docs/modules/data_connection/document_loaders/pdf
- https://www.amazon.com/Generative-AI-LangChain-language-ChatGPT/dp/1835083463
- https://python.langchain.com/docs/integrations/llms/replicate
- https://python.langchain.com/docs/integrations/tools/tavily_search

# Thank you

Any questions or concerns

Rod Soto

@rodsoto

Rodsoto.net

https://github.com/rsfl/