LangChain

deeplake

```
import os
```

```
os.environ['OPENAI_API_KEY']="sk-"
os.environ['ACTIVELOOP_TOKEN']="..-"
```

```
!pip install langchain==0.0.208 deeplake openai==0.27.8 tiktoken
```

```
Collecting langchain==0.0.208
  Downloading langchain-0.0.208-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 16.5 MB/s eta 0:00:00
Collecting deeplake
  Downloading deeplake-3.8.12.tar.gz (583 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 583.4/583.4 kB 47.5 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting openai==0.27.8
  Downloading openai-0.27.8-py3-none-any.whl (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.6/73.6 kB 9.9 MB/s eta 0:00:00
Collecting tiktoken
  Downloading tiktoken-0.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 60.2 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=5.4.1 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (2.0.23)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (3.9.1)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (
Collecting dataclasses-json<0.6.0,>=0.5.7 (from langchain==0.0.208)
  Downloading dataclasses_json-0.5.14-py3-none-any.whl (26 kB)
Collecting langchainplus-sdk>=0.0.13 (from langchain==0.0.208)
  Downloading langchainplus_sdk-0.0.20-py3-none-any.whl (25 kB)
Requirement already satisfied: numexpr<3.0.0,>=2.8.4 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (2.8.8)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (1.23.5)
Collecting openapi-schema-pydantic<2.0,>=1.2 (from langchain==0.0.208)
  Downloading openapi_schema_pydantic-1.2.4-py3-none-any.whl (90 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 90.0/90.0 kB 12.3 MB/s eta 0:00:00
Requirement already satisfied: pydantic<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (1.10.13)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.208) (8.2.3
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai==0.27.8) (4.66.1)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from deeplake) (9.4.0)
Collecting boto3 (from deeplake)
  Downloading boto3-1.34.4-py3-none-any.whl (139 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.3/139.3 kB 18.7 MB/s eta 0:00:00
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from deeplake) (8.1.7)
Collecting pathos (from deeplake)
  Downloading pathos-0.3.1-py3-none-any.whl (82 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.1/82.1 kB 11.7 MB/s eta 0:00:00
Collecting humbug>=0.3.1 (from deeplake)
  Downloading humbug-0.3.2-py3-none-any.whl (15 kB)
Collecting lz4 (from deeplake)
  Downloading lz4-4.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
```

```
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 71.3 MB/s eta 0:00:00
     Requirement already satisfied: pyjwt in /usr/lib/python3/dist-packages (from deeplake) (2.3.0)
     Collecting libdeeplake==0.0.92 (from deeplake)
       Downloading libdeeplake-0.0.92-cp310-cp310-manylinux2014_x86_64.whl (14.7 MB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.7/14.7 MB 63.6 MB/s eta 0:00:00
     Collecting aioboto3>=10.4.0 (from deeplake)
       Downloading aioboto3-12.1.0-py3-none-any.whl (32 kB)
     Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-packages (from deeplake) (1.5.8)
     Collecting dill (from libdeeplake==0.0.92->deeplake)
       Downloading dill-0.3.7-py3-none-any.whl (115 kB)
          ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 16.9 MB/s eta 0:00:00
     Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2023.6.3)
```

```python
from langchain.llms import OpenAI
```

```python
llm = OpenAI(model="text-davinci-003",temperature=0.9)
```

```python
text = "Suggest a personalized workout routine for someone looking to improve cardiovascular endurance and prefers outdoor activities."
print(llm(text))
```

```


     1. Jogging/Running - 3-4 times a week for 20-30 minutes.

     2. Hill/Trail Running - 2-3 times a week for 20-30 minutes.

     3. Cycling - 2-3 times a week for 25-30 minutes.

     4. Swimming - 2-3 times a week for 20-30 minutes.

     5. Rowing - 2-3 times a week for 15-20 minutes.

     6. Hiking - 1-2 times a week for 30-60 minutes.

     7. Interval Training - 1-2 times a week for 20-30 minutes.

     8. Yoga/Stretching - 2-3 times a week for 10-15 minutes.
```

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

llm = OpenAI(model="text-davinci-003",temperature=0.9)

prompt = PromptTemplate(
    input_variables=["product"],
    template="What is a good name for a company that makes {product}?",
)

chain = LLMChain(
    llm=llm,
    prompt=prompt
)
```

```python
print(chain.run("eco-friendly water bottles"))
```

```


     EcoLife Water Bottles.
```

```python
from langchain.llms import OpenAI
from langchain.chains import ConversationChain
from langchain.memory import ConversationBufferMemory

llm = OpenAI(
    model="text-davinci-003",
    temperature=0
)
```

```
conversation= ConversationChain(
    llm=llm,
    verbose=True,
    memory=ConversationBufferMemory()
)

conversation.predict(input="Tell me about yourself")

conversation.predict(input="What can you do?")
conversation.predict(input="How can you help me with data analysis?")

print(conversation)
```

```
> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from t

Current conversation:

Human: Tell me about yourself
AI:

> Finished chain.


> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from t

Current conversation:
Human: Tell me about yourself
AI:  Hi there! My name is AI and I'm a virtual assistant. I'm here to help you with any questions you may have. I'm powered by arti
Human: What can you do?
AI:

> Finished chain.


> Entering new  chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is talkative and provides lots of specific details from t

Current conversation:
Human: Tell me about yourself
AI:  Hi there! My name is AI and I'm a virtual assistant. I'm here to help you with any questions you may have. I'm powered by arti
Human: What can you do?
AI:  I can help you with a variety of tasks. I can provide you with information from my context, such as the current weather, news,
Human: How can you help me with data analysis?
AI:

> Finished chain.
memory=ConversationBufferMemory(chat_memory=ChatMessageHistory(messages=[HumanMessage(content='Tell me about yourself', additional_k
```

```
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import DeepLake
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.llms import OpenAI
from langchain.chains import RetrievalQA

llm = OpenAI(model="text-davinci-003",temperature=0)

embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")

texts = [
    "Napoleon Bonaparte was born in 15 August 1769",
    "Louis XIV was born in 5 September 1638"
]

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=0
)

docs = text_splitter.create_documents(texts)

org_id = "..."
datasetname="..."

dataset_path = f"hub://{org_id}/{datasetname}"

db = DeepLake(
    dataset_path=dataset_path,
    embedding_function=embeddings
)

db.add_documents(docs)
```

```
Your Deep Lake dataset has been successfully created!
Creating 2 embeddings in 1 batches of size 2:: 100%|██████████| 1/1 [00:03<00:00,  3.63s/it]Dataset(path='hub://anananantha28/Langcl

  tensor      htype      shape     dtype  compression
  -------    -------    -------   -------  -------
   text       text      (2, 1)      str     None
 metadata     json      (2, 1)      str     None
 embedding  embedding  (2, 1536)  float32   None
    id        text      (2, 1)      str     None

['e6cee848-9f66-11ee-8ce2-0242ac1c000c',
 'e6ceea6e-9f66-11ee-8ce2-0242ac1c000c']
```

```
retrieval_qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=db.as_retriever()
)
```

```
from langchain.agents import initialize_agent,Tool
from langchain.agents import AgentType

tools = [
    Tool(
        name="Retrieval QA System",
        func=retrieval_qa.run,
        description="Useful for answering questions."
    ),
]

agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

```
response = agent.run("When was Napoleone born?")
print(response)
```

```
> Entering new  chain...
 I need to find out when Napoleone was born.
Action: Retrieval QA System
Action Input: When was Napoleone born?
Observation:  Napoleon Bonaparte was born on 15 August 1769.
Thought: I now know the final answer.
Final Answer: Napoleon Bonaparte was born on 15 August 1769.

> Finished chain.
Napoleon Bonaparte was born on 15 August 1769.
```

```
db = DeepLake(
    dataset_path=dataset_path,
    embedding_function=embeddings
)

texts = [
    "Lady Gaga was born in 28 March 1986",
    "Michael Jeffrey Jordan was born in 17 February 1963"
]

text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
docs = text_splitter.create_documents(texts)

db.add_documents(docs)
```

```
Deep Lake Dataset in hub://anananantha28/Langchain_anantha already exists, loading from the storage
Creating 2 embeddings in 1 batches of size 2:: 100%|██████████| 1/1 [00:02<00:00,  2.62s/it]Dataset(path='hub://anananantha28/Langcl

  tensor      htype      shape     dtype  compression
  -------    -------    -------   -------  -------
 embedding  embedding  (4, 1536)  float32   None
    id        text      (4, 1)      str     None
 metadata     json      (4, 1)      str     None
```

```
    text        text       (4, 1)       str      None

['206d82de-9f68-11ee-8ce2-0242ac1c000c',
 '206d8478-9f68-11ee-8ce2-0242ac1c000c']
```

```
llm = OpenAI(model="text-davinci-003", temperature=0)

retrieval_qa = RetrievalQA.from_chain_type(
    llm=llm,
  chain_type="stuff",
  retriever=db.as_retriever()
)

tools = [
    Tool(
        name="Retrieval QA System",
        func=retrieval_qa.run,
        description="Useful for answering questions."
    ),
]

agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

```
response = agent.run("When was Michael Jordan born?")
print(response)
```

```
> Entering new  chain...
 I need to find out when Michael Jordan was born.
Action: Retrieval QA System
Action Input: When was Michael Jordan born?
Observation:  Michael Jordan was born on 17 February 1963.
Thought: I now know the final answer.
Final Answer: Michael Jordan was born on 17 February 1963.

> Finished chain.
Michael Jordan was born on 17 February 1963.
```

```
from langchain.llms import OpenAI

from langchain.agents import AgentType
from langchain.agents import load_tools
from langchain.agents import initialize_agent

from langchain.agents import Tool
from langchain.utilities import GoogleSearchAPIWrapper
```

```
llm = OpenAI(model="text-davinci-003", temperature=0)
```

```
!pip install -U duckduckgo-search
```

```
Requirement already satisfied: duckduckgo-search in /usr/local/lib/python3.10/dist-packages (4.1.0)
Requirement already satisfied: click>=8.1.7 in /usr/local/lib/python3.10/dist-packages (from duckduckgo-search) (8.1.7)
Requirement already satisfied: lxml>=4.9.3 in /usr/local/lib/python3.10/dist-packages (from duckduckgo-search) (4.9.3)
Requirement already satisfied: curl-cffi>=0.5.10 in /usr/local/lib/python3.10/dist-packages (from duckduckgo-search) (0.5.10)
Requirement already satisfied: cffi>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from curl-cffi>=0.5.10->duckduckgo-search)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12.0->curl-cffi>=0.5.10->duckduckg
```

```
from langchain.tools import DuckDuckGoSearchRun
```

```
search = DuckDuckGoSearchRun()
```

```
tools = [
    Tool(
        name = "google-search",
        func = search.run,
        description = "useful for when you need to search google to answer questions about current events"
    )
]
```

```
agent = initialize_agent(tools,
                         llm,
                         agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
                         verbose=True,
                         max_iterations=6)
```

```
response = agent("What's the latest news about the Mars rover?")
print(response['output'])
```

```
> Entering new  chain...
 I need to find out the latest news about the Mars rover
Action: google-search
Action Input: "latest news Mars rover"
Observation: CNN — After spending 1,000 days on the Martian surface, NASA's Perseverance rover has uncovered new details about the l
Thought: I now know the final answer
Final Answer: NASA's Perseverance rover has recently completed its exploration of the ancient river delta that holds evidence of a l

> Finished chain.
NASA's Perseverance rover has recently completed its exploration of the ancient river delta that holds evidence of a lake that fille
```

```
from langchain.llms import OpenAIChat
from langchain.agents import Tool
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.agents import initialize_agent, AgentType
```

```
from langchain.tools import DuckDuckGoSearchRun
```

```
llm = OpenAI(model="text-davinci-003", temperature=0)

prompt = PromptTemplate(
    input_variables=["query"],
    template="Write a summary of the following text: {query}"
)

summarize_chain = LLMChain(llm=llm, prompt=prompt)
```

```
search = DuckDuckGoSearchRun()

tools = [
    Tool(
        name="Search",
        func=search.run,
        description="useful for finding information about recent events"
    ),

    Tool(
        name="Summarizer",
        func=summarize_chain.run,
        description='useful for summarizing texts'
    )
]
```

```
agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True
)
```

```
response = agent("What's the latest news about the Mars rover? Then please summarize the results.")
print(response['output'])
```

```
> Entering new  chain...
 I need to find the latest news about the Mars rover and then summarize it.
Action: Search
Action Input: Latest news about the Mars rover
Observation: CNN.com View 3 comments Sponsored Content KARD Monroe Story by Lauren Sforza • 3d After 1,000 days on the Martian surfa
Thought: I now have the latest news about the Mars rover.
Action: Summarizer
Action Input: CNN.com View 3 comments Sponsored Content KARD Monroe Story by Lauren Sforza • 3d After 1,000 days on the Martian surf

Observation:
```

```
The Perseverance rover has been on the Martian surface for 1,000 days and has collected samples that reveal the history of the plane
Thought:WARNING:langchain.llms.openai:Retrying langchain.llms.openai.completion_with_retry.<locals>._completion_with_retry in 4.0 se
WARNING:langchain.llms.openai:Retrying langchain.llms.openai.completion_with_retry.<locals>._completion_with_retry in 4.0 seconds as
WARNING:langchain.llms.openai:Retrying langchain.llms.openai.completion_with_retry.<locals>._completion_with_retry in 8.0 seconds as
 I now know the final answer.
Final Answer: The Perseverance rover has been on the Martian surface for 1,000 days and has collected samples that reveal the histor

> Finished chain.
The Perseverance rover has been on the Martian surface for 1,000 days and has collected samples that reveal the history of the plane
```