D:₩ai₩langchain₩langchaintest.py

```python
# pip install tiktoken langchain openai sentence_transformers pytube youtube-transcript-api faiss-cpu wikipedia
import os
import langchain

os.environ["OPENAI_API_KEY"] = "sk-r8zDrchK02d7G57tNp5dT******MGwTDKGW4uQdFv"
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "hf_mkvaVivwXWbs******HRYhsguBJxJbAC"

'''
from langchain.embeddings import OpenAIEmbeddings
embeddings = OpenAIEmbeddings()
'''
# open-source text embedding model hosted on Hugging Face
# text embedding test

from langchain.embeddings import HuggingFaceEmbeddings
embeddings = HuggingFaceEmbeddings(model_name = "sentence-transformers/all-MiniLM-L6-v2")
text = "Alice has a parrot. What animal is Alice's pet?"
text_embedding = embeddings.embed_query(text)
#print(text_embedding)


from langchain import HuggingFaceHub
'''
llm = HuggingFaceHub(repo_id = "google/flan-t5-xxl", model_kwargs={"temperature":0.1, "max_length":512})
'''

from langchain.llms import OpenAI
llm = OpenAI(model_name="text-davinci-003")

# The LLM takes a prompt as an input and outputs a completion
prompt = "Alice has a parrot. What animal is Alice's pet?"
completion = llm(prompt)
print(completion)

#============================================================
from langchain import PromptTemplate, FewShotPromptTemplate

examples = [
    {"word": "happy", "antonym": "sad"},
    {"word": "tall", "antonym": "short"},
]

example_template = """
Word: {word}
Antonym: {antonym}\n
"""

example_prompt = PromptTemplate(
    input_variables=["word", "antonym"],
    template=example_template,
)

few_shot_prompt = FewShotPromptTemplate(
    examples=examples,
    example_prompt=example_prompt,
```

```
    prefix="Give the antonym of every input",
    suffix="Word: {input}\nAntonym:",
    input_variables=["input"],
    example_separator="\n",
)

print(few_shot_prompt.format(input="big"))
#=========================================================
from langchain.document_loaders import YoutubeLoader
from langchain.vectorstores import FAISS
from langchain.chains import RetrievalQA

loader = YoutubeLoader.from_youtube_url("https://www.youtube.com/watch?v=dQw4w9WgXcQ")

documents = loader.load()

# create the vectorestore to use as the index
db = FAISS.from_documents(documents, embeddings)

retriever = db.as_retriever()

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

query = "What am I never going to do?"
result = qa({"query": query})

print(result['result'])

#=========================================================
from langchain import ConversationChain

conversation = ConversationChain(llm=llm, verbose=True)

conversation.predict(input="Alice has a parrot.")

conversation.predict(input="Bob has two cats.")

print(conversation.predict(input="Who has the parrot?"))

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

tools = load_tools(["wikipedia", "llm-math"], llm=llm)
agent = initialize_agent(tools,
                         llm,
                         agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
                         verbose=True)


agent.run("When was Barack Obama born? How old was he in 2022?")
```