

react-hierarchy

November 28, 2023

```
[ ]: import os

from dotenv import load_dotenv
from langchain import hub
from langchain.agents import AgentExecutor, AgentType, initialize_agent, \
    load_tools
from langchain.agents.format_scratchpad import format_log_to_str
from langchain.agents.output_parsers import (
    JSONAgentOutputParser,
    ReActSingleInputOutputParser,
)
from langchain.chains.conversation.memory import ConversationBufferWindowMemory
from langchain.chat_models import ChatOpenAI
from langchain.llms import OpenAI
from langchain.tools import ArxivQueryRun, WikipediaQueryRun, tool
from langchain.tools.render import render_text_description_and_args, \
    format_tool_to_openai_function
from langchain.utilities import ArxivAPIWrapper, WikipediaAPIWrapper
from langchain.prompts import MessagesPlaceholder
from langchain.schema import ChatMessage, SystemMessage

from llamp.mp.agents import (
    MPSummaryExpert,
    MPThermoExpert,
    MPElasticityExpert,
    MPDielectricExpert,
)

load_dotenv()

OPENAI_API_KEY = os.getenv("OPENAI_API_KEY", None)

# OPENAI_GPT_MODEL = "gpt-4-1106-preview"
OPENAI_GPT_MODEL = "gpt-3.5-turbo-1106"
```

```
[ ]: import re
```

```

mp_llm = ChatOpenAI(
    temperature=0,
    model=OPENAI_GPT_MODEL,
    openai_api_key=OPENAI_API_KEY,
)

llm = ChatOpenAI(
    temperature=0,
    model=OPENAI_GPT_MODEL,
    openai_api_key=OPENAI_API_KEY
)

wikipedia = WikipediaQueryRun(api_wrapper=WikipediaAPIWrapper())
arxiv = ArxivQueryRun(api_wrapper=ArxivAPIWrapper())

tools = [
    MPThermoExpert(llm=mp_llm).
    ↪as_tool(agent_kwargs=dict(return_intermediate_steps=True)),
    MPElasticityExpert(llm=mp_llm).
    ↪as_tool(agent_kwargs=dict(return_intermediate_steps=True)),
    MPSummaryExpert(llm=mp_llm).
    ↪as_tool(agent_kwargs=dict(return_intermediate_steps=True)),
    # arxiv,
    # wikipedia,
]

prompt = hub.pull("hwchase17/react-multi-input-json")
prompt.messages[0].prompt.template = re.sub(
    r"\s+", " ",
    """You are a helpful data-aware agent that can consult materials-related
data through Materials Project (MP) database, arXiv, and Wikipedia. Ask
user to clarify their queries if needed. Please note that you don't have
direct control to MP but through multiple assistant agents to help you.
You need to provide complete context in the input for them to do their job.
""").replace("\n", " ") + prompt.messages[0].prompt.template

prompt = prompt.partial(
    tools=render_text_description_and_args(tools),
    tool_names=", ".join([t.name for t in tools]),
)

agent = (
    {
        "input": lambda x: x["input"],
        "agent_scratchpad": lambda x:
    ↪format_log_to_str(x["intermediate_steps"]),
    }

```

```

    | prompt
    | llm.bind(stop=["Observation"])
    # / map_reduce_chain # TODO: Add map-reduce after LLM
    | JSONAgentOutputParser()
)

conversational_memory = ConversationBufferWindowMemory(
    memory_key='chat_history',
    k=5,
    return_messages=True
)

agent_kwargs = {
    "extra_prompt_messages": [
        MessagesPlaceholder(variable_name="chat_history"),
        # SystemMessage(content=re.sub(
        #     r"\s+", " ",
        #     """You are a helpful data-aware agent that can consult
        ↪materials-related
        #     data through Materials Project (MP) database, arXiv, and
        ↪Wikipedia. Ask
        #     user to clarify their queries if needed. Please note that you
        ↪don't have
        #     direct control to MP but through multiple assistant agents to
        ↪help you.
        #     You need to provide complete context for them to do their job.
        #     """).replace("\n", " ")
        # )
    ],
    # "early_stopping_method": 'generate',
    # "extra_prompt_messages":
    # )
}

agent_executor = initialize_agent(
    agent=AgentType.STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    tools=tools,
    llm=llm,
    verbose=True,
    max_iterations=5,
    memory=conversational_memory,
    # agent_kwargs=agent_kwargs,
    handle_parsing_errors=True,
)

# agent_executor = initialize_agent(

```

```
# tools=tools,
# llm=llm,
# agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
# verbose=True,
# max_iterations=5,
# )
```

```
/home/cyrus/miniconda3/envs/llamp/lib/python3.11/site-
packages/mp_api/client/mprester.py:230: UserWarning: mpcontribs-client not
installed. Install the package to query MPContribs data, or construct pourbaix
diagrams: 'pip install mpcontribs-client'
warnings.warn(
```

```
[ ]: prompt
```

```
[ ]: ChatPromptTemplate(input_variables=['agent_scratchpad', 'input'],
partial_variables={'tools': "MPThermoExpert: MPThermoExpert(input: str) -
Thermodynamics expert that has access to Materials Project thermo endpoint,
args: {'input': {'title': 'Input', 'description': 'Complete question to ask the
assistatn agent. Should include all the context and details needed to answer the
question holistically.', 'type': 'string'}}\nMPElasticityExpert:
MPElasticityExpert(input: str) - Elasticity expert that has access to Materials
Project elasticity endpoint, including bulk, shear, and young's modulus, poisson
ratio, and universal anisotropy index, args: {'input': {'title': 'Input',
'description': 'Complete question to ask the assistatn agent. Should include all
the context and details needed to answer the question holistically.', 'type':
'string'}}\nMPSummaryExpert: MPSummaryExpert(input: str) - Summary expert that
has access to Materials Project summary endpoint, args: {'input': {'title':
'Input', 'description': 'Complete question to ask the assistatn agent. Should
include all the context and details needed to answer the question
holistically.', 'type': 'string'}}", 'tool_names': 'MPThermoExpert,
MPElasticityExpert, MPSummaryExpert'}, messages=[SystemMessagePromptTemplate(pro
mpt=PromptTemplate(input_variables=['tool_names', 'tools'], template='You are a
helpful data-aware agent that can consult materials-related data through
Materials Project (MP) database, arXiv, and Wikipedia. Ask user to clarify their
queries if needed. Please note that you don\'t have direct control to MP but
through multiple assistant agents to help you. You need to provide complete
context in the input for them to do their job. Respond to the human as helpfully
and accurately as possible. You have access to the following
tools:\n\n{tools}\n\nUse a json blob to specify a tool by providing an action
key (tool name) and an action_input key (tool input).\n\nValid "action" values:
"Final Answer" or {tool_names}\n\nProvide only ONE action per $JSON_BLOB, as
shown:\n\n```\n{\n  "action": $TOOL_NAME,\n  "action_input":
$INPUT\n}\n```\n\nFollow this format:\n\nQuestion: input question to
answer\nThought: consider previous and subsequent
steps\nAction:\n```\n$JSON_BLOB\n```\nObservation: action result\n... (repeat
Thought/Action/Observation N times)\nThought: I know what to
```

```

respond\nAction:\n```\n{\n  "action": "Final Answer",\n  "action_input": "Final response to human"\n}\n\nBegin! Reminder to ALWAYS respond with a valid json blob of a single action. Use tools if necessary. Respond directly if appropriate. Format is Action:```${JSON_BLOB}``then Observation'))), HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['agent_scratchpad', 'input'], template='{input}\n\n{agent_scratchpad}\n (reminder to respond in a JSON blob no matter what)')))]

```

```
[ ]: prompt.messages
```

```
[ ]: [SystemMessagePromptTemplate(prompt=PromptTemplate(input_variables=['tool_names', 'tools'], template='You are a helpful data-aware agent that can consult materials-related data through Materials Project (MP) database, arXiv, and Wikipedia. Ask user to clarify their queries if needed. Please note that you don\'t have direct control to MP but through multiple assistant agents to help you. You need to provide complete context in the input for them to do their job. Respond to the human as helpfully and accurately as possible. You have access to the following tools:\n\n{tools}\n\nUse a json blob to specify a tool by providing an action key (tool name) and an action_input key (tool input).\n\nValid "action" values: "Final Answer" or {tool_names}\n\nProvide only ONE action per ${JSON_BLOB}, as shown:\n\n```\n{\n  "action": $TOOL_NAME,\n  "action_input": $INPUT\n}\n```\n\nFollow this format:\n\nQuestion: input question to answer\nThought: consider previous and subsequent steps\nAction:\n```\n${JSON_BLOB}\n```\nObservation: action result\n... (repeat Thought/Action/Observation N times)\nThought: I know what to respond\nAction:\n```\n{\n  "action": "Final Answer",\n  "action_input": "Final response to human"\n}\n\nBegin! Reminder to ALWAYS respond with a valid json blob of a single action. Use tools if necessary. Respond directly if appropriate. Format is Action:```${JSON_BLOB}``then Observation'))), HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['agent_scratchpad', 'input'], template='{input}\n\n{agent_scratchpad}\n (reminder to respond in a JSON blob no matter what)')))]

```

```
[ ]: agent_executor
```

```
[ ]: AgentExecutor(memory=ConversationBufferWindowMemory(return_messages=True, memory_key='chat_history'), verbose=True, tags=['structured-chat-zero-shot-react-description'], agent=StructuredChatAgent(llm_chain=LLMChain(prompt=ChatPromptTemplate(input_variables=['agent_scratchpad', 'input'], messages=[SystemMessagePromptTemplate(prompt=PromptTemplate(input_variables=[], template='Respond to the human as helpfully and accurately as possible. You have access to the following tools:\n\nMPThermoExpert: MPThermoExpert(input: str) - Thermodynamics expert that has access to Materials Project thermo endpoint, args: {{{{\input\': {{{{\title\': \'Input\', \'description\': \'Complete question to ask the assistatn agent. Should include all the context and details needed to answer the question holistically.\', \'type\': \'string\'}}}}}}}\nMPElasticityExpert: MPElasticityExpert(input: str) -

```

```

Elasticity expert that has access to Materials Project elasticity endpoint,
including bulk, shear, and young's modulus, poisson ratio, and universal
anisotropy index, args: {{{{'input': {{{{'title': 'Input',
'description': 'Complete question to ask the assistatn agent. Should include
all the context and details needed to answer the question holistically.'},
'type': 'string'}}}}}}}\nMPSummaryExpert: MPSummaryExpert(input: str) -
Summary expert that has access to Materials Project summary endpoint, args:
{{{{'input': {{{{'title': 'Input', 'description': 'Complete question to
ask the assistatn agent. Should include all the context and details needed to
answer the question holistically.'}, 'type': 'string'}}}}}}}\n\nUse a json
blob to specify a tool by providing an action key (tool name) and an
action_input key (tool input).\n\nValid "action" values: "Final Answer" or
MPThermoExpert, MPElasticityExpert, MPSummaryExpert\n\nProvide only ONE action
per $JSON_BLOB, as shown:\n\n```\n{"action": $TOOL_NAME,\n "action_input":
$INPUT\n}\n```\n\nFollow this format:\n\nQuestion: input question to
answer\nThought: consider previous and subsequent
steps\nAction:\n```\n$JSON_BLOB\n```\nObservation: action result\n... (repeat
Thought/Action/Observation N times)\nThought: I know what to
respond\nAction:\n```\n{"action": "Final Answer",\n "action_input": "Final
response to human"\n}\n```\n\nBegin! Reminder to ALWAYS respond with a valid
json blob of a single action. Use tools if necessary. Respond directly if
appropriate. Format is Action:```\n$JSON_BLOB```\nthen Observation:.\nThought:')),
HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['agent_scratchp
ad', 'input'], template='{input}\n\n{agent_scratchpad}')))),
llm=ChatOpenAI(client=<class
'openai.api_resources.chat_completion.ChatCompletion'>,
model_name='gpt-3.5-turbo-1106', temperature=0.0, openai_api_key='sk-
XZjGaV4xEz10A2LbUlqvT3BlbkFJhBaXjUgZNjljo0qVjK7R', openai_api_base='',
openai_organization='', openai_proxy='')), output_parser=StructuredChatOutputPar
serWithRetries(output_fixing_parser=OutputFixingParser(parser=StructuredChatOutp
utParser(),
retry_chain=LLMChain(prompt=PromptTemplate(input_variables=['completion',
'error', 'instructions'], template='Instructions:\n-----
\n{instructions}\n-----\nCompletion:\n-----\n{completion}\n---
-----\n\nAbove, the Completion did not satisfy the constraints given in
the Instructions.\nError:\n-----\n{error}\n-----\n\nPlease try
again. Please only respond with an answer that satisfies the constraints laid
out in the Instructions:'), llm=ChatOpenAI(client=<class
'openai.api_resources.chat_completion.ChatCompletion'>,
model_name='gpt-3.5-turbo-1106', temperature=0.0, openai_api_key='sk-
XZjGaV4xEz10A2LbUlqvT3BlbkFJhBaXjUgZNjljo0qVjK7R', openai_api_base='',
openai_organization='', openai_proxy='')))), allowed_tools=['MPThermoExpert',
'MPElasticityExpert', 'MPSummaryExpert']),
tools=[StructuredTool(name='MPThermoExpert', description='MPThermoExpert(input:
str) - Theromodynamics expert that has access to Materials Project thermo
endpoint', args_schema=<class 'llamp.mp.agents.ChainInputSchema'>,
func=<function MPAgent.as_tool.<locals>.run at 0x7f3511443880>),

```

```
StructuredTool(name='MPElasticityExpert', description="MPElasticityExpert(input: str) - Elasticity expert that has access to Materials Project elasticity endpoint, including bulk, shear, and young's modulus, poisson ratio, and universal anisotropy index", args_schema=<class 'llamp.mp.agents.ChainInputSchema'>, func=<function MPAgent.as_tool.<locals>.run at 0x7f3511443240>), StructuredTool(name='MPSummaryExpert', description='MPSummaryExpert(input: str) - Summary expert that has access to Materials Project summary endpoint', args_schema=<class 'llamp.mp.agents.ChainInputSchema'>, func=<function MPAgent.as_tool.<locals>.run at 0x7f3511490360>)], max_iterations=5, handle_parsing_errors=True)
```

```
[ ]: agent_executor.invoke({
    "input": "Please give me the elastic tensor of NaCl"
})
```

```
> Entering new AgentExecutor chain...
I will use the MPElasticityExpert tool to retrieve the elastic
tensor of NaCl.
```

Action:

```
```json
{
 "action": "MPElasticityExpert",
 "action_input": "Elastic tensor of NaCl"
}
```
```

```
> Entering new AgentExecutor chain...
```

```
/home/cyrus/miniconda3/envs/llamp/lib/python3.11/site-
packages/mp_api/client/mprester.py:230: UserWarning: mpcontribs-client not
installed. Install the package to query MPContribs data, or construct pourbaix
diagrams: 'pip install mpcontribs-client'
  warnings.warn(
/home/cyrus/miniconda3/envs/llamp/lib/python3.11/site-
packages/mp_api/client/mprester.py:230: UserWarning: mpcontribs-client not
installed. Install the package to query MPContribs data, or construct pourbaix
diagrams: 'pip install mpcontribs-client'
  warnings.warn(
```

Action:

```
```{  
 "action": "search_materials_elasticity__get",
 "action_input": {
 "formula": "NaCl"
 }
}```{"formula": "NaCl"}
```

Retrieving SummaryDoc documents: 0%| | 0/3 [00:00<?, ?it/s]

Retrieving ElasticityDoc documents: 0%| | 0/2 [00:00<?, ?it/s]



```
[{'formula_pretty': 'NaCl', 'material_id': 'mp-22851',
'elastic_tensor': {'raw': [[75.98221656025066, 0.7598099394417421,
0.7598099394417421, 4.510281037539697e-16, 4.440892098500625e-16,
4.440892098500625e-16], [0.7598099394417421, 75.98221656025066,
0.7598099394417424, 4.440892098500625e-16, 4.510281037539697e-16,
5.229936620407271e-16], [0.7598099394417421, 0.7598099394417424,
75.98221656025068, 4.440892098500626e-16, 4.440892098500626e-16,
4.510281037539696e-16], [4.510281037539697e-16, 4.440892098500625e-16,
4.440892098500626e-16, -2.6118973682358293, 0.0, -7.081558243585513e-32],
[4.440892098500625e-16, 4.510281037539697e-16, 4.440892098500626e-16, 0.0,
-2.6118973682358297, 7.249471490433169e-17], [4.440892098500625e-16,
5.229936620407271e-16, 4.510281037539696e-16, -7.081558243585513e-32,
7.249471490433169e-17, -2.611897368235813]], 'ieee_format': [[76.0, 1.0, 1.0,
-0.0, 0.0, -0.0], [1.0, 76.0, 1.0, -0.0, -0.0, 0.0], [1.0, 1.0, 76.0, 0.0, -0.0,
-0.0], [-0.0, -0.0, 0.0, -3.0, -0.0, -0.0], [0.0, -0.0, -0.0, -0.0, -3.0, 0.0],
[-0.0, 0.0, -0.0, -0.0, 0.0, -3.0]]}], {'formula_pretty': 'NaCl', 'material_id':
'mp-22862', 'elastic_tensor': {'raw': [[47.49860799851083, 11.905982188412978,
11.905982188412978, -2.8969882048812674e-16, -8.326672684688672e-17,
-4.0245584642661915e-16], [11.905982188412978, 47.49860799851083,
11.905982188412978, -8.326672684688672e-17, -2.8969882048812674e-16,
8.755854048151501e-16], [11.905982188412978, 11.905982188412978,
47.498607998510835, -2.7755575615628873e-17, -4.0245584642661934e-16,
-2.8969882048812674e-16], [-2.8969882048812674e-16, -8.326672684688672e-17,
-2.7755575615628873e-17, 12.346528650005286, 0.0, 4.548539333379926e-32],
[-8.326672684688672e-17, -2.8969882048812674e-16, -4.0245584642661934e-16, 0.0,
12.346528650005286, -3.426850095357504e-16], [-4.0245584642661915e-16,
8.755854048151501e-16, -2.8969882048812674e-16, 4.548539333379926e-32,
-3.426850095357504e-16, 12.346528650005297]], 'ieee_format': [[47.0, 12.0, 12.0,
0.0, -0.0, -0.0], [12.0, 47.0, 12.0, 0.0, -0.0, -0.0], [12.0, 12.0, 47.0, 0.0,
0.0, -0.0], [0.0, 0.0, 0.0, 12.0, -0.0, -0.0], [-0.0, -0.0, 0.0, -0.0, 12.0,
0.0], [-0.0, -0.0, -0.0, -0.0, 0.0, 12.0]]}}]Thought: The
elastic tensor for NaCl has been successfully retrieved. Now, I can provide the
elastic tensor for NaCl to the user.
```

Action:

...

{

> Finished chain.



Thought: It looks like there are two different elastic tensors for NaCl. The first one is  $\begin{bmatrix} 75.98 & 0.76 & 0.76 & 0.0 & 0.0 & 0.0 \\ 0.76 & 75.98 & 0.76 & 0.0 & 0.0 & 0.0 \\ 0.76 & 0.76 & 75.98 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -2.61 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -2.61 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -2.61 \end{bmatrix}$ . The second one is  $\begin{bmatrix} 47.49860799851083 & 11.905982188412978 & 11.905982188412978 & -2.8969882048812674e-16 & -8.326672684688672e-17 & -4.0245584642661915e-16 \\ 11.905982188412978 & 47.49860799851083 & 11.905982188412978 & -2.8969882048812674e-16 & 8.755854048151501e-16 & -2.7755575615628873e-17 \\ 11.905982188412978 & 11.905982188412978 & 47.49860799851083 & -2.7755575615628873e-17 & -4.0245584642661934e-16 & -2.8969882048812674e-16 \\ -2.8969882048812674e-16 & -8.326672684688672e-17 & -2.7755575615628873e-17 & 12.346528650005286 & 0.0 & 4.548539333379926e-32 \\ -8.326672684688672e-17 & -2.8969882048812674e-16 & -4.0245584642661934e-16 & 0.0 & 12.346528650005286 & -3.426850095357504e-16 \\ -4.0245584642661915e-16 & 8.755854048151501e-16 & -2.8969882048812674e-16 & 4.548539333379926e-32 & -3.426850095357504e-16 & 12.346528650005297 \end{bmatrix}$ .

Which one would you like to use?

> Finished chain.

```
[]: {'input': 'Please give me the elastic tensor of NaCl',
 'chat_history': [],
 'output': 'It looks like there are two different elastic tensors for NaCl. The first one is [[75.98, 0.76, 0.76, 0.0, 0.0, 0.0], [0.76, 75.98, 0.76, 0.0, 0.0, 0.0], [0.76, 0.76, 75.98, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, -2.61, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, -2.61, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, -2.61]]. The second one is [[47.49860799851083, 11.905982188412978, 11.905982188412978, -2.8969882048812674e-16, -8.326672684688672e-17, -4.0245584642661915e-16], [11.905982188412978, 47.49860799851083, 11.905982188412978, -2.8969882048812674e-16, 8.755854048151501e-16], [11.905982188412978, 11.905982188412978, 47.49860799851083, -2.7755575615628873e-17, -4.0245584642661934e-16, -2.8969882048812674e-16], [-2.8969882048812674e-16, -8.326672684688672e-17, -2.7755575615628873e-17, 12.346528650005286, 0.0, 4.548539333379926e-32], [-8.326672684688672e-17, -2.8969882048812674e-16, -4.0245584642661934e-16, 0.0, 12.346528650005286, -3.426850095357504e-16], [-4.0245584642661915e-16, 8.755854048151501e-16, -2.8969882048812674e-16, 4.548539333379926e-32, -3.426850095357504e-16, 12.346528650005297]].\n\nWhich one would you like to use?'}
```

[ ]: