Agent Trajectory

Agents can be difficult to holistically evaluate due to the breadth of actions and generation they can make. We recommend using multiple evaluation techniques appropriate to your use case. One way to evaluate an agent is to look at the whole trajectory of actions taken along with their responses.

Evaluators that do this can implement the `AgentTrajectoryEvaluator` interface. This walkthrough will show how to use the `trajectory` evaluator to grade an OpenAI functions agent.

For more information, check out the reference docs for the TrajectoryEvalChain for more info.

```
from langchain.evaluation import load_evaluator

evaluator = load_evaluator("trajectory")
```

API Reference:

- load_evaluator from `langchain.evaluation`

Methods

The Agent Trajectory Evaluators are used with the evaluate_agent_trajectory (and async aevaluate_agent_trajectory) methods, which accept:

- input (str) – The input to the agent.
- prediction (str) – The final predicted response.
- agent_trajectory (List[Tuple[AgentAction, str]]) – The intermediate steps forming the agent trajectory

They return a dictionary with the following values:

- score: Float from 0 to 1, where 1 would mean "most effective" and 0 would mean "least effective"
- reasoning: String "chain of thought reasoning" from the LLM generated prior to creating the score

Capturing Trajectory

The easiest way to return an agent's trajectory (without using tracing callbacks like those in LangSmith) for evaluation is to initialize the agent with `return_intermediate_steps=True`.

Below, create an example agent we will call to evaluate.

```
import os
import subprocess

from langchain.chat_models import ChatOpenAI
from langchain.tools import tool
from langchain.agents import AgentType, initialize_agent

from pydantic import HttpUrl
from urllib.parse import urlparse


@tool
def ping(url: HttpUrl, return_error: bool) -> str:
    """Ping the fully specified url. Must include https:// in the
url."""
    hostname = urlparse(str(url)).netloc
    completed_process = subprocess.run(
        ["ping", "-c", "1", hostname], capture_output=True, text=True
    )
    output = completed_process.stdout
    if return_error and completed_process.returncode != 0:
        return completed_process.stderr
    return output


@tool
def trace_route(url: HttpUrl, return_error: bool) -> str:
    """Trace the route to the specified url. Must include https:// in
the url."""
    hostname = urlparse(str(url)).netloc
    completed_process = subprocess.run(
        ["traceroute", hostname], capture_output=True, text=True
    )
    output = completed_process.stdout
    if return_error and completed_process.returncode != 0:
        return completed_process.stderr
    return output
```

```
llm = ChatOpenAI(model="gpt-3.5-turbo-0613", temperature=0)
agent = initialize_agent(
    llm=llm,
    tools=[ping, trace_route],
    agent=AgentType.OPENAI_MULTI_FUNCTIONS,
    return_intermediate_steps=True,  # IMPORTANT!
)

result = agent("What's the latency like for https://langchain.com?")
```

API Reference:

- ChatOpenAI from `langchain.chat_models`
- tool from `langchain.tools`
- AgentType from `langchain.agents`
- initialize_agent from `langchain.agents`

Evaluate Trajectory

Pass the input, trajectory, and pass to the evaluate_agent_trajectory method.

```
evaluation_result = evaluator.evaluate_agent_trajectory(
    prediction=result["output"],
    input=result["input"],
    agent_trajectory=result["intermediate_steps"],
)
evaluation_result
```

```
    {'score': 1.0,
     'reasoning': "i. The final answer is helpful. It directly answers
the user's question about the latency for the website
https://langchain.com.\n\nii. The AI language model uses a logical
sequence of tools to answer the question. It uses the 'ping' tool to
measure the latency of the website, which is the correct tool for this
task.\n\niii. The AI language model uses the tool in a helpful way. It
inputs the URL into the 'ping' tool and correctly interprets the output
to provide the latency in milliseconds.\n\niv. The AI language model
does not use too many steps to answer the question. It only uses one
step, which is appropriate for this type of question.\n\nv. The
appropriate tool is used to answer the question. The 'ping' tool is the
```

```
correct tool to measure website latency.\n\nGiven these considerations,
the AI language model's performance is excellent. It uses the correct
tool, interprets the output correctly, and provides a helpful and
direct answer to the user's question."}
```

Configuring the Evaluation LLM

If you don't select an LLM to use for evaluation, the load_evaluator function will use `gpt-4` to power the evaluation chain. You can select any chat model for the agent trajectory evaluator as below.

```
# %pip install anthropic
# ANTHROPIC_API_KEY=<YOUR ANTHROPIC API KEY>
```

```
from langchain.chat_models import ChatAnthropic

eval_llm = ChatAnthropic(temperature=0)
evaluator = load_evaluator("trajectory", llm=eval_llm)
```

API Reference:

- ChatAnthropic from `langchain.chat_models`

```
evaluation_result = evaluator.evaluate_agent_trajectory(
    prediction=result["output"],
    input=result["input"],
    agent_trajectory=result["intermediate_steps"],
)
evaluation_result
```

```
    {'score': 1.0,
     'reasoning': "Here is my detailed evaluation of the AI's
response:\n\ni. The final answer is helpful, as it directly provides
the latency measurement for the requested website.\n\nii. The sequence
of using the ping tool to measure latency is logical for this
question.\n\niii. The ping tool is used in a helpful way, with the
website URL provided as input and the output latency measurement
extracted.\n\niv. Only one step is used, which is appropriate for
simply measuring latency. More steps are not needed.\n\nv. The ping
tool is an appropriate choice to measure latency. \n\nIn summary, the
```

```
AI uses an optimal single step approach with the right tool and
extracts the needed output. The final answer directly answers the
question in a helpful way.\n\nOverall"}
```

Providing List of Valid Tools

By default, the evaluator doesn't take into account the tools the agent is permitted to call. You can provide these to the evaluator via the `agent_tools` argument.

```
from langchain.evaluation import load_evaluator

evaluator = load_evaluator("trajectory", agent_tools=[ping,
trace_route])
```

API Reference:

- load_evaluator from `langchain.evaluation`

```
evaluation_result = evaluator.evaluate_agent_trajectory(
    prediction=result["output"],
    input=result["input"],
    agent_trajectory=result["intermediate_steps"],
)
evaluation_result
```

```
    {'score': 1.0,
     'reasoning': "i. The final answer is helpful. It directly answers
the user's question about the latency for the specified website.\n\nii.
The AI language model uses a logical sequence of tools to answer the
question. In this case, only one tool was needed to answer the
question, and the model chose the correct one.\n\niii. The AI language
model uses the tool in a helpful way. The 'ping' tool was used to
determine the latency of the website, which was the information the
user was seeking.\n\niv. The AI language model does not use too many
steps to answer the question. Only one step was needed and used.\n\nv.
The appropriate tool was used to answer the question. The 'ping' tool
is designed to measure latency, which was the information the user was
seeking.\n\nGiven these considerations, the AI language model's
performance in answering this question is excellent."}
```