Custom Trajectory Evaluator

You can make your own custom trajectory evaluators by inheriting from the AgentTrajectoryEvaluator class and overwriting the `_evaluate_agent_trajectory` (and `_aevaluate_agent_action`) method.

In this example, you will make a simple trajectory evaluator that uses an LLM to determine if any actions were unnecessary.

```
from typing import Any, Optional, Sequence, Tuple
from langchain.chat_models import ChatOpenAI
from langchain.chains import LLMChain
from langchain.schema import AgentAction
from langchain.evaluation import AgentTrajectoryEvaluator


class StepNecessityEvaluator(AgentTrajectoryEvaluator):
    """Evaluate the perplexity of a predicted string."""

    def __init__(self) -> None:
        llm = ChatOpenAI(model="gpt-4", temperature=0.0)
        template = """Are any of the following steps unnecessary in
answering {input}? Provide the verdict on a new line as a single "Y"
for yes or "N" for no.

        DATA
        ------
        Steps: {trajectory}
        ------

        Verdict:"""
        self.chain = LLMChain.from_string(llm, template)

    def _evaluate_agent_trajectory(
        self,
        *,
        prediction: str,
        input: str,
        agent_trajectory: Sequence[Tuple[AgentAction, str]],
        reference: Optional[str] = None,
        **kwargs: Any,
    ) -> dict:
```

```
        vals = [
            f"{i}: Action=[{action.tool}] returned observation =
[{observation}]"
            for i, (action, observation) in enumerate(agent_trajectory)
        ]
        trajectory = "\n".join(vals)
        response = self.chain.run(dict(trajectory=trajectory,
input=input), **kwargs)
        decision = response.split("\n")[-1].strip()
        score = 1 if decision == "Y" else 0
        return {"score": score, "value": decision, "reasoning":
response}
```

API Reference:

- ChatOpenAI from `langchain.chat_models`
- LLMChain from `langchain.chains`
- AgentAction from `langchain.schema`
- AgentTrajectoryEvaluator from `langchain.evaluation`

The example above will return a score of 1 if the language model predicts that any of the actions were unnecessary, and it returns a score of 0 if all of them were predicted to be necessary. It returns the string 'decision' as the 'value', and includes the rest of the generated text as 'reasoning' to let you audit the decision.

You can call this evaluator to grade the intermediate steps of your agent's trajectory.

```
evaluator = StepNecessityEvaluator()

evaluator.evaluate_agent_trajectory(
    prediction="The answer is pi",
    input="What is today?",
    agent_trajectory=[
        (
            AgentAction(tool="ask", tool_input="What is today?",
log=""),
            "tomorrow's yesterday",
        ),
        (
            AgentAction(tool="check_tv", tool_input="Watch tv for half
hour", log=""),
            "bzzz",
        ),
```

```
    ],
)
```

```
{'score': 1, 'value': 'Y', 'reasoning': 'Y'}
```