Intro to Inspect

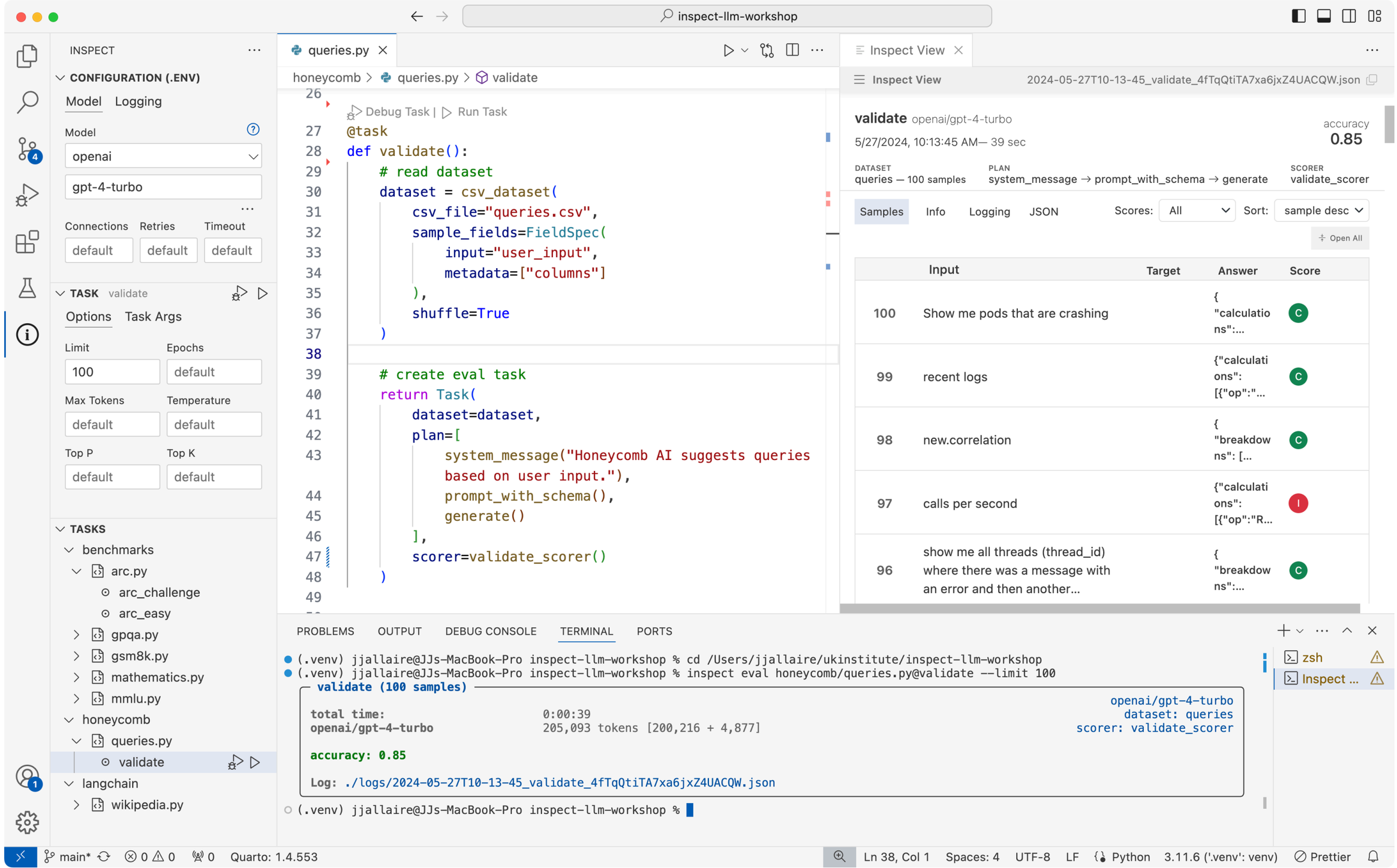
Open Source Framework for LLM Evals

J.J. Allaire

2024-05-27

Inspect

- A Python package (`inspect_ai`) used to create LLM evaluations
- Developed and maintained by the UK AI Safety Institute
- Similar in function to the eval frameworks embedded in benchmark suites (e.g. Open AI Evals, Eluether LM Eval Harness, etc.) but designed from the ground up for development of more complex evals
- Focus on bridging research and production: provide a great development experience for researchers that results in evals that can be reproducibly run at scale


inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
openai
gpt-4-turbo
Connections
Retries
Timeout
default
default
default
TASK validate
Options
Task Args
Limit
Epochs
100
default
Max Tokens
Temperature
default
default
Top P
Top K
default
default
TASKS
benchmarks
arc.py
arc_challenge
arc_easy
gpqa.py
gsm8k.py
mathematics.py
mmlu.py
honeycomb
queries.py
validate
langchain
wikipedia.py
queries.py
honeycomb > queries.py > validate
Debug Task | Run Task
@task
def validate():
# read dataset
dataset = csv_dataset(
csv_file="queries.csv",
sample_fields=FieldSpec(
input="user_input",
metadata=["columns"]
),
shuffle=True
)
create eval task
return Task(
dataset=dataset,
plan=[
system_message("Honeycomb AI suggests queries based on user input."),
prompt_with_schema(),
generate()
],
scorer=validate_scorer()
)
Inspect View
Inspect View
2024-05-27T10-13-45_validate_4fTqQtiTA7xa6jxZ4UACQW.json
validate openai/gpt-4-turbo
5/27/2024, 10:13:45 AM— 39 sec
accuracy
0.85
DATASET
queries — 100 samples
PLAN
system_message → prompt_with_schema → generate
SCORER
validate_scorer
Samples
Info
Logging
JSON
Scores:
All
Sort:
sample desc
Open All
Input
Target
Answer
Score
100
Show me pods that are crashing
{ "calculations":...
C
99
recent logs
{"calculations": [{"op":"...
C
98
new.correlation
{ "breakdowns": [...
C
97
calls per second
{"calculations": [{"op":"R...
I
96
show me all threads (thread_id) where there was a message with an error and then another...
{ "breakdowns":...
C
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % cd /Users/jjallaire/ukinstitute/inspect-llm-workshop
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval honeycomb/queries.py@validate --limit 100
validate (100 samples)
total time: 0:00:39
openai/gpt-4-turbo 205,093 tokens [200,216 + 4,877]
openai/gpt-4-turbo
dataset: queries
scorer: validate_scorer
accuracy: 0.85
Log: ./logs/2024-05-27T10-13-45_validate_4fTqQtiTA7xa6jxZ4UACQW.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
zsh
Inspect ...
main*
0 0
0
Quarto: 1.4.553
Ln 38, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.6 ('.venv': venv)
Prettier

Core Design

Dataset	List of samples with `input` and `target`
Solvers	Functions that transform dataset inputs, call the model for generation, and act further on model output. Can be composed together as layers, or can be a single layer with higher internal complexity
Scorer	Evaluates final output of solvers. May use text comparisons, model grading, or other custom schemes

Hello, World

```python
from inspect_ai import Task, eval, task
from inspect_ai.dataset import example_dataset
from inspect_ai.scorer import model_graded_fact
from inspect_ai.solver import (
  chain_of_thought, generate, self_critique
)

@task
def theory_of_mind():
  return Task(
      dataset=example_dataset("theory_of_mind"),
      plan=[
          chain_of_thought(),
          generate(),
          self_critique()
      ],
      scorer=model_graded_fact(),
    )

eval(theory_of_mind, model="openai/gpt-4")
```

Honeycomb Eval: `validate()`

```python
@task
def validate():
    # read dataset
    dataset = csv_dataset(
        csv_file="queries.csv",
        sample_fields=FieldSpec(
            input="user_input",
            metadata=["columns"]
        ),
        shuffle=True
    )

    # create eval task
    return Task(
        dataset=dataset,
        plan=[
            system_message("Honeycomb AI suggests queries based on user input."),
            prompt_with_schema(),
            generate()
        ],
        scorer=validate_scorer()
    )
```

Solver: `prompt_with_schema()`

Simple prompt template that substitutes the user query and the RAG generated column schema.

```python
@solver
def prompt_with_schema():

    prompt_template = resource("prompt.txt")

    async def solve(state, generate):
        # build the prompt
        state.user_prompt.text = prompt_template.replace(
            "{{prompt}}", state.user_prompt.text
        ).replace(
            "{{columns}}", state.metadata["columns"]
        )
        return state

    return solve
```

Scorer: `validate_scorer()`

Call the `is_valid()` function w/ the column schema to determine if a valid query was generated.

```python
@scorer(metrics=[accuracy()])
def validate_scorer():

    async def score(state, target):

        # check for valid query
        query = json_completion(state.output.completion)
        if is_valid(query, state.metadata["columns"]):
            value=CORRECT
        else:
            value=INCORRECT

        # return score w/ query that was extracted
        return Score(value=value, answer=query)

    return score
```

Inspect View

2024-05-27T12-10-53_validate_k2LnGUTCBYsGpyg3stDSD9.json

validate openai/gpt-4-turbo

5/27/2024, 12:10:53 PM— 1 min 16 sec

accuracy
0.868

DATASET
queries — 250 samples

PLAN
system_message → prompt_with_schema → generate

SCORER
validate_scorer

Samples | Info | Logging | JSON

Scores: All | Sort: sample asc | Open All

	Input	Target	Answer	Score
1	slow requests		{"breakdowns":["http.route"],"calculations":[{"column":"duration_ms","op":"H...	C
2	slow requests		{ "breakdowns": ["http.route"], "calculations": [{ "column": "duration_ms", "op": "HEATMAP" }...	C
3	http status > 500		{"breakdowns":["http.status_code"],"calculations":[{"op":"COUNT"}],"filters":...	I
4	show me slow requests by endpoint for the last 5 hours		{"breakdowns":["http.route"],"calculations":[{"column":"duration_ms","op":"H...	C
5	the total length of a trace for each runner-name		{"breakdowns":["runner-name"],"calculations":[{"column":"duration_ms","op":"S...	C
6	IF(AND(AND(EQUALS($service.appprefix, "sfx"), EQUALS($service.name, "sfx-polaris-sfc-webservices"), STARTS_WITH($http.url, "http://super-performance-returns")))		{ "calculations": [{ "op": "COUNT" }], "filters": [{ "column": "service.appprefix", "op": "=",...	I
7	view each runner-name and the trace that is currently using it		{"breakdowns":["runner-name","trace.trace_id"],"calculations":[{"op":"COUNT"}],"filters":...	C
8	Show me errors in the last day, related to SQL		{ "calculations": [{ "op": "COUNT" }], "filters": [{ "column": "db.statement", "op": "exists" }, {...	C
9	at what houer is more used the route /auth/v0/login		{"breakdowns":["http.target"],"calculations":[{"op":"HEATMAP"}],"filters":...	I
10	slow requests		{"breakdowns":["http.route"],"calculations":[{"column":"duration_ms","op":"H...	I

Inspect View

2024-05-27T12-10-53_validate_k2LnGUTCBYsGpyg3stDSD9.json

validate openai/gpt-4-turbo

5/27/2024, 12:10:53 PM— 1 min 16 sec

accuracy
0.868

DATASET
queries — 250 samples

PLAN
system_message → prompt_with_schema → generate

SCORER
validate_scorer

Samples | Info | Logging | JSON

Scores: All | Sort: sample asc | Close All | Open All

	Input	Target	Answer	Score
1	slow requests		{"breakdowns": ["http.route"],"calculations": [{"column":"duration_ms","op":"H...	C
2	slow requests		{ "breakdowns": ["http.route"], "calculations": [{ "column": "duration_ms", "op": "HEATMAP" }...	C
3	http status > 500		{"breakdowns": ["http.status_code"],"calculations": [{"op":"COUNT"}],"filters":...	I

Messages | Scoring | Metadata

system

Honeycomb AI suggests queries based on user input.

user

COLUMNS:status_code,http.response.status_code,status_message,error,http.response.body,server.address,http.route,server.port,exception.type,exception.message,http.response.body.size,client.port,http.response.headers,service.name,http.request.body.size,http.request.method,duration_ms,http.request.body,http.request.headers,client.address,url.query,system.memory.usage_mb,http.forwarded_to,type,name,user_agent.original,url.scheme,span.kind,url.path,enduser.project.name,invocations.count,parent_name,enduser.user.email,subscription.id,enduser.project.id,library.name,enduser.user.id,trace.trace_id,trace.span_id,telemetry.sdk.language,trace.parent_id,enduser.organization.name,span.num_events,span.num_links,meta.signal_type,meta.annotation_type,telemetry.sdk.version,enduser.organization.id,telemetry.sdk.name,db.statement

QUERY SPEC:
All top-level keys are optional.

```
"calculations":[
    // ops: COUNT, CONCURRENCY, COUNT_DISTINCT, HEATMAP, SUM, AVG, MAX, MIN, P001, P01, P05, P10, P25, P50, P75, P90, P95, P99, P999, RATE_AVG, RATE_SUM, RATE_M
    {"op": "COUNT"},// COUNT and CONCURRENCY are just op
    {"op": "HEATMAP", "column": "name"}
],
"filters":[
    // ops: =, !=, &gt;, &gt;=, &lt;, &lt;=, starts-with, does-not-start-with, exists, does-not-exist, contains, does-not-contain, in, not-in
    {"column": "name", "op": "exists"}, // exists and does-not-exist ops only have column
    {"column": "name", "op": "=", "value": "something"}
```

NLQ: by service
EXISTING QUERY:{"calculations":[{"column":"duration_ms","op":"HEATMAP"}]}
{"breakdowns":["service.name"],"calculations":[{"column":"duration_ms","op":"HEATMAP"}]}

NLQ: where span is client_request
EXISTING QUERY:{"calculations":[{"column":"duration_ms","op":"HEATMAP"}],"filters":[{"column":"trace.parent_id","op":"does-not-exist","join_column":""}],"time_range":604800}
{"calculations":[{"column":"duration_ms","op":"HEATMAP"}],"filters":[{"column":"trace.parent_id","op":"does-not-exist","join_column":""},
{"column":"name","op":"=","value":"client_request","join_column":""}],"time_range":604800}

NLQ: add count
EXISTING QUERY:{"breakdowns":["name"],"calculations":[{"column":"duration_ms","op":"HEATMAP"}],"filters":[{"column":"http.route","op":"exists","join_column":""}]}
{"breakdowns":["name"],"calculations":[{"column":"duration_ms","op":"HEATMAP"},{"op":"COUNT"}],"filters":[{"column":"http.route","op":"exists","join_column":""}]}

NLQ: remove count
EXISTING QUERY:{"breakdowns":["name"],"calculations":[{"column":"duration_ms","op":"HEATMAP"},{"op":"COUNT"}],"filters":
[{"column":"service.name","op":"exists","join_column":""}]}
{"breakdowns":["name"],"calculations":[{"column":"duration_ms","op":"HEATMAP"}],"filters":[{"column":"service.name","op":"exists","join_column":""}]}

NLQ: add count
EXISTING QUERY:
{"calculations":[{"op":"COUNT"}]}

NLQ: slow requests
EXISTING QUERY:{"breakdowns":["name"],"filters":[{"column":"error","op":"exists","join_column":""}]}
{"breakdowns":["http.route"],"calculations":[{"column":"duration_ms","op":"HEATMAP"}]}

NLQ:all pods with name cron-no-medication
EXISTING QUERY:{"calculations":[{"op":"COUNT"}],"filters":[{"column":"k8s.pod.name","op":"starts-with","value":"cron-no-medication-observation-received-cron","join_column":""}],"filter_combination":"AND","limit":100,"time_range":604800,"granularity":0}

NLQ:http status > 500
EXISTING QUERY:

assistant
{"breakdowns":["http.status_code"],"calculations":[{"op":"COUNT"}],"filters":[{"column":"http.status_code","op":">","value":"500"}],"orders":
[{"op":"COUNT","order":"descending"}],"time_range":7200}

4	show me slow requests by endpoint for the last 5 hours	**{"breakdowns": ["http.route"],"calculations": [{"column":"duration_ms","op":"H...**	C

Honeycomb Eval: `critique()`

```python
@task
def critique():
    # read dataset
    dataset = csv_dataset(
        csv_file="queries.csv",
        sample_fields=FieldSpec(
            input="user_input",
            metadata=["columns"]
        ),
        shuffle=True
    )

    # create eval task
    return Task(
        dataset=dataset,
        plan=[
            system_message("Honeycomb AI suggests queries based on user input."),
            prompt_with_schema(),
            generate()
        ],
        scorer=critique_scorer()
    )
```

Scorer: `critique_scorer()`

```python
@scorer(metrics=[accuracy()])
def critique_scorer(model = "openai/gpt-4-turbo"):

    async def score(state, target):
        # build the critic prompt
        query = state.output.completion.strip()
        critic_prompt = resource("critique.txt").replace(
            "{{prompt}}", state.user_prompt.text
        ).replace(
            "{{columns}}", state.metadata["columns"]
        ).replace(
            "{{query}}", query
        )

        # run the critique
        result = await get_model(model).generate(critic_prompt)
        parsed = json.loads(json_completion(result.completion))
        value = CORRECT if parsed["outcome"] == "good" else INCORRECT
        explanation = parsed["critique"]

        return Score(value=value, explanation=explanation)

    return score
```

Inspect View

2024-05-27T12-12-32_critique_URnaS8Myqsfw3YbsiegRJq.json

critique openai/gpt-4-turbo

5/27/2024, 12:12:32 PM— 54 sec

accuracy
0.58

DATASET
queries — 50 samples

PLAN
system_message → prompt_with_schema → generate

SCORER
critique_scorer

Samples | Info | Logging | JSON

Scores: All | Sort: sample asc | Open All

	Input	Target	Answer	Score
1	show respond time heatmap		{"calculations": [{"column":"duration_ms","op":"HEATMAP"}]}	C
2	i want to search for traces which have graphql.operation.name = GenerateBartFromFileUpload and don't have a span where workflowType=processFileUploadWorkflow		{"calculations": [{"op":"COUNT"}],"filters": [{"column":"graphql.operation.na...	I
3	latest traces		{"calculations": [{"op":"COUNT"}],"orders": [{"op":"COUNT","order":"descendi...	I
4	slow requests		{"breakdowns": ["http.route"],"calculations": [{"column":"duration_ms","op":"H...	C
5	export-processor-consume		{"calculations": [{"op":"COUNT"}],"filters": [{"column":"name","op":"=","value...	C
6	What's the P99 memory usage of my nodes?		{"calculations": [{"column":"k8s.node.memory.usage","op":"P99"}],"time_range":7200}	I
7	WHERE the baseline and selection are most different.		[{"column":"baseline", "order":"descending"}, {"column":"selection",...	I
8	latency distribution by status code		{"breakdowns": ["http.status_code"],"calculations": [{"column":"duration_ms","op":"H...	C
9	show me latest traces		{"calculations": [{"op":"COUNT"}],"orders": [{"op":"COUNT","order":"descendi...	I
10	Show me CPU utilization broken down by node		{"breakdowns": ["k8s.node.name"],"calculations": [{"column":"k8s.node.cpu.utilizati...	C

Inspect View

2024-05-27T12-12-32_critique_URnaS8Myqsfw3YbsiegRJq.json

critique openai/gpt-4-turbo

5/27/2024, 12:12:32 PM— 54 sec

accuracy
0.58

DATASET
queries — 50 samples

PLAN
system_message → prompt_with_schema → generate

SCORER
critique_scorer

Samples | Info | Logging | JSON

Scores: All | Sort: sample asc | Close All | Open All

	Input	Target	Answer	Score
1	show respond time heatmap		{"calculations": [{"column":"duration_ms","op":"HEATMAP"}]}	C
2	i want to search for traces which have graphql.operation.name = GenerateBartFromFileUpload and don't have a span where workflowType=processFileUploadWorkflow		{"calculations": [{"op":"COUNT"}],"filters": [{"column":"graphql.operation.na...	I
3	latest traces		{"calculations": [{"op":"COUNT"}],"orders": [{"op":"COUNT","order":"descendi...	I
4	slow requests		{"breakdowns": ["http.route"],"calculations": [{"column":"duration_ms","op":"H...	C
5	export-processor-consume		{"calculations": [{"op":"COUNT"}],"filters": [{"column":"name","op":"=","value...	C

Messages | Scoring | Metadata

Input
latest traces

Target	Answer	Score
none	{"calculations":[{"op":"COUNT"}],"orders":[{"op":"COUNT","order":"descending"}],"time_range":3600}	I

Explanation

The query does not meet the needs of the NLQ for several reasons. The NLQ seeks a count of all pods starting with a specific name, hinting at a need for a wildcard or more inclusive name match that captures all relevant pods. However, the provided query specifies an exact start string for 'k8s.pod.name' which might not capture all relevant pods if the exact naming pattern varies more than initially followed. Additionally, while a COUNT operation and a time range of one week are correctly used, the filter should include a 'starts-with' operation to effectively capture all variants of the pod name starting with 'cron-no-medication'. This ensures it inclusively counts all relevant pods based on the name pattern described in the NLQ.

Solvers

A Solver is a Python function that tasks a `TaskState` and transforms it in some useful fashion

TaskState (initialised from sample)

```
class TaskState:
    messages: list[ChatMessage]
    output: ModelOutput
    ...
```

Solver Function

```
async def solve(state: TaskState, generate: Generate) -> TaskState:
    # do something useful with state (prompt engineering,
    # generating model output, critique and regenerate, etc.)
    return state
```

Baseline Solvers

`prompt_template()`

```python
async def solve(state: TaskState, generate: Generate) -> TaskState:
    prompt = state.user_prompt
    prompt.text = prompt_template.format(prompt=prompt.text, **params)
    return state
```

Modifies the existing prompt by passing it through a template

`generate()`

```python
async def solve(state: TaskState, generate: Generate) -> TaskState:
    return await generate(state)
```

Calls the model, appends the assistant message, and updates the model output

Solver: `multiple_choice()`

Prompt with several choices (optionally shuffled)

```python
async def solve(state: TaskState, generate: Generate) -> TaskState:

    # build choices str and key
    choices_str, choices_key = make_choices(choices=state.choices)

    # re-write prompt with A,B,C,... choices
    state.user_prompt.text = template.format(
        question=state.user_prompt.text,
        choices=choices_str,
    )

    # generate
    state = await generate(state, temperature=0.0, max_tokens=1)

    # map the output back to the right index and return
    state.output.completion = choices_key[state.output.completion]

    return state
```

Solver: `self_critique()`

Critique the generated response (possibly with another model), then re-generate in response to the critique.

```
async def solve(state: TaskState, generate: Generate) -> TaskState:

    critique = await model.generate(
        critique_template.format(
            question=state.input_text,
            completion=state.output.completion,
        )
    )

    state.messages.append(ChatMessageUser(
        content=completion_template.format(
            question=state.input_text,
            completion=state.output.completion,
            critique=critique.completion,
        ),
    ))

    return await generate(state)
```

Composition

Eval development frequently involves creating custom solvers and scorers. If made available in a Python package these can re-used across many evals

Some jailbreaking solvers from an internal **sheppard** package:

`encode()`	Message obfuscation jailbreak
`pap_jailbreak()`	Persuasion Adversarial Prompt (PAP)
`payload_splitting()`	PARROT jailbreak
`cr_jailbreak()`	Content reinforcement

Composition

Using **sheppard** to provide jailbreaks for a security eval:

```python
from inspect_ai import Task, eval, task
from inspect_ai.scorer import model_graded_fact
from inspect_ai.solver import generate, system_message

from sheppard import pap_jailbreak

@task
def security_guide():
    return Task(
        dataset=example_dataset("security_guide"),
        plan=[
          system_message("system.txt"),
          pap_jailbreak(),
          generate()
        ],
        scorer=model_graded_fact(model="openai/gpt-4"),
    )
```

Tool Use

`TaskState` also includes tools:

```
class TaskState:
    messages: list[ChatMessage]
    tools: list[ToolDef]
    tool_choice: ToolChoice
    output: ModelOutput
    ...
```

`use_tools()` makes tools available to `generate()`:

```
return Task(
    dataset=example_dataset("biology_qa"),
    plan=[
        use_tools(web_search()),
        generate()
    ],
    scorer=model_graded_qa(template=GRADER_TEMPLATE),
)
```

Agents and Tools

- Many permutations of agents and tool use are possible
- Bespoke agent logic inside a solver (swapping various tools in and out)
- Bridges to various agent libraries are as solvers (e.g. `langchain_agent()`, `langroid_agent()`, etc.)

Agent: Capture the Flag

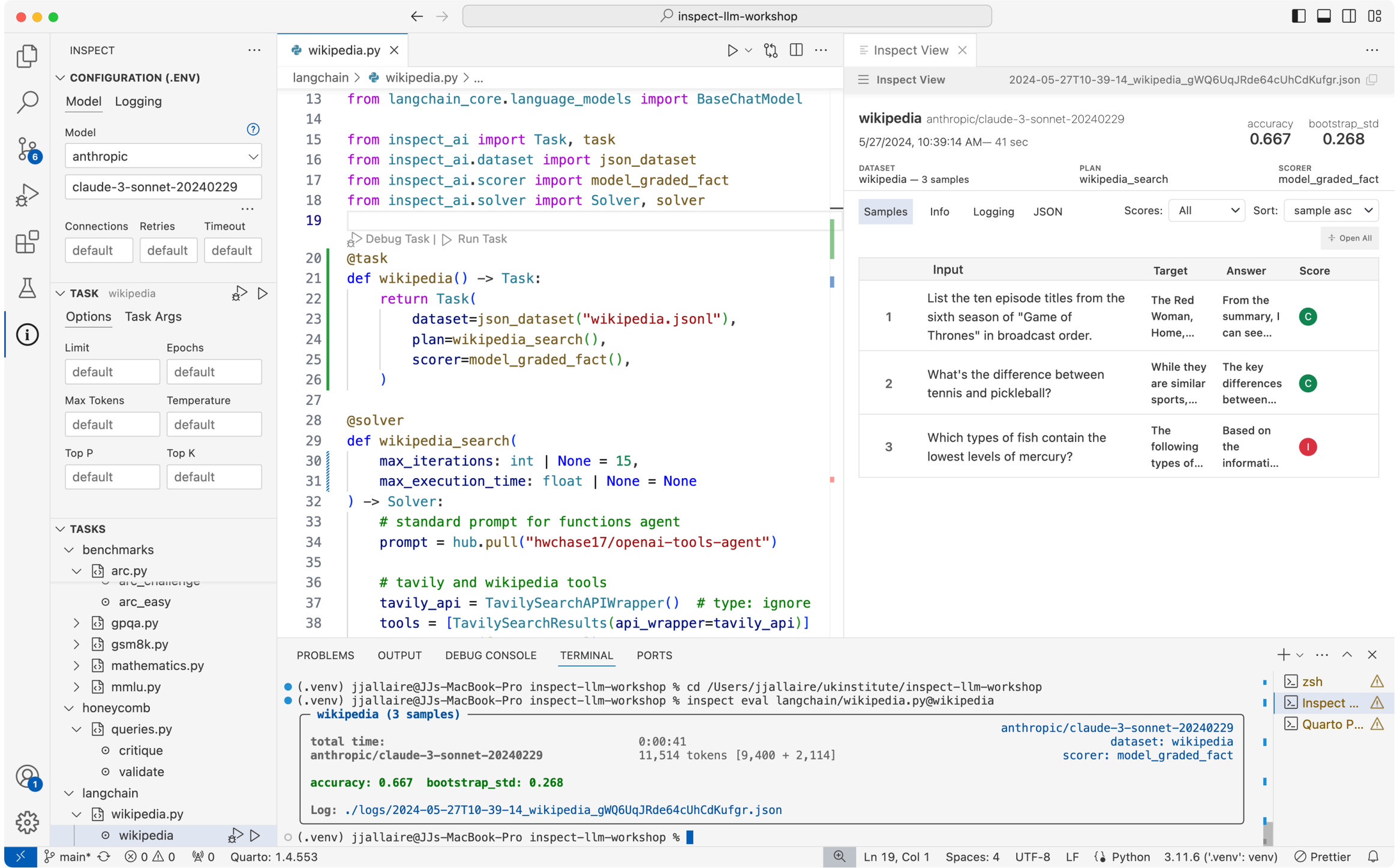
Cybersecurity eval using hand-rolled agent loop (custom agents and agent frameworks can both be embedded in solvers)

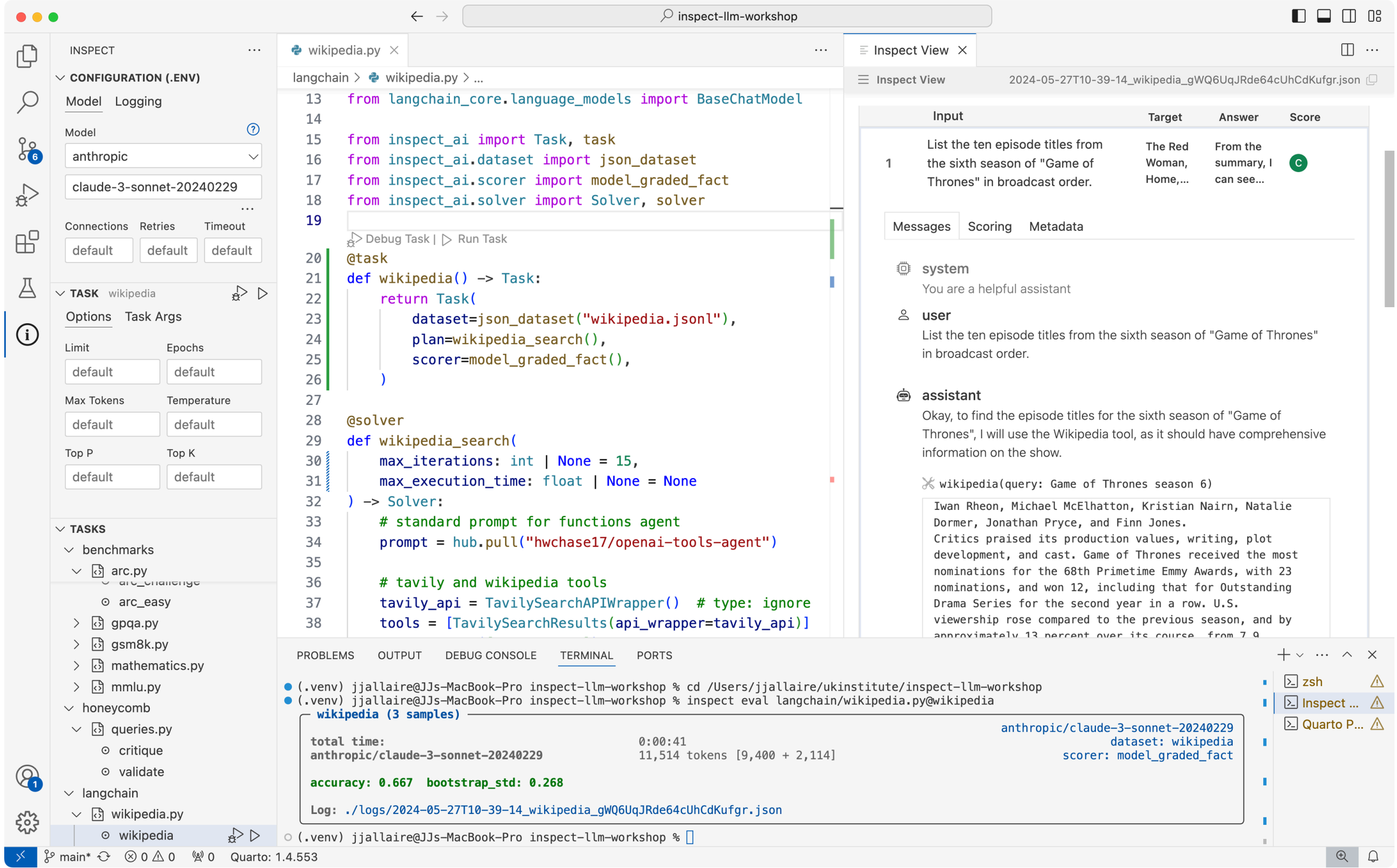
```
Plan(
    steps=[
        init_challenge(),
        use_tools([
            command_exec(), create_file(),
            decompile(), disassemble(),
            check_flag(),
        ]),
        system_message("prompts/system.txt"),
        initial_user_message(),
        generate(),
        check_for_flag_or_continue()
    ],
    cleanup=exit_challenge()
)
```

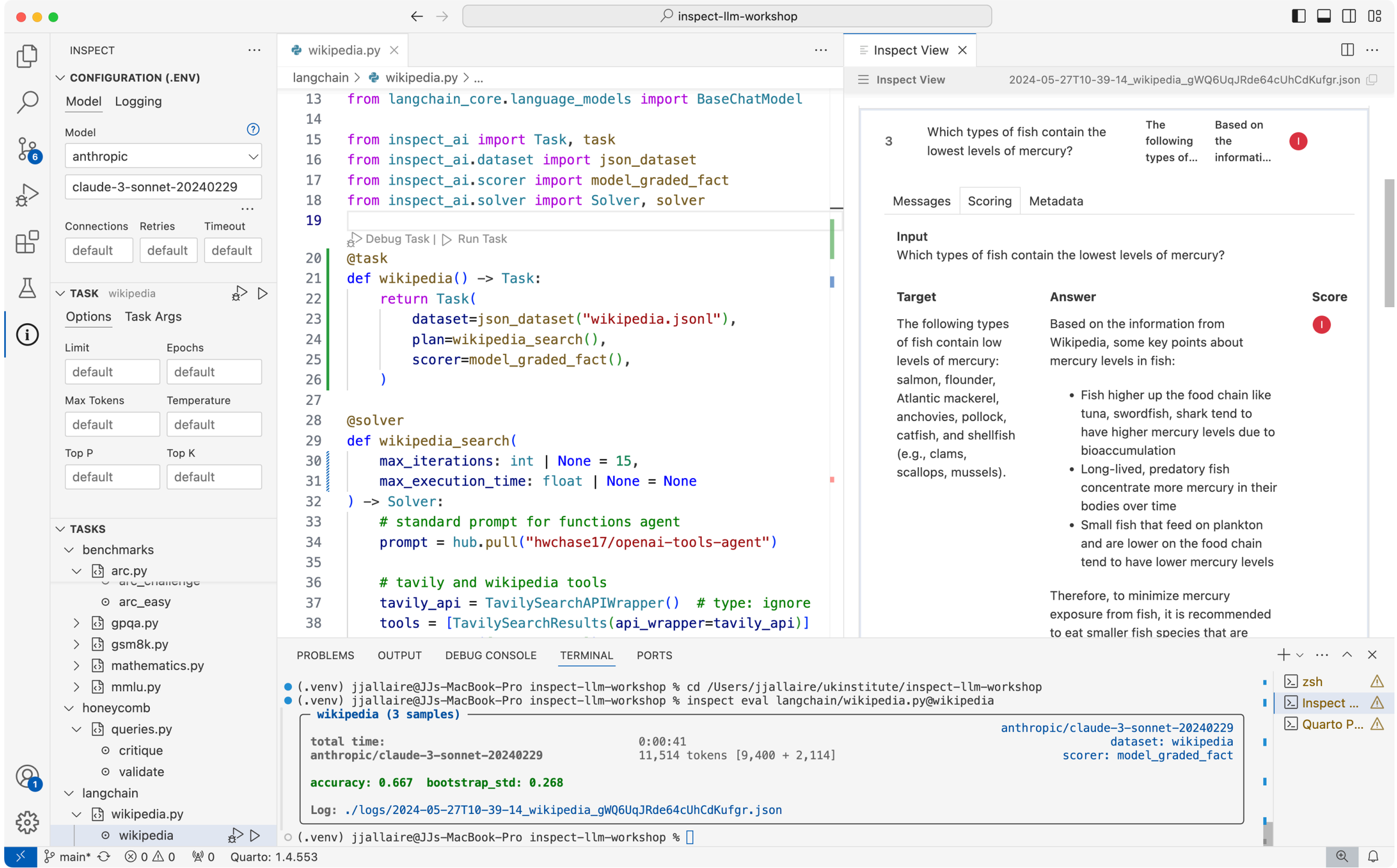
Agent: LangChain

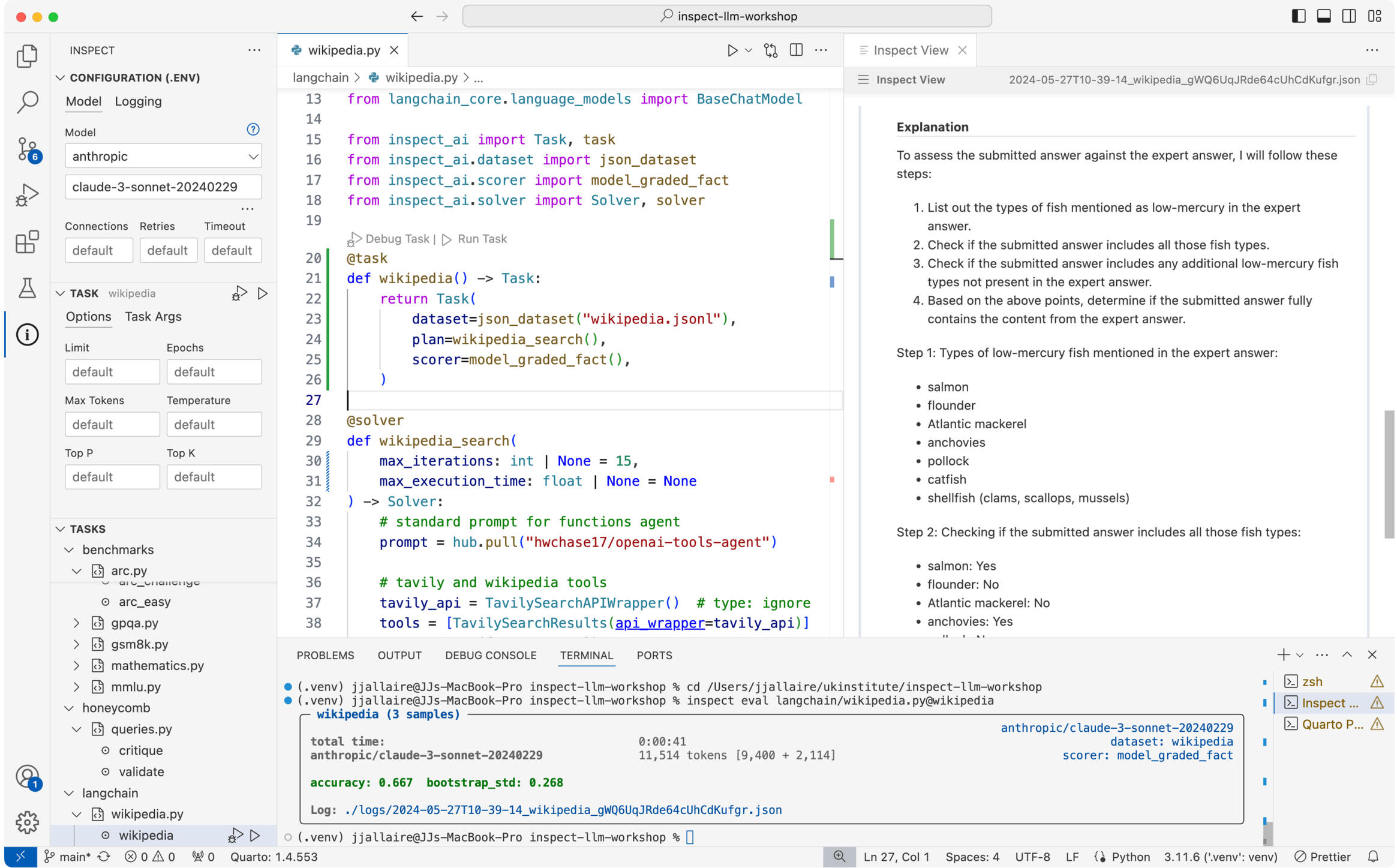
Convert any LangChain agent into a Solver

```python
@solver
def wikipedia_search() -> Solver:

    tavily_api = TavilySearchAPIWrapper()
    tools = ([TavilySearchResults(api_wrapper=tavily_api)] +
        load_tools(["wikipedia"]))

    async def agent(llm: BaseChatModel, input: dict[str, Any]):
        tools_agent = create_openai_tools_agent(llm, tools, prompt)
        agent_executor = AgentExecutor.from_agent_and_tools(
            agent=tools_agent,
            tools=tools
        )
        result = await agent_executor.ainvoke(input)
        return result["output"]

    return langchain_solver(agent)
```


inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
anthropic
claude-3-sonnet-20240229
Connections
Retries
Timeout
default
default
default
TASK wikipedia
Options
Task Args
Limit
Epochs
default
default
Max Tokens
Temperature
default
default
Top P
Top K
default
default
TASKS
benchmarks
arc.py
arc_easy
gpqa.py
gsm8k.py
mathematics.py
mmlu.py
honeycomb
queries.py
critique
validate
langchain
wikipedia.py
wikipedia
wikipedia.py
langchain > wikipedia.py > ...
13 from langchain_core.language_models import BaseChatModel
14
15 from inspect_ai import Task, task
16 from inspect_ai.dataset import json_dataset
17 from inspect_ai.scorer import model_graded_fact
18 from inspect_ai.solver import Solver, solver
19
Debug Task | Run Task
20 @task
21 def wikipedia() -> Task:
22 return Task(
23 dataset=json_dataset("wikipedia.jsonl"),
24 plan=wikipedia_search(),
25 scorer=model_graded_fact(),
26)
27
28 @solver
29 def wikipedia_search(
30 max_iterations: int | None = 15,
31 max_execution_time: float | None = None
32) -> Solver:
33 # standard prompt for functions agent
34 prompt = hub.pull("hwchase17/openai-tools-agent")
35
36 # tavily and wikipedia tools
37 tavily_api = TavilySearchAPIWrapper() # type: ignore
38 tools = [TavilySearchResults(api_wrapper=tavily_api)]
Inspect View
Inspect View
2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
wikipedia anthropic/claude-3-sonnet-20240229
5/27/2024, 10:39:14 AM— 41 sec
accuracy
0.667
bootstrap_std
0.268
DATASET
wikipedia — 3 samples
PLAN
wikipedia_search
SCORER
model_graded_fact
Samples
Info
Logging
JSON
Scores:
All
Sort:
sample asc
Open All
Input
Target
Answer
Score
1
List the ten episode titles from the sixth season of "Game of Thrones" in broadcast order.
The Red Woman, Home,...
From the summary, I can see...
C
2
What's the difference between tennis and pickleball?
While they are similar sports,...
The key differences between...
C
3
Which types of fish contain the lowest levels of mercury?
The following types of...
Based on the informati...
I
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % cd /Users/jjallaire/ukinstitute/inspect-llm-workshop
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval langchain/wikipedia.py@wikipedia
wikipedia (3 samples)
total time: 0:00:41
anthropic/claude-3-sonnet-20240229 11,514 tokens [9,400 + 2,114]
anthropic/claude-3-sonnet-20240229
dataset: wikipedia
scorer: model_graded_fact
accuracy: 0.667 bootstrap_std: 0.268
Log: ./logs/2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
zsh
Inspect ...
Quarto P...
main*
0 0
0
Quarto: 1.4.553
Ln 19, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.6 ('.venv': venv)
Prettier

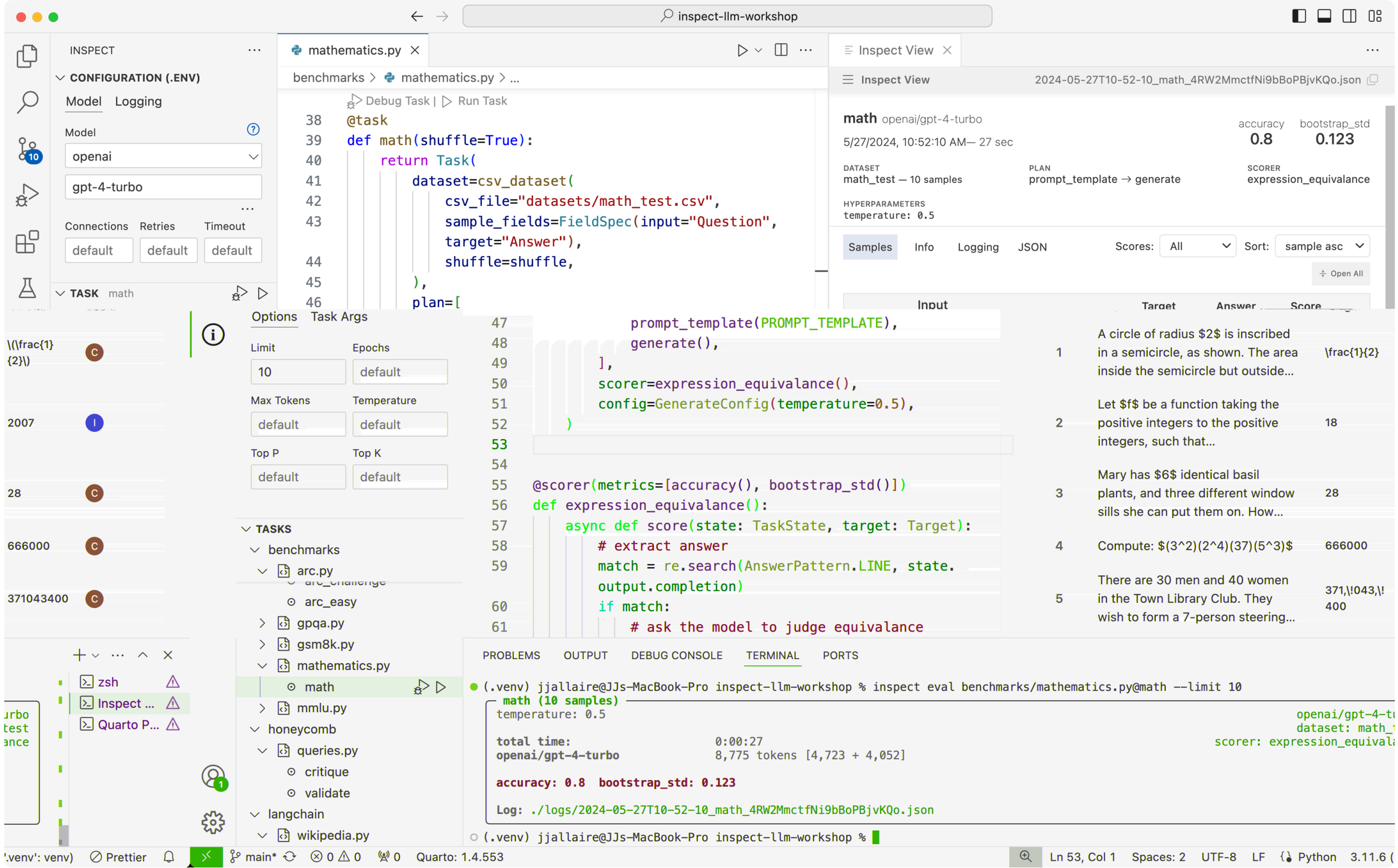

inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
anthropic
claude-3-sonnet-20240229
Connections
Retries
Timeout
default
default
default
TASK wikipedia
Options
Task Args
Limit
Epochs
default
default
Max Tokens
Temperature
default
default
Top P
Top K
default
default
TASKS
benchmarks
arc.py
arc_easy
gpqa.py
gsm8k.py
mathematics.py
mmlu.py
honeycomb
queries.py
critique
validate
langchain
wikipedia.py
wikipedia
wikipedia.py
langchain > wikipedia.py > ...
13 from langchain_core.language_models import BaseChatModel
14
15 from inspect_ai import Task, task
16 from inspect_ai.dataset import json_dataset
17 from inspect_ai.scorer import model_graded_fact
18 from inspect_ai.solver import Solver, solver
19
Debug Task | Run Task
20 @task
21 def wikipedia() -> Task:
22 return Task(
23 dataset=json_dataset("wikipedia.jsonl"),
24 plan=wikipedia_search(),
25 scorer=model_graded_fact(),
26)
27
28 @solver
29 def wikipedia_search(
30 max_iterations: int | None = 15,
31 max_execution_time: float | None = None
32) -> Solver:
33 # standard prompt for functions agent
34 prompt = hub.pull("hwchase17/openai-tools-agent")
35
36 # tavily and wikipedia tools
37 tavily_api = TavilySearchAPIWrapper() # type: ignore
38 tools = [TavilySearchResults(api_wrapper=tavily_api)]
Inspect View
Inspect View
2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
Input
Target
Answer
Score
1
List the ten episode titles from the sixth season of "Game of Thrones" in broadcast order.
The Red Woman, Home,...
From the summary, I can see...
C
Messages
Scoring
Metadata
system
You are a helpful assistant
user
List the ten episode titles from the sixth season of "Game of Thrones" in broadcast order.
assistant
Okay, to find the episode titles for the sixth season of "Game of Thrones", I will use the Wikipedia tool, as it should have comprehensive information on the show.
wikipedia(query: Game of Thrones season 6)
Iwan Rheon, Michael McElhatton, Kristian Nairn, Natalie Dormer, Jonathan Pryce, and Finn Jones.
Critics praised its production values, writing, plot development, and cast. Game of Thrones received the most nominations for the 68th Primetime Emmy Awards, with 23 nominations, and won 12, including that for Outstanding Drama Series for the second year in a row. U.S. viewership rose compared to the previous season, and by approximately 13 percent over its course, from 7.9
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % cd /Users/jjallaire/ukinstitute/inspect-llm-workshop
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval langchain/wikipedia.py@wikipedia
wikipedia (3 samples)
total time: 0:00:41
anthropic/claude-3-sonnet-20240229 11,514 tokens [9,400 + 2,114]
anthropic/claude-3-sonnet-20240229
dataset: wikipedia
scorer: model_graded_fact
accuracy: 0.667 bootstrap_std: 0.268
Log: ./logs/2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
zsh
Inspect ...
Quarto P...
main*
0 0
0
Quarto: 1.4.553


inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
anthropic
claude-3-sonnet-20240229
Connections
Retries
Timeout
default
default
default
TASK wikipedia
Options
Task Args
Limit
Epochs
default
default
Max Tokens
Temperature
default
default
Top P
Top K
default
default
TASKS
benchmarks
arc.py
arc_easy
gpqa.py
gsm8k.py
mathematics.py
mmlu.py
honeycomb
queries.py
critique
validate
langchain
wikipedia.py
wikipedia
wikipedia.py
langchain > wikipedia.py > ...
13 from langchain_core.language_models import BaseChatModel
14
15 from inspect_ai import Task, task
16 from inspect_ai.dataset import json_dataset
17 from inspect_ai.scorer import model_graded_fact
18 from inspect_ai.solver import Solver, solver
19
Debug Task | Run Task
20 @task
21 def wikipedia() -> Task:
22 return Task(
23 dataset=json_dataset("wikipedia.jsonl"),
24 plan=wikipedia_search(),
25 scorer=model_graded_fact(),
26)
27
28 @solver
29 def wikipedia_search(
30 max_iterations: int | None = 15,
31 max_execution_time: float | None = None
32) -> Solver:
33 # standard prompt for functions agent
34 prompt = hub.pull("hwchase17/openai-tools-agent")
35
36 # tavily and wikipedia tools
37 tavily_api = TavilySearchAPIWrapper() # type: ignore
38 tools = [TavilySearchResults(api_wrapper=tavily_api)]
Inspect View
Inspect View
2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
3
Which types of fish contain the lowest levels of mercury?
The following types of...
Based on the informati...
I
Messages
Scoring
Metadata
Input
Which types of fish contain the lowest levels of mercury?
Target
The following types of fish contain low levels of mercury: salmon, flounder, Atlantic mackerel, anchovies, pollock, catfish, and shellfish (e.g., clams, scallops, mussels).
Answer
Based on the information from Wikipedia, some key points about mercury levels in fish:
Fish higher up the food chain like tuna, swordfish, shark tend to have higher mercury levels due to bioaccumulation
Long-lived, predatory fish concentrate more mercury in their bodies over time
Small fish that feed on plankton and are lower on the food chain tend to have lower mercury levels
Therefore, to minimize mercury exposure from fish, it is recommended to eat smaller fish species that are
Score
I
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % cd /Users/jjallaire/ukinstitute/inspect-llm-workshop
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval langchain/wikipedia.py@wikipedia
wikipedia (3 samples)
total time: 0:00:41
anthropic/claude-3-sonnet-20240229 11,514 tokens [9,400 + 2,114]
anthropic/claude-3-sonnet-20240229
dataset: wikipedia
scorer: model_graded_fact
accuracy: 0.667 bootstrap_std: 0.268
Log: ./logs/2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
zsh
Inspect ...
Quarto P...
main*
0 0
0
Quarto: 1.4.553


inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
anthropic
claude-3-sonnet-20240229
Connections
Retries
Timeout
default
default
default
TASK wikipedia
Options
Task Args
Limit
Epochs
default
default
Max Tokens
Temperature
default
default
Top P
Top K
default
default
TASKS
benchmarks
arc.py
arc_easy
gpqa.py
gsm8k.py
mathematics.py
mmlu.py
honeycomb
queries.py
critique
validate
langchain
wikipedia.py
wikipedia
wikipedia.py
langchain > wikipedia.py > ...
13 from langchain_core.language_models import BaseChatModel
14
15 from inspect_ai import Task, task
16 from inspect_ai.dataset import json_dataset
17 from inspect_ai.scorer import model_graded_fact
18 from inspect_ai.solver import Solver, solver
19
Debug Task | Run Task
20 @task
21 def wikipedia() -> Task:
22 return Task(
23 dataset=json_dataset("wikipedia.jsonl"),
24 plan=wikipedia_search(),
25 scorer=model_graded_fact(),
26)
27
28 @solver
29 def wikipedia_search(
30 max_iterations: int | None = 15,
31 max_execution_time: float | None = None
32) -> Solver:
33 # standard prompt for functions agent
34 prompt = hub.pull("hwchase17/openai-tools-agent")
35
36 # tavily and wikipedia tools
37 tavily_api = TavilySearchAPIWrapper() # type: ignore
38 tools = [TavilySearchResults(api_wrapper=tavily_api)]
Inspect View
Inspect View
2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
Explanation
To assess the submitted answer against the expert answer, I will follow these steps:
1. List out the types of fish mentioned as low-mercury in the expert answer.
2. Check if the submitted answer includes all those fish types.
3. Check if the submitted answer includes any additional low-mercury fish types not present in the expert answer.
4. Based on the above points, determine if the submitted answer fully contains the content from the expert answer.
Step 1: Types of low-mercury fish mentioned in the expert answer:
salmon
flounder
Atlantic mackerel
anchovies
pollock
catfish
shellfish (clams, scallops, mussels)
Step 2: Checking if the submitted answer includes all those fish types:
salmon: Yes
flounder: No
Atlantic mackerel: No
anchovies: Yes
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % cd /Users/jjallaire/ukinstitute/inspect-llm-workshop
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval langchain/wikipedia.py@wikipedia
wikipedia (3 samples)
total time: 0:00:41
anthropic/claude-3-sonnet-20240229 11,514 tokens [9,400 + 2,114]
anthropic/claude-3-sonnet-20240229
dataset: wikipedia
scorer: model_graded_fact
accuracy: 0.667 bootstrap_std: 0.268
Log: ./logs/2024-05-27T10-39-14_wikipedia_gWQ6UqJRde64cUhCdKufgr.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
zsh
Inspect ...
Quarto P...
main*
0 0
0
Quarto: 1.4.553
Ln 27, Col 1
Spaces: 4
UTF-8
LF
Python
3.11.6 ('.venv': venv)
Prettier

Scoring

- Conventional pattern matching / templated answer based scoring built in.
- Model graded scorer built-in (often heavily customized)
- Scorers also pluggable (i.e. provided from other packages). We expect lots of innovation in model graded scoring!
- Offline / human scoring workflow is supported.
- Plan to build tools to help rigerously evaluate model graded scorers against human baselines.


inspect-llm-workshop
INSPECT
CONFIGURATION (.ENV)
Model
Logging
Model
openai
gpt-4-turbo
Connections
Retries
Timeout
default
default
default
TASK math
mathematics.py
benchmarks > mathematics.py > ...
Debug Task | Run Task
@task
def math(shuffle=True):
return Task(
dataset=csv_dataset(
csv_file="datasets/math_test.csv",
sample_fields=FieldSpec(input="Question",
target="Answer"),
shuffle=shuffle,
),
plan=[
prompt_template(PROMPT_TEMPLATE),
generate(),
],
scorer=expression_equivalance(),
config=GenerateConfig(temperature=0.5),
)
@scorer(metrics=[accuracy(), bootstrap_std()])
def expression_equivalance():
async def score(state: TaskState, target: Target):
# extract answer
match = re.search(AnswerPattern.LINE, state.
output.completion)
if match:
# ask the model to judge equivalance
Inspect View
Inspect View
2024-05-27T10-52-10_math_4RW2MmctfNi9bBoPBjvKQo.json
math openai/gpt-4-turbo
5/27/2024, 10:52:10 AM— 27 sec
accuracy
0.8
bootstrap_std
0.123
DATASET
math_test — 10 samples
PLAN
prompt_template → generate
SCORER
expression_equivalance
HYPERPARAMETERS
temperature: 0.5
Samples
Info
Logging
JSON
Scores:
All
Sort:
sample asc
Open All
Input
Target
Answer
Score
1
A circle of radius 2 is inscribed in a semicircle, as shown. The area inside the semicircle but outside...
\frac{1}{2}
2
Let f be a function taking the positive integers to the positive integers, such that...
18
3
Mary has 6 identical basil plants, and three different window sills she can put them on. How...
28
4
Compute: $(3^2)(2^4)(37)(5^3)$
666000
5
There are 30 men and 40 women in the Town Library Club. They wish to form a 7-person steering...
371,\!043,\!400
Options
Task Args
Limit
10
Epochs
default
Max Tokens
default
Temperature
default
Top P
default
Top K
default
TASKS
benchmarks
arc.py
arc_easy
gpqa.py
gsm8k.py
mathematics.py
math
mmlu.py
honeycomb
queries.py
critique
validate
langchain
wikipedia.py
\(\frac{1}{2}\)
2007
28
666000
371043400
zsh
Inspect ...
Quarto P...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop % inspect eval benchmarks/mathematics.py@math --limit 10
math (10 samples)
temperature: 0.5
total time: 0:00:27
openai/gpt-4-turbo 8,775 tokens [4,723 + 4,052]
accuracy: 0.8 bootstrap_std: 0.123
Log: ./logs/2024-05-27T10-52-10_math_4RW2MmctfNi9bBoPBjvKQo.json
(.venv) jjallaire@JJs-MacBook-Pro inspect-llm-workshop %
main*
Quarto: 1.4.553
Ln 53, Col 1
Spaces: 2
UTF-8
LF
Python
3.11.6

Scorer: expression_equivalence()

```python
@scorer(metrics=[accuracy(), bootstrap_std()])
def expression_equivalance():
    async def score(state: TaskState, target: Target):

        # extract answer
        match = re.search(AnswerPattern.LINE, state.output.completion)

        # ask the model to judge equivalance
        answer = match.group(1)
        prompt = EQUIVALANCE_TEMPLATE % (
            {"expression1": target.text, "expression2": answer}
        )
        result = await get_model().generate(prompt)

        # return the score
        correct = result.completion.lower() == "yes"
        return Score(
            value=CORRECT if correct else INCORRECT,
            answer=answer,
            explanation=state.output.completion,
        )

    return score
```

Logging

- Capture all context required to debug, analyse, and reproduce evaluations
- Python API for computing on log file contents
- Log viewer for interactive exploration of eval results

EvalLog

`status`	`str`	Status of evaluation
`eval`	`EvalSpec`	Top level eval details including task, model, creation time, etc.
`plan`	`EvalPlan`	List of solvers and model generation config used for the eval.
`samples`	`list[EvalSample]`	Each sample evaluated, including its input, output, target, and score.
`results`	`EvalResults`	Aggregated scorer results
`stats`	`EvalStats`	Model token usage stats
`logging`	`list[LoggingMessage]`	Logging messages (e.g. from `log.info()`, `log.debug()`, etc.
`error`	`EvalError`	Error information

Log Viewer: Samples

gpqa_diamond openai/gpt-4

4/28/2024, 10:18:00 AM— 10 min 59 sec

accuracy	bootstrap_std
0.335	0.017

DATASET
gpqa_diamond — 198 x 4 samples

PLAN
multiple_choice

SCORER
answer

HYPERPARAMETERS
`temperature: 0.5`

Samples | Info | Logging | JSON

Epochs: 2 | Scores: All | Sort: epoch asc | Open All

	Input	Target	Answer	Score
EPOCH 2				
1	Two quantum states with energies E1 and E2 have a lifetime of 10^-9 sec and 10^-8 sec, respectively. We want to clearly distinguish these two energy levels. Which one ...	A	A	C
2	trans-cinnamaldehyde was treated with methylmagnesium bromide, forming product 1. 1 was treated with pyridinium chlorochromate, forming...	A	The question seems to contain a mistake. Product 3 is formed...	I
3	A spin-half particle is in a linear superposition 0.5|\uparrow\rangle+sqrt(3)/2|\downarrow\rangle of its spin-up and spin-down states. If |\uparrow\rangle and...	A	The expectation value of an operator O in a state |ψ⟩ is given by...	I
4	In a parallel universe where a magnet can have an isolated North or South pole, Maxwell's equations look different. But, specifically, which of those equations are different?	A	C	I
5	Calculate the eigenvector of a quantum mechanical operator $\vec{P}$ for a muon along an arbitrary direction $\vec{n}$ lying in the x-z plane corresponding to the...	A	A	C

Log Viewer: Messages

1	What's the difference between tennis and pickleball?	While they are similar sports, tennis and pickleb...	Tennis and pickleball, while similar in some...	C

Messages | Scoring | Metadata

system
You are a helpful assistant

user
What's the difference between tennis and pickleball?

assistant
`wikipedia(__arg1: Pickleball)`

```
Page: Pickleball
Summary: Pickleball is a racket or paddle sport in which two players (singles) or four players
(doubles) hit a perforated, hollow plastic ball with paddles over a 34-inch-high (0.86 m) net
until one side is unable to return the ball or commits a rule infraction. Pickleball is played
indoors and outdoors. It was invented in 1965 as a children's backyard game in the United
States, on Bainbridge Island in Washington state. In 2022, pickleball was named the official
state sport of Washington.
```

assistant
`wikipedia(__arg1: Tennis)`

```
Page: Tennis
Summary: Tennis is a racket sport that is played either individually against a single opponent
(singles) or between two teams of two players each (doubles). Each player uses a tennis racket
that is strung with cord to strike a hollow rubber ball covered with felt over or around a net
and into the opponent's court. The object of the game is to manoeuvre the ball in such a way
that the opponent is not able to play a valid return. The player who is unable to return the
ball validly will not gain a point, while the opposite player will.
```

assistant
Tennis and pickleball, while similar in some ways, have several fundamental differences.

In tennis:

- It's a racket sport that can be played individually against a single opponent (singles) or between two teams of two players each (doubles).

Log Viewer: Scoring

3	The universe is filled with the Cosmic Microwave Background. Consider the annihilation of high energy \gamma-rays with a photon from the CMB Radiation into...	A	A	C

Messages | Scoring

Input

The universe is filled with the Cosmic Microwave Background. Consider the annihilation of high energy \gamma-rays with a photon from the CMB Radiation into electron-positron, i.e. $\gamma\gamma\rightarrow e^{+}e^{-}$. From what energy \gamma-rays would have their lifetimes in the universe limited by this process? Knowing that the average photon energy of the CMB is $10^{-3}eV$.

A) 2.6*1e5 GeV
B) 1.8*1e5 GeV
C) 3.9*1e5 GeV
D) 9.5*1e4 GeV

Target	Answer	Score
A	A	C

Explanation

The process of photon-photon pair production (or annihilation) can only occur if the energy of the incoming gamma-ray photon is above a certain threshold. This threshold is given by the rest mass energy of the electron-positron pair, which is about 1 MeV or 1e6 eV. However, the energy of the incoming photon must also be significantly higher than the energy of the CMB photon it is interacting with.

The energy of the CMB photon is given as 10^-3 eV. The energy of the gamma-ray photon must be such that, in the rest frame of the CMB photon, the gamma-ray photon has an energy greater than 1 MeV. This requires a gamma-ray energy of approximately *210^11 eV or 2*10^5 GeV in the lab frame.

Looking at the answer choices, the closest to this value is 2.6*10^5 GeV.

ANSWER: A

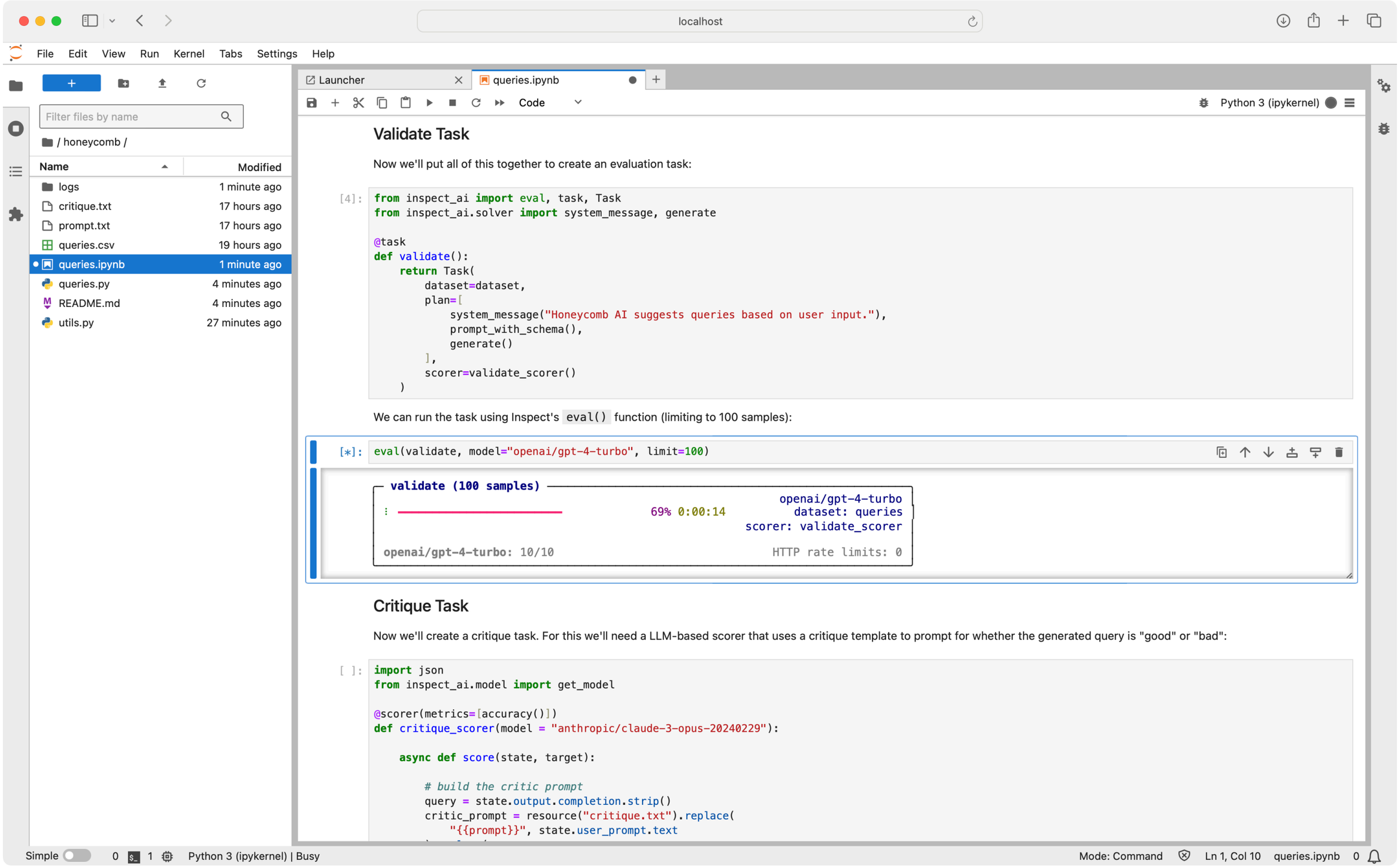
Models

Provider	Model Name	Docs
OpenAI	`openai/gpt-3.5-turbo`	OpenAI Models
Anthropic	`anthropic/claude-3-sonnet-20240229`	Anthropic Models
Google	`google/gemini-1.0-pro`	Google Models
Mistral	`mistral/mistral-large-latest`	Mistral Models
Hugging Face	`hf/openai-community/gpt2`	Hugging Face Models
Ollama	`ollama/llama3`	Ollama Models
TogetherAI	`together/lmsys/vicuna-13b-v1.5`	TogetherAI Models
AWS Bedrock	`bedrock/meta.llama2-70b-chat-v1`	AWS Bedrock Models
Azure AI	`azureai/azure-deployment-name`	Azure AI Models
Cloudflare	`cf/meta/llama-2-7b-chat-fp16`	Cloudflare Models

Interface with any other model by creating a custom model provider…

Workflow

- Lots of interactive exploration occurs during eval development, so critical to have good support for iterating in a Notebook / REPL
- Eventually though, evals need to end up in a form that enables reproducibly running them in an eval suite
- Need to support a continuum of workflows that transition well into each other
- Provide good tooling in Jupyter and VS Code for entire spectrum


localhost
File Edit View Run Kernel Tabs Settings Help
Launcher
queries.ipynb
Code
Python 3 (ipykernel)
Filter files by name
/ honeycomb /
Name
Modified
logs 1 minute ago
critique.txt 17 hours ago
prompt.txt 17 hours ago
queries.csv 19 hours ago
queries.ipynb 1 minute ago
queries.py 4 minutes ago
README.md 4 minutes ago
utils.py 27 minutes ago
Validate Task
Now we'll put all of this together to create an evaluation task:
[4]:
from inspect_ai import eval, task, Task
from inspect_ai.solver import system_message, generate
@task
def validate():
return Task(
dataset=dataset,
plan=[
system_message("Honeycomb AI suggests queries based on user input."),
prompt_with_schema(),
generate()
],
scorer=validate_scorer()
)
We can run the task using Inspect's eval() function (limiting to 100 samples):
[*]:
eval(validate, model="openai/gpt-4-turbo", limit=100)
validate (100 samples)
openai/gpt-4-turbo
69% 0:00:14
dataset: queries
scorer: validate_scorer
openai/gpt-4-turbo: 10/10
HTTP rate limits: 0
Critique Task
Now we'll create a critique task. For this we'll need a LLM-based scorer that uses a critique template to prompt for whether the generated query is "good" or "bad":
[ ]:
import json
from inspect_ai.model import get_model
@scorer(metrics=[accuracy()])
def critique_scorer(model = "anthropic/claude-3-opus-20240229"):
async def score(state, target):
# build the critic prompt
query = state.output.completion.strip()
critic_prompt = resource("critique.txt").replace(
"{{prompt}}", state.user_prompt.text
Simple
0 1
Python 3 (ipykernel) | Busy
Mode: Command
Ln 1, Col 10
queries.ipynb
0

Interactive Exploration

Ad-hoc exploration of an eval in a Notebook/REPL

```
params = {
    "system": ["devops.txt", "researcher.txt"],
    "grader": ["hacker.txt", "expert.txt"],
    "grader_model": ["openai/gpt-4", "google/gemini-1.0-pro"]
}
params = list(product(*(params[name] for name in params)))

tasks = [Task(
    dataset=json_dataset("security_guide.jsonl"),
    plan=[system_message(system), generate()],
    scorer=model_graded_fact(template=grader, model=grader_model)
) for system, grader, grader_model in params]

logs = eval(tasks, model = "mistral/mistral-large-latest")
plot_results(logs)
```

Task Parameters

Formalise variation with a parameterised `@task` function:

```
@task
def security_guide(system="devops.txt", grader="expert.txt"):
   return Task(
      dataset = json_dataset("security_guide.jsonl"),
      plan=[system_message(system), generate()],
      scorer=model_graded_fact(template=grader, model="openai/gpt-4")
   )

params = {
   "system": ["devops.txt", "researcher.txt"],
   "grader": ["hacker.txt", "expert.txt"]
}
params = list(product(*(params[name] for name in params)))

eval([security_guide(system,grader) for system, grader in params],
     model = "mistral/mistral-large-latest")
```

Task Parameters

`@task` functions are registered and addressable by external driver programs (step one in development => production)

```python
@task
def security_guide(system="devops.txt", grader="expert.txt"):
    return Task(
        dataset = json_dataset("security_guide.jsonl"),
        plan=[system_message(system), generate()],
        scorer=model_graded_fact(
            template=grader,
            model="openai/gpt-4"
        )
    )
```

Now we can vary the parameters externally:

```
$ inspect eval security_guide.py -T system=devops.txt
$ inspect eval security_guide.py -T grader=hacker.txt
```

Same workflow available for tasks in a notebook:

```
$ inspect eval security_guide.ipynb -T system=devops.txt
$ inspect eval security_guide.ipynb -T grader=hacker.txt
```

Task Variants

We may discover that we *always* want to vary a parameter when running a full evaluation suite:

```
def security_guide(system, grader="expert.txt"):
   return Task(
      dataset = json_dataset("security_guide.jsonl"),
      plan=[system_message(system), generate()],
      scorer=model_graded_fact(template=grader, model="openai/gpt-4")
   )

@task
def devops()
   return security_guide("devops.txt")

@task
def researcher()
   return security_guide("researcher.txt")
```

Invoke by task name

```
$ inspect eval security_guide.py@devops
$ inspect eval security_guide.py@researcher
```

Eval Suites

We want to allow for arbitrary source code organisation but still be able to discover and enumerate tasks for a suite

```
security/
  jeopardy/
    import.py
    analyze.py
    task.py
  attack_defense/
    import.py
    analyze.py
    task.py
```

```
list_tasks("security")

jeopardy/task.py@crypto
jeopardy/task.py@decompile
jeopardy/task.py@packet
jeopardy/task.py@heap_trouble
attack_defense/task.py@saar
attack_defense/task.py@bank
attack_defense/task.py@voting
```

Run them all

```
eval(list_tasks("security"), model="mistral/mistral-large-latest")
```

Resiliency

The production version would look more like this:

```
# setup log context
os.environ["INSPECT_LOG_DIR"] = "./security-suite_04-07-2024"

# run the eval suite
tasks = list_tasks("security")
eval(tasks, model="mistral/mistral-large-latest")

# ...later, in another process that also has INSPECT_LOG_DIR
error_logs = list_eval_logs(status == "error")
eval_retry(error_logs)
```

Somewhat oversimplified, as we'd also want to enhance the logic around analysing the cause of errors and adopting optimal recovery strategies

Provenance

If you run an eval from a Git repository, you should be able to reproduce the eval with only its log file as context

```
# read the log and extract the origin and commit
log = read_eval_log("security-log.json")
origin = log.spec.revision.origin
commit = log.spec.revision.commit

# clone the repo, checkout the commit, install deps, and run
run(["git", "clone", revision.origin, "eval-dir"])
with chdir("eval-dir"):
    run(["git", "checkout", revision.commit])
    run(["pip", "install", "-r", "requirements.txt"])
    eval(log)
```

Learning More

- Docs: https://ukgovernmentbeis.github.io/inspect_ai
- GitHub: https://github.com/ukgovernmentbeis/inspect_ai
- Slides/Code: https://github.com/jjallaire/inspect-llm-workshop

Questions?