LangSmith Walkthrough

Open in Colab

Open In Colab

LangChain makes it easy to prototype LLM applications and Agents. However, delivering LLM applications to production can be deceptively difficult. You will have to iterate on your prompts, chains, and other components to build a high-quality product.

LangSmith makes it easy to debug, test, and continuously improve your LLM applications.

When might this come in handy? You may find it useful when you want to:

- Quickly debug a new chain, agent, or set of tools
- Create and manage datasets for fine-tuning, few-shot prompting, and evaluation
- Run regression tests on your application to confidently develop

- Capture production analytics for product insights and continuous improvements

Prerequisites

Create a LangSmith account and create an API key (see bottom left corner). Familiarize yourself with the platform by looking through the docs

Note LangSmith is in closed beta; we're in the process of rolling it out to more users. However, you can fill out the form on the website for expedited access.

Now, let's get started!

Log runs to LangSmith

First, configure your environment variables to tell LangChain to log traces. This is done by setting the `LANGCHAIN_TRACING_V2` environment variable to true. You can tell LangChain which project to log to by setting the `LANGCHAIN_PROJECT` environment variable (if this isn't set, runs will be logged to the `default` project). This will automatically create the project for you if it doesn't exist. You must also set the `LANGCHAIN_ENDPOINT` and `LANGCHAIN_API_KEY` environment variables.

For more information on other ways to set up tracing, please reference the LangSmith documentation.

NOTE: You can also use a context manager in python to log traces using

```
from langchain_core.tracers.context import tracing_v2_enabled

with tracing_v2_enabled(project_name="My Project"):
    agent.run("How many people live in canada as of 2023?")
```

However, in this example, we will use environment variables.

```
%pip install --upgrade --quiet  langchain langsmith langchainhub --quiet
%pip install --upgrade --quiet  langchain-openai tiktoken pandas duckduckgo-search --quiet
```

```
import os
from uuid import uuid4
```

```
unique_id = uuid4().hex[0:8]
os.environ["LANGCHAIN_TRACING_V2"] = "true"
os.environ["LANGCHAIN_PROJECT"] = f"Tracing
Walkthrough - {unique_id}"
os.environ["LANGCHAIN_ENDPOINT"] =
"https://api.smith.langchain.com"
os.environ["LANGCHAIN_API_KEY"] = "<YOUR-API-
KEY>"  # Update to your API key

# Used by the agent in this tutorial
os.environ["OPENAI_API_KEY"] = "<YOUR-OPENAI-
API-KEY>"
```

Create the langsmith client to interact with the API

```
from langsmith import Client

client = Client()
```

Create a LangChain component and log runs to the platform. In this example, we will create a ReAct-style agent with access to a general search tool (DuckDuckGo). The agent's prompt can be viewed in the Hub here.

```
from langchain import hub
from langchain.agents import AgentExecutor
from langchain.agents.format_scratchpad
```

```
import format_to_openai_function_messages
from langchain.agents.output_parsers import
OpenAIFunctionsAgentOutputParser
from langchain_community.tools import
DuckDuckGoSearchResults
from langchain_openai import ChatOpenAI

# Fetches the latest version of this prompt
prompt = hub.pull("wfh/langsmith-agent-
prompt:5d466cbc")

llm = ChatOpenAI(
    model="gpt-3.5-turbo-16k",
    temperature=0,
)

tools = [
    DuckDuckGoSearchResults(
        name="duck_duck_go"
    ),  # General internet search using
DuckDuckGo
]

llm_with_tools = llm.bind_functions(tools)

runnable_agent = (
    {
        "input": lambda x: x["input"],
        "agent_scratchpad": lambda x:
format_to_openai_function_messages(
            x["intermediate_steps"]
        ),
    }
    | prompt
    | llm_with_tools
    | OpenAIFunctionsAgentOutputParser()
)

agent_executor = AgentExecutor(
    agent=runnable_agent, tools=tools,
handle_parsing_errors=True
)
```

We are running the agent concurrently on multiple inputs to reduce latency. Runs get logged to LangSmith in the background so execution latency is unaffected.

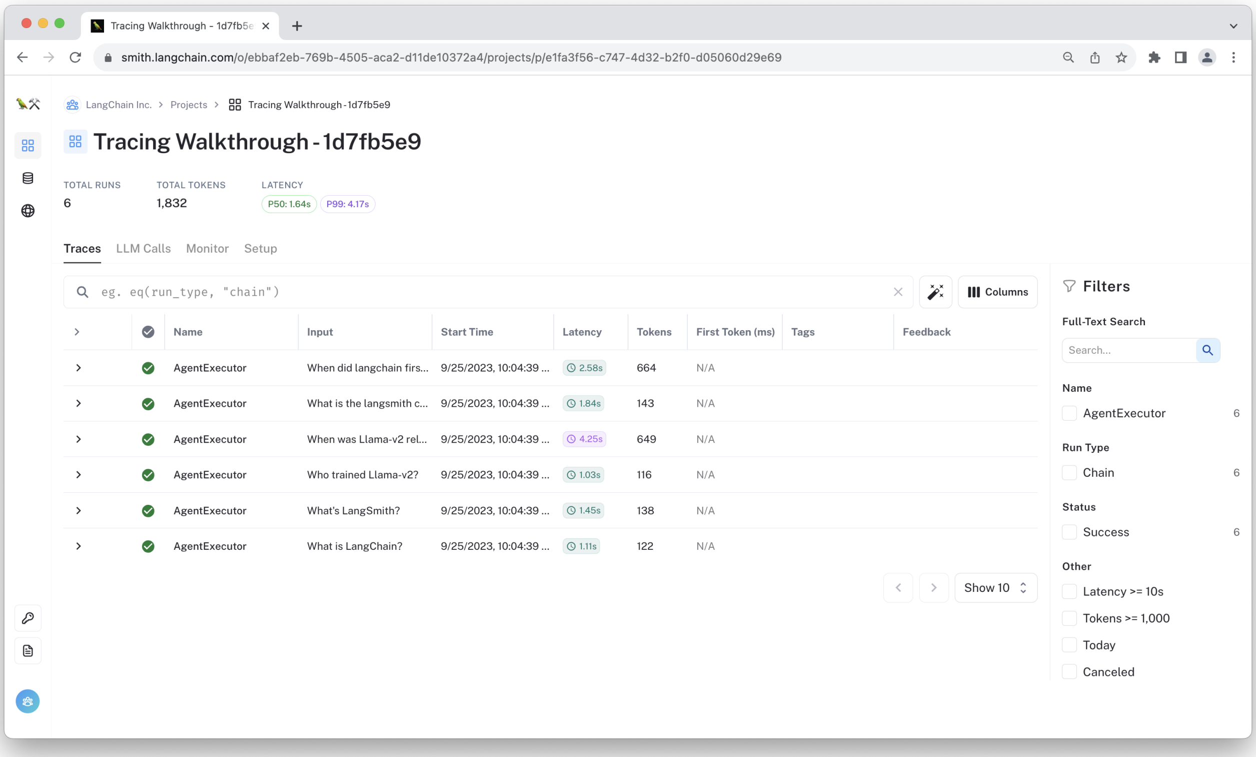
```
inputs = [
    "What is LangChain?",
    "What's LangSmith?",
    "When was Llama-v2 released?",
    "What is the langsmith cookbook?",
    "When did langchain first announce the
hub?",
]

results = agent_executor.batch([{"input": x}
for x in inputs], return_exceptions=True)
```

```
results[:2]
```

```
[{'input': 'What is LangChain?',
  'output': 'I\'m sorry, but I couldn\'t find any information about "LangChain". Could you please provide more context or clarify your question?'},
 {'input': "What's LangSmith?",
  'output': 'I\'m sorry, but I couldn\'t find any information about "LangSmith". It could be a company, a product, or a person. Can you provide more context or details about what you are referring to?'}]
```

Assuming you've successfully set up your environment, your agent traces should show up in the `Projects` section in the app. Congrats!


Tracing Walkthrough - 1d7fb5e
smith.langchain.com/o/ebbaf2eb-769b-4505-aca2-d11de10372a4/projects/p/e1fa3f56-c747-4d32-b2f0-d05060d29e69
LangChain Inc. > Projects > Tracing Walkthrough - 1d7fb5e9
Traces
LLM Calls
Monitor
Setup
eg. eq(run_type, "chain")
Columns
Show 10

Tracing Walkthrough - 1d7fb5e9

TOTAL RUNS	TOTAL TOKENS	LATENCY
6	1,832	P50: 1.64s P99: 4.17s

Name	Input	Start Time	Latency	Tokens	First Token (ms)	Tags	Feedback
AgentExecutor	When did langchain firs...	9/25/2023, 10:04:39 ...	2.58s	664	N/A		
AgentExecutor	What is the langsmith c...	9/25/2023, 10:04:39 ...	1.84s	143	N/A		
AgentExecutor	When was Llama-v2 rel...	9/25/2023, 10:04:39 ...	4.25s	649	N/A		
AgentExecutor	Who trained Llama-v2?	9/25/2023, 10:04:39 ...	1.03s	116	N/A		
AgentExecutor	What's LangSmith?	9/25/2023, 10:04:39 ...	1.45s	138	N/A		
AgentExecutor	What is LangChain?	9/25/2023, 10:04:39 ...	1.11s	122	N/A		

Filters

Full-Text Search

Name
- AgentExecutor 6

Run Type
- Chain 6

Status
- Success 6

Other
- Latency >= 10s
- Tokens >= 1,000
- Today
- Canceled

It looks like the agent isn't effectively using the tools though. Let's evaluate this so we have a baseline.

Evaluate Agent

In addition to logging runs, LangSmith also allows you to test and evaluate your LLM applications.

In this section, you will leverage LangSmith to create a benchmark dataset and run AI-assisted evaluators on an agent. You will do so in a few steps:

1. Create a dataset
2. Initialize a new agent to benchmark

3. Configure evaluators to grade an agent's output
4. Run the agent over the dataset and evaluate the results

1. Create a LangSmith dataset

Below, we use the LangSmith client to create a dataset from the input questions from above and a list labels. You will use these later to measure performance for a new agent. A dataset is a collection of examples, which are nothing more than input-output pairs you can use as test cases to your application.

For more information on datasets, including how to create them from CSVs or other files or how to create them in the platform, please refer to the LangSmith documentation.

```
outputs = [
    "LangChain is an open-source framework
for building applications using large
language models. It is also the name of the
company building LangSmith.",
    "LangSmith is a unified platform for
debugging, testing, and monitoring language
model applications and agents powered by
LangChain",
    "July 18, 2023",
    "The langsmith cookbook is a github
repository containing detailed examples of
how to use LangSmith to debug, evaluate, and
monitor large language model-powered
applications.",
    "September 5, 2023",
]
```

```
dataset_name = f"agent-qa-{unique_id}"

dataset = client.create_dataset(
    dataset_name,
    description="An example dataset of
questions over the LangSmith documentation.",
)

client.create_examples(
    inputs=[{"input": query} for query in
inputs],
    outputs=[{"output": answer} for answer in
outputs],
    dataset_id=dataset.id,
)
```

2. Initialize a new agent to benchmark

LangSmith lets you evaluate any LLM, chain, agent, or even a custom function. Conversational agents are stateful (they have memory); to ensure that this state isn't shared between dataset runs, we will pass in a `chain_factory` (aka a `constructor`) function to initialize for each call.

In this case, we will test an agent that uses OpenAI's function calling endpoints.

```
from langchain import hub
from langchain.agents import AgentExecutor,
AgentType, initialize_agent, load_tools
from langchain.agents.format_scratchpad
import format_to_openai_function_messages
from langchain.agents.output_parsers import
OpenAIFunctionsAgentOutputParser
from langchain_openai import ChatOpenAI


# Since chains can be stateful (e.g. they can
have memory), we provide
# a way to initialize a new chain for each
row in the dataset. This is done
# by passing in a factory function that
returns a new chain for each row.
def create_agent(prompt, llm_with_tools):
    runnable_agent = (
        {
            "input": lambda x: x["input"],
            "agent_scratchpad": lambda x:
format_to_openai_function_messages(
                x["intermediate_steps"]
            ),
        }
        | prompt
        | llm_with_tools
```

```
            | OpenAIFunctionsAgentOutputParser()
    )
    return
AgentExecutor(agent=runnable_agent,
tools=tools, handle_parsing_errors=True)
```

3. Configure evaluation

Manually comparing the results of chains in the UI is effective, but it can be time consuming. It can be helpful to use automated metrics and AI-assisted feedback to evaluate your component's performance.

Below, we will create a custom run evaluator that logs a heuristic evaluation.

Heuristic evaluators

```
from langsmith.evaluation import
EvaluationResult, run_evaluator
from langsmith.schemas import Example, Run


@run_evaluator
def check_not_idk(run: Run, example:
Example):
    """Illustration of a custom evaluator."""
    agent_response = run.outputs["output"]
    if "don't know" in agent_response or "not
```

```
sure" in agent_response:
            score = 0
        else:
            score = 1
        # You can access the dataset labels in
example.outputs[key]
        # You can also access the model inputs in
run.inputs[key]
        return EvaluationResult(
            key="not_uncertain",
            score=score,
        )
```

Below, we will configure the evaluation with the custom evaluator from above, as well as some pre-implemented run evaluators that do the following: - Compare results against ground truth labels. - Measure semantic (dis)similarity using embedding distance - Evaluate 'aspects' of the agent's response in a reference-free manner using custom criteria

For a longer discussion of how to select an appropriate evaluator for your use case and how to create your own custom evaluators, please refer to the LangSmith documentation.

```
from langchain.evaluation import EvaluatorType
from langchain.smith import RunEvalConfig

evaluation_config = RunEvalConfig(
    # Evaluators can either be an evaluator typ
"qa", "criteria", "embedding_distance", etc.) o
configuration for that evaluator
    evaluators=[
        # Measures whether a QA response is "Co
based on a reference answer
        # You can also select via the raw strin
        EvaluatorType.QA,
        # Measure the embedding distance betwee
and the reference answer
        # Equivalent to:
EvalConfig.EmbeddingDistance(embeddings=OpenAIE
        EvaluatorType.EMBEDDING_DISTANCE,
        # Grade whether the output satisfies th
criteria.
        # You can select a default one such as
"helpfulness" or provide your own.
        RunEvalConfig.LabeledCriteria("helpfuln
        # The LabeledScoreString evaluator outp
on a scale from 1-10.
        # You can use default criteria or write
rubric
        RunEvalConfig.LabeledScoreString(
            {
                "accuracy": """
Score 1: The answer is completely unrelated to
reference.
Score 3: The answer has minor relevance but doe
with the reference.
Score 5: The answer has moderate relevance but 
inaccuracies.
Score 7: The answer aligns with the reference b
errors or omissions.
Score 10: The answer is completely accurate and
perfectly with the reference."""
            },
            normalize_by=10,
        ),
    ],
    # You can add custom StringEvaluator or Run
objects here as well, which will automatically
    # applied to each prediction. Check out the
examples.
    custom_evaluators=[check_not_idk],
)
```

4. Run the agent and evaluators

Use the run_on_dataset (or asynchronous arun_on_dataset) function to evaluate your model. This will: 1. Fetch example rows from the specified dataset. 2. Run your agent (or any custom function) on each example. 3. Apply evaluators to the resulting run traces and corresponding reference examples to generate automated feedback.

The results will be visible in the LangSmith app.

```
from langchain import hub
```

```
# We will test this version of the prompt
prompt = hub.pull("wfh/langsmith-agent-
prompt:798e7324")
```

```
import functools

from langchain.smith import arun_on_dataset,
run_on_dataset

chain_results = run_on_dataset(
    dataset_name=dataset_name,
    llm_or_chain_factory=functools.partial(
        create_agent, prompt=prompt,
llm_with_tools=llm_with_tools
    ),
    evaluation=evaluation_config,
    verbose=True,
    client=client,
    project_name=f"runnable-agent-test-
5d466cbc-{unique_id}",
    # Project metadata communicates the
experiment parameters,
    # Useful for reviewing the test results
    project_metadata={
        "env": "testing-notebook",
        "model": "gpt-3.5-turbo",
        "prompt": "5d466cbc",
    },
)
```

```
# Sometimes, the agent will error due to
parsing issues, incompatible tool inputs,
etc.
# These are logged as warnings here and
captured as errors in the tracing UI.
```

```
View the evaluation results for project
'runnable-agent-test-5d466cbc-97e1' at:
https://smith.langchain.com/o/ebbaf2eb-769b-
4505-aca2-d11de10372a4/datasets/14d8a382-
3c0f-48e7-b212-33489ee8a13e/compare?
selectedSessions=62f0a0c0-73bf-420c-a907-
2c6b2f4625c4

View all tests for Dataset agent-qa-e2d24144
at:
https://smith.langchain.com/o/ebbaf2eb-769b-
4505-aca2-d11de10372a4/datasets/14d8a382-
3c0f-48e7-b212-33489ee8a13e
[>
] 0/5[---------------------------------------
----------->] 5/5
```

```
Server error caused failure to patch
https://api.smith.langchain.com/runs/e9d26fe6-
bf4a-4f88-81c5-f5d0f70977f0 in LangSmith API.
HTTPError('500 Server Error: Internal Server
Error for url:
```

```
https://api.smith.langchain.com/runs/e9d26fe6-
bf4a-4f88-81c5-f5d0f70977f0',
'{"detail":"Internal server error"}')
```

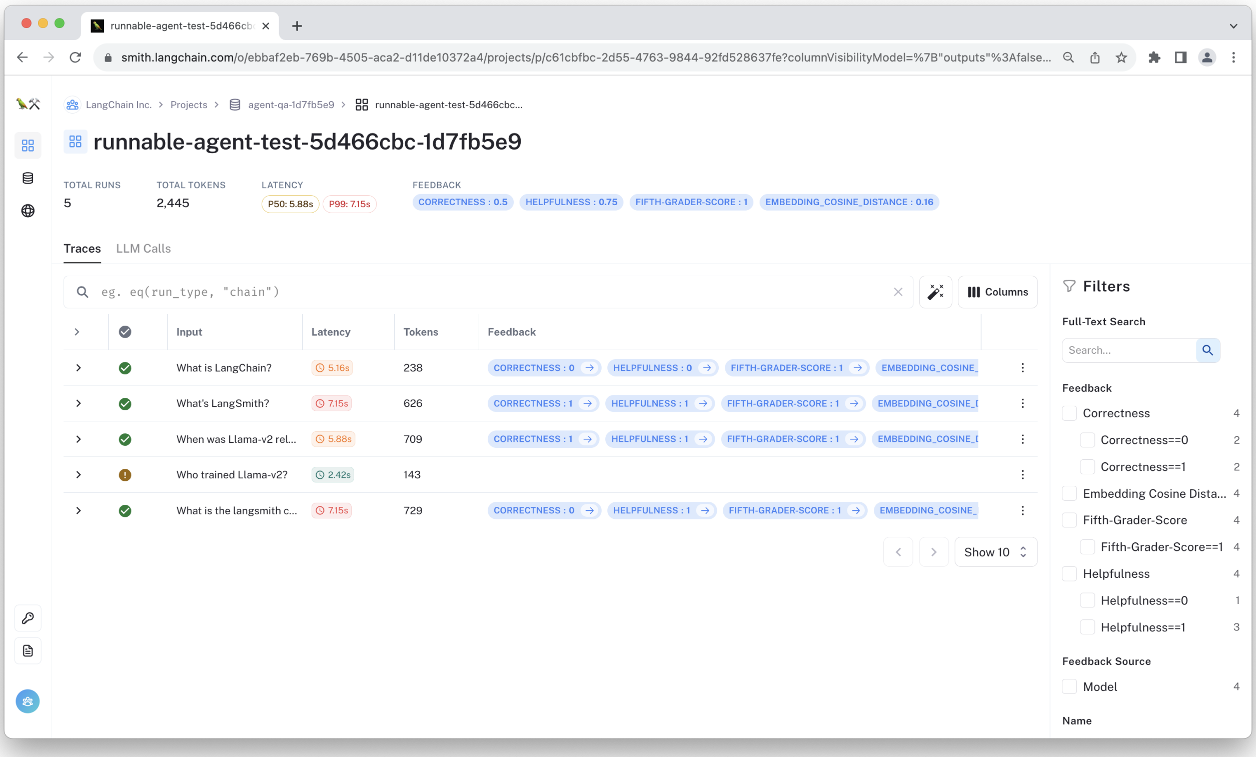
Experiment Results:

	feedback.correctness	feedback.embedding_co
count	5.000000	5.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	0.800000	0.110268
std	0.447214	0.060680
min	0.000000	0.049442
25%	1.000000	0.064186

	feedback.correctness	feedback.embedding_co
50%	1.000000	0.092256
75%	1.000000	0.153003
max	1.000000	0.192453

Review the test results

You can review the test results tracing UI below by clicking the URL in the output above or navigating to the "Testing & Datasets" page in LangSmith **"agent-qa-{unique_id}"** dataset.


runnable-agent-test-5d466cb

smith.langchain.com/o/ebbaf2eb-769b-4505-aca2-d11de10372a4/projects/p/c61cbfbc-2d55-4763-9844-92fd528637fe?columnVisibilityModel=%7B"outputs"%3Afalse...

LangChain Inc. > Projects > agent-qa-1d7fb5e9 > runnable-agent-test-5d466cbc...

runnable-agent-test-5d466cbc-1d7fb5e9

TOTAL RUNS 5 | TOTAL TOKENS 2,445 | LATENCY P50: 5.88s P99: 7.15s | FEEDBACK CORRECTNESS : 0.5 HELPFULNESS : 0.75 FIFTH-GRADER-SCORE : 1 EMBEDDING_COSINE_DISTANCE : 0.16

Traces | LLM Calls

eg. eq(run_type, "chain")

Input	Latency	Tokens	Feedback
What is LangChain?	5.16s	238	CORRECTNESS : 0 HELPFULNESS : 0 FIFTH-GRADER-SCORE : 1 EMBEDDING_COSINE_
What's LangSmith?	7.15s	626	CORRECTNESS : 1 HELPFULNESS : 1 FIFTH-GRADER-SCORE : 1 EMBEDDING_COSINE_I
When was Llama-v2 rel...	5.88s	709	CORRECTNESS : 1 HELPFULNESS : 1 FIFTH-GRADER-SCORE : 1 EMBEDDING_COSINE_I
Who trained Llama-v2?	2.42s	143	
What is the langsmith c...	7.15s	729	CORRECTNESS : 0 HELPFULNESS : 1 FIFTH-GRADER-SCORE : 1 EMBEDDING_COSINE_

Show 10

Filters

Full-Text Search

Feedback

Correctness	4
Correctness==0	2
Correctness==1	2
Embedding Cosine Dista...	4
Fifth-Grader-Score	4
Fifth-Grader-Score==1	4
Helpfulness	4
Helpfulness==0	1
Helpfulness==1	3

Feedback Source

Model	4

Name

This will show the new runs and the feedback logged from the selected evaluators. You can also explore a summary of the results in tabular format below.

```
chain_results.to_dataframe()
```

	inputs.input	outputs.input	outputs.c
63e7ff81-b3f6-40aa-81c8-3600c504d54f	When did langchain first announce the hub?	When did langchain first announce the hub?	LangChai announce LangChai on...
48c2b719-93a3-47cb-baa6-ee93ecb1ba30	What is the langsmith cookbook?	What is the langsmith cookbook?	The Lang Cookboo collectior reci...
96ed16c3-cbc6-4ebb-a335-ec70e64db109	When was Llama-v2 released?	When was Llama-v2 released?	Llama-v2 released 2023.

	inputs.input	outputs.input	outputs.(
7342bb9c-9733-47bd-b88e-cd4ab4f4f82d	What's LangSmith?	What's LangSmith?	LangSmit platform helps develope
aee03fdf-c63e-4f63-9075-a90be3922c7f	What is LangChain?	What is LangChain?	LangChai decentral blockchai platfo...

(Optional) Compare to another prompt

Now that we have our test run results, we can make changes to our agent and benchmark them. Let's try this again with a different prompt and see the results.

```
candidate_prompt = hub.pull("wfh/langsmith-
agent-prompt:39f3bbd0")

chain_results = run_on_dataset(
    dataset_name=dataset_name,
    llm_or_chain_factory=functools.partial(
        create_agent,
prompt=candidate_prompt,
llm_with_tools=llm_with_tools
```

```
    ),
    evaluation=evaluation_config,
    verbose=True,
    client=client,
    project_name=f"runnable-agent-test-
39f3bbd0-{unique_id}",
    project_metadata={
        "env": "testing-notebook",
        "model": "gpt-3.5-turbo",
        "prompt": "39f3bbd0",
    },
)
```

```
View the evaluation results for project
'runnable-agent-test-39f3bbd0-97e1' at:
https://smith.langchain.com/o/ebbaf2eb-769b-
4505-aca2-d11de10372a4/datasets/14d8a382-
3c0f-48e7-b212-33489ee8a13e/compare?
selectedSessions=7753a05e-8235-4bc2-a227-
d0622c1a36a4

View all tests for Dataset agent-qa-e2d24144
at:
https://smith.langchain.com/o/ebbaf2eb-769b-
4505-aca2-d11de10372a4/datasets/14d8a382-
3c0f-48e7-b212-33489ee8a13e
[---------------------------------------------
----->] 5/5
```

Experiment Results:

	feedback.correctness	feedback.embedding_co
count	5.000000	5.000000
unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	0.800000	0.119282
std	0.447214	0.080145
min	0.000000	0.043368
25%	1.000000	0.053311
50%	1.000000	0.107826
75%	1.000000	0.153003

	feedback.correctness	feedback.embedding_co
max	1.000000	0.238903

Exporting datasets and runs

LangSmith lets you export data to common formats such as CSV or JSONL directly in the web app. You can also use the client to fetch runs for further analysis, to store in your own database, or to share with others. Let's fetch the run traces from the evaluation run.

Note: It may be a few moments before all the runs are accessible.

```
runs =
client.list_runs(project_name=chain_results["pr
execution_order=1)
```

```
# After some time, these will be populated.
client.read_project(project_name=chain_results[
{'correctness': {'n': 5, 'avg': 0.8},
 'embedding_cosine_distance': {'n': 5, 'avg':
0.11926},
 'helpfulness': {'n': 5, 'avg': 1.0},
 'not_uncertain': {'n': 5, 'avg': 1.0},
 'score_string:accuracy': {'n': 5, 'avg':
0.82}}
```

Conclusion

Congratulations! You have successfully traced and evaluated an agent using LangSmith!

This was a quick guide to get started, but there are many more ways to use LangSmith to speed up your developer flow and produce better results.

For more information on how you can get the most out of LangSmith, check out LangSmith documentation, and please reach out with questions, feature requests, or feedback at support@langchain.dev.