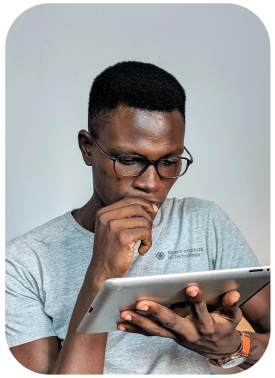
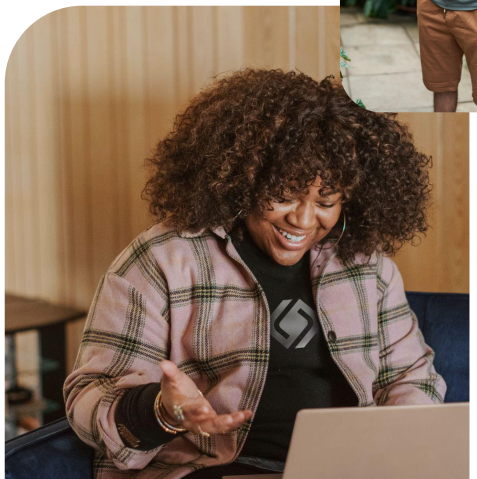
Bloom Institute of Technology

LangSmith

Gain insights into interactions between your code, LLMs, vector databases, and more. Discover a comprehensive tool for clear observability and effective debugging of LLM applications.




Core Competencies

The student must demonstrate...

1. An understanding of what LangSmith is (10 min)
2. How to setup LangSmith (10 min)
3. How to setup LangFuse (5 min)
4. How to observe LLM calls within LangSmith (10 min)
5. Score and create datasets using LangSmith (10 min)

Why do we need LangSmith?

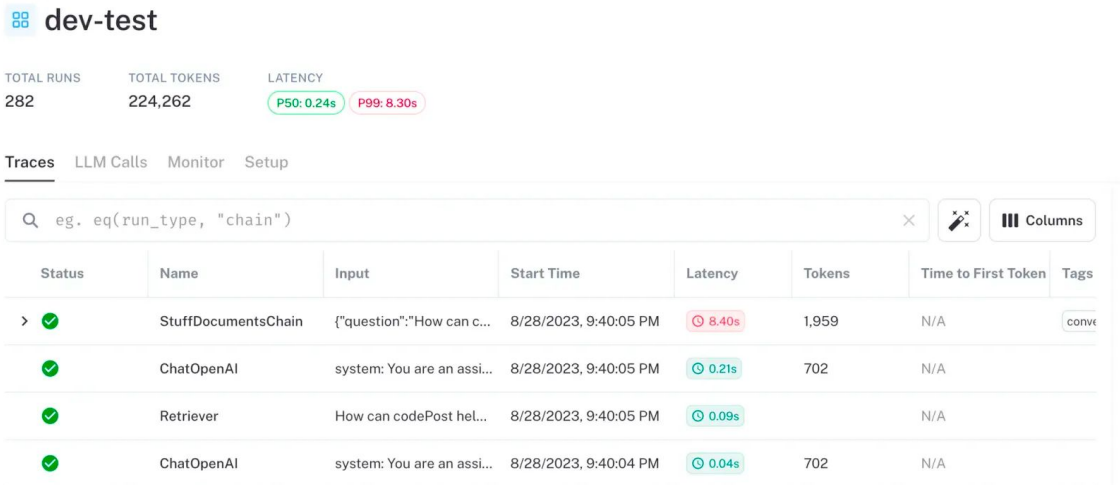
We need LangSmith because it helps manage and troubleshoot large projects by organizing LLM interactions, making it easier to monitor performance and implement improvements.

1. **Track and Debug Interactions**: LangSmith maintains a history of LLM calls, providing insight into the often opaque LLM-based systems.

2. **Assess Response Quality**: Score LLM responses using human-gathered and auto-generated performance metrics.

3. **Utilize Stored Data**: Save LLM responses to build datasets in real-time. This makes future fine-tuning easier to start.

Project Dashboard on LangSmith:



See above: LangSmith tracks every LLM interaction, provides useful data, and helps trace bugs in production apps.

Setting Up and Testing LangSmith

1. **Install LangSmith**
   Command: `pip install langsmith openai`

2. **Create an API Key**
   Navigate to the Settings page.
   Click on `Create API Key.`

3. **Set Up Your Environment**
   Set environment variables:
   ```
   export LANGCHAIN_TRACING_V2=true
   export LANGCHAIN_API_KEY=<your-api-key>
   export OPENAI_API_KEY=<your-openai-api-key>
   ```

4. **Log your first trace**
   Follow the code given to log your first trace. See a demo hosted by LangSmith.

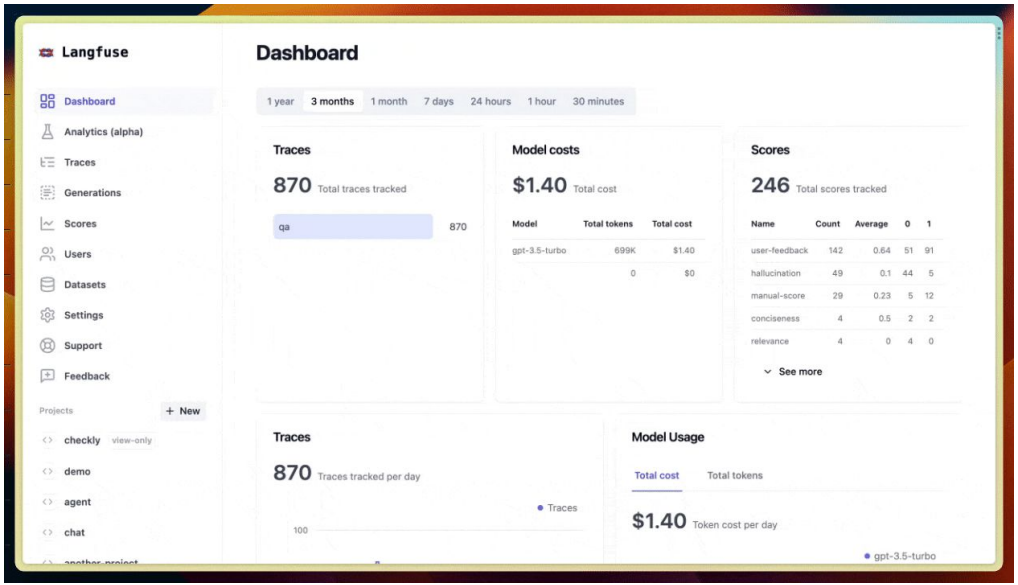
Follow the complete LangSmith setup guide here.

```python
import openai
from langsmith.wrappers import wrap_openai
from langsmith import traceable

# Auto-trace LLM calls in-context
client = wrap_openai(openai.Client())

@traceable # Auto-trace this function
def pipeline(user_input: str):
    result = client.chat.completions.create(
        messages=[{"role": "user", "content":
              user_input}],
        model="gpt-3.5-turbo")
    return result.choices[0].message.content

pipeline("Hello, world!")
# Out:  Hello there! How can I assist you today?
```

***Check For Understanding**: What kind of data is collected by LangSmith?*

Open-Source Alternative: Langfuse

Langfuse is a cheaper alternative and offers a similar feature set to Langsmith. You can run Langfuse privately for free, which is useful for HIPAA compliance.



Langfuse

Setting Up and Testing LangFuse

1. **Install LangSmith**
   Command: `pip install langfuse openai`

2. **Create an API Key**
   Navigate to the Settings page.
   Click on `Create API Key.`

3. **Set Up Your Environment**
   Set environment variables:
   ```
   export LANGFUSE_SECRET_KEY=sk-lf-...
   export LANGFUSE_PUBLIC_KEY=pk-lf-...
   export LANGFUSE_HOST=https://us.cloud.langfuse.com
   ```

4. **Log your first trace**
   Follow the code given to log your first trace.

[Follow the complete LangFuse setup guide here.]

```python
from langfuse.decorators import observe
from langfuse.openai import openai # OpenAI integration

@observe()
def story():
    return openai.chat.completions.create(
        model="gpt-3.5-turbo",
        max_tokens=100,
        messages=[
          {"role": "system", "content": "You are a great storyteller."},
          {"role": "user", "content": "Once upon a time in a galaxy far, far away..."}
        ],
    ).choices[0].message.content

@observe()
def main():
    return story()

main()
```

***Check For Understanding**: Does LangFuse collect the same data as LangSmith? Any differences?*

Class Activity: RAG Inside LangSmith

Please implement LangSmith in the RAG example as demonstrated in the previous slide, and work through the steps on your own. **Answer the outlined questions based on your observations and findings**.

1. What are the three main steps tracked in this system?
2. How long does each step take, and what costs are associated, particularly with the ChatOpenAI call?
3. What documents are retrieved during this step, and how do they contribute to the process?
4. How is the prompt assembled using the documents retrieved in the previous step?
5. Can you identify and explain the components of the prompt in plain text?
6. What response does the system generate based on the given prompt?

Scoring and Datasets on LangSmith

1. **Generate Jokes**:
   Use the provided code to have the LLM generate story-based jokes with a punchline on a specific topic.
2. **Annotation Process in LangSmith**
   Enter the run in LangSmith and annotate each joke call. Add annotated calls to an annotation queue.
3. **Create and Manage Dataset**
   Create a dataset named `dad-jokes`.
   Review each joke in the annotation queue:
   Step 1: Assign a score to each joke (initial scores will be 0).
   Step 2: Edit the joke to refine the punchline. Save the revised joke to the `dad-jokes` dataset. If a joke is unsalvageable, do not add it to the dataset.
   Step 3: Press `Done` to remove the joke from the queue and proceed to the next one.
4. **Utilize the Dataset**
   Use the curated "dad-jokes" dataset for few-shot prompting in your code, drawing from the collection of high-quality examples.

```python
from langchain_openai import ChatOpenAI
from langsmith import traceable

#this traceable call will group all the LLM
calls together in LangSmith

@traceable
def run():
   llm = ChatOpenAI(model="gpt-3.5-turbo")
   for i in range(5):
      llm.invoke("Tell me a dad joke. The
      joke should be in a story format.")

run()
```

***Check For Understanding**: How can you involve other non-devs to help you evaluate your app's output?*

Hands-on Homework.

1. Create an application that generates a response to customer feedback for the fictional company WidgetWorld.

2. 9 reviews are provided for you (3 positive, 3 neutral, 3 negative) on the next slide.

3. Once the reviews have been sent to LangSmith, score each result. You're the evaluator, so decide what looks like good responses.

4. Pay attention to what the response focused on and the response's tone. Save all the results for a dataset.

Appendix: Widget World Reviews

Positive:

"WidgetWorld exceeded my expectations with their top-notch widgets! Our machines run smoother than ever before."
"Impressed by WidgetWorld's attention to detail and commitment to quality. Their widgets are a game-changer for our business."
“WidgetWorld's widgets are reliable and durable, making them an essential component in our manufacturing process."

Neutral:

"WidgetWorld's widgets are decent, but nothing exceptional. They get the job done adequately."
"Average experience with WidgetWorld. Their widgets function as expected, but nothing to write home about."
"WidgetWorld's widgets are alright, but there's room for improvement in terms of durability."

Negative:

"Avoid WidgetWorld at all costs! Their widgets are cheaply made and constantly malfunction."
"Terrible experience with WidgetWorld. Their widgets broke down within weeks of installation."
"WidgetWorld's widgets are a nightmare. We've had nothing but problems since day one."