

RAG 평가

강사장철원

□ LangSmith 소개

RAG 평가

Section 1. LangSmith

Section 1-1. LangSmith의 개념

Section

LangSmith의 개념

LangSmith의 개념

A screenshot of a Google search results page for the query "langsmith". The search bar shows "langsmith". Below the search bar are navigation links: 전체 (selected), 이미지, 동영상, 쇼핑, 뉴스, 짧은 동영상, 도서, and 더보기 (More). The first result is from LangChain, featuring a logo of a person at a keyboard, the text "LangSmith", and a snippet describing it as a unified observability & evals platform. Below this are three additional sections: "Introduction to LangSmith", "How to select examples from a ...", and a link to langchain.com.

Google langsmith X

전체 이미지 동영상 쇼핑 뉴스 짧은 동영상 도서 : 더보기

 LangChain
https://www.langchain.com › langsmith :

[LangSmith](#)

LangSmith is a unified observability & evals platform where teams can debug, test, and monitor AI app performance — whether building with LangChain or not ...

[Introduction to LangSmith](#) >

Learn the essentials of LangSmith — our platform for LLM ...

[How to select examples from a ...](#) >

LangSmith datasets have built-in support for similarity search ...

[langchain.com 검색결과 더보기 »](#)

Section

LangSmith의 개념

LangSmith의 개념

LangSmith

Products ▾ Resources ▾ Docs ▾ Company ▾ Pricing

Get a demo

Sign up

Ship agents with confidence.

LangSmith is a unified observability & evals platform where teams can debug, test, and monitor AI app performance – whether building with LangChain or not.

Get a demo

Sign up for free

Section

LangSmith의 개념

가격 정책

[Products](#) ▾[Resources](#) ▾[Docs](#) ▾[Company](#) ▾[Pricing](#)[Get a demo](#)[Sign up](#)

Plans for teams of any size

Get all the LangChain products – pay for what you use

Developer

For hobbyist projects by solo devs.

Starting at
\$0 / month

then pay as you go

[Start for free](#)

Plus

Batteries-included tooling for building reliable agents fast.

Starting at
\$39 / month

then pay as you go

[Sign up](#)

Enterprise

For teams with advanced deployment, security, and support needs.

Custom

[Get a demo](#)

LangSmith의 개념



How would you like to start?

Select a workflow to begin exploring LangSmith's capabilities

Trace an Application

Analyze and debug applications using traces.



Test Prompts in the Playground

Iterate and test on prompts across any model or provider.



Run an Evaluation

Measure app performance: identify failures, compare changes, ensure reliability.



Build and Deploy Agents

Deploy agentic applications to production with LangGraph Platform.



Skip →

Section

LangSmith의 개념

LangSmith의 개념

The screenshot shows the LangSmith web interface with a dark theme. At the top, there is a navigation bar with icons for back, forward, search, and home, followed by the URL smith.langchain.com/o/fbd38fe2-6868-4a85-9212-e4c671723814, a star icon, a copy icon, a refresh icon, and a user profile icon.

The main area is titled "Personal" and features a sidebar with various icons: a person (Personal), a house (Home), a gear (Observability), a plus sign (Tracing Projects), a double arrow (Custom Dashboards), and a gear (Settings).

Personal section:

- ID**: A purple circular icon with a white person symbol.
- Get Started** buttons:
 - Set up tracing**: Contains a gear icon and a link.
 - Run an evaluation**: Contains a stopwatch icon and a link.
 - Try out playground**: Contains a circular arrow icon and a link.

Observability section:

- Tracing Projects**: 1 project listed.
- Custom Dashboards**: 0 dashboards listed.

Name	Most Recent Run (7D)	Feedback (7D)	Run Count (7D)	Error Rate (7D)	P50 Latency (7D)	P99 Latency (7D)
default	2025. 7. 12. 오전 7:29:20		1	0%	⌚ 7.31s	⌚ 7.31s

At the bottom, it says "Showing 5 most active projects." and "View all tracing projects →".

Section

LangSmith의 개념

LangSmith의 개념

The screenshot displays the LangSmith tracing interface. On the left, the main dashboard shows a project named "default" with tabs for Runs, Threads, Alerts, and Setup. A search bar at the top allows filtering by name or ID. The "Runs" tab is selected, showing a list of traces. One trace, "Sample Agent Trace", is highlighted. The main panel on the right provides detailed information about this trace:

- Sample Agent Trace**: The trace ID is shown.
- Run**: The trace was run at **START TIME** 07/12/2025, 07:29:20 AM and completed at **END TIME** 07/12/2025, 07:29:28 AM.
- Input**: The input question was "What is a document loader?". The trace details show calls to "search_latest_knowledgebase" (0.76s), "VectorStoreRetriever" (0.52s), and "ChatOpenAI" (3.44s).
- Output**: The output was "BEEP BOOP! A document loader is a component that retrieves data from a specific source and returns it as a LangChain "Document." You can find more information about document loader integrations [here] (/docs/modules/data_connection/document_loaders/). Each loader is designed to retrieve data from a particular source and return it in a format that can be processed by LangChain."
- Status**: The status is **Success**.
- Total Tokens**: 540 tokens.
- Latency**: 7.31s.
- Type**: Chain.

LangSmith

Section 1. LangSmith

Section 1-2. Tracing - LangChain

Section

Tracing - LangChain

Tracing

The screenshot shows the 'Tracing Projects' page. At the top left is a sidebar with icons for Home, Personal, Tracing Projects, and other developer tools. The top right features a 'Setup resource tags' button, a 'DEVELOPER' badge, and a red-bordered 'New Project' button. The main title 'Tracing Projects' is centered above a subtitle 'Analyze and debug applications.' Below this is a search bar with placeholder text 'Search by name...' and a 'Columns' button. A table lists one project: 'default'. The table includes columns for Name (sorted by default), Status, and Actions. Navigation at the bottom includes 'Page 1' and 'Show 10' buttons.

Personal > Tracing Projects

Setup resource tags

DEVELOPER

Tracing Projects

Analyze and debug applications.

Search by name...

Columns

Name ↑	Status	Actions
default	Active	⋮

Page 1 < > Show 10

Section

Tracing - LangChain

Tracing

GET STARTED WITH LANGSMITH

Set up observability

Trace, debug and monitor your application

With LangChain Without LangChain

1. Generate API Key

2. Install dependencies

Python TypeScript

```
1 pip install -U langchain langchain-openai
```

Copy

3. Configure environment to connect to LangSmith.

Project Name

pr-warmhearted-making-60

```
1 LANGSMITH_TRACING=true
2 LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
3 LANGSMITH_API_KEY="<your-api-key>"
4 LANGSMITH_PROJECT="pr-warmhearted-making-60"
5 OPENAI_API_KEY="<your-openai-api-key>"
```

Copy

4. Run any LLM, Chat model, or Chain. Its trace will be sent to the set project.

```
1 from langchain_openai import ChatOpenAI
2
3 llm = ChatOpenAI()
4 llm.invoke("Hello, world!")
```

Copy

Section

Tracing - LangChain

Tracing

보관 주의

GET STARTED WITH LANGSMITH

Set up observability

Trace, debug and monitor your application

With LangChain Without LangChain

1. New key created:

```
1
```

2. Install dependencies

Python TypeScript

```
1 pip install -U langchain langchain-openai
```

3. Configure environment to connect to LangSmith.

Project Name

pr-warmhearted-making-60

```
1 LANGSMITH_TRACING=true
2 LANGSMITH_ENDPOINT="https://api.smith.langchain.com"
3 LANGSMITH_API_KEY=
4 LANGSMITH_PROJECT="pr-warmhearted-making-60"
5 OPENAI_API_KEY="<your-openai-api-key>"
```

4. Run any LLM, Chat model, or Chain. Its trace will be sent to the set project.

Section

Tracing - LangChain

코드 실행(1)

```
[10]: import os  
  
os.environ["LANGCHAIN_TRACING_V2"]="true"  
os.environ["LANGCHAIN_ENDPOINT"]="https://api.smith.langchain.com"  
os.environ["LANGCHAIN_API_KEY"]=[REDACTED]  
os.environ["LANGCHAIN_PROJECT"]="pr-shadowy-scow-41"
```

```
[11]: from langchain_huggingface import HuggingFacePipeline  
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
```

```
[12]: # 토크나이저 & 모델 불러오기  
model_id = "skt/kogpt2-base-v2"  
tokenizer = AutoTokenizer.from_pretrained(model_id)  
model = AutoModelForCausalLM.from_pretrained(model_id)
```

```
[13]: # 파이프라인 생성  
text_gen = pipeline(  
    "text-generation",  
    model=model,  
    tokenizer=tokenizer,  
    device=-1,  
    truncation=True,  
    max_length=50,  
    do_sample=True,  
    temperature=0.7,  
)
```

Device set to use cpu

Section

Tracing - LangChain

코드 실행(2)

[14]: # 프롬프트 예시

```
prompt = "산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 "
```

LangChain으로 실행

```
response = llm.invoke(prompt)
```

[15]: print("생성된 문장:", response)

생성된 문장: 산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 뭔가 일이 벌어졌습니다. 토끼는 어디론가 사라졌습니다. 그 토끼는 어디선가 날아온 건데, 이상하게도 이상한 소리를 내며 날아가고 있었습니다. 그 소리는 바로 '꿈틀거리는 토끼'였습니다."

1942년 4월, 나는 토끼에게 달려들었다.

토끼는 내가 미처 생각하지 못했던 바로 그 소리를 냈습니다.

"아! 이것은 꿈틀거리는 토끼입니다."

"그래요! 꿈틀거리는 토끼입니다."

토끼는 나를 향해 달려가더니 나를 향해 손을 뻗었습니다.

그 순간

Section

Tracing - LangChain

실행 결과

The screenshot displays two main panels from the OpenTelemetry Tracer interface.

Left Panel (Runs View):

- Path: Personal > Tracing Projects > pr-shadowy-scow-41
- Project Name: pr-shadowy-scow-41
- Filters: 1 filter (Last 7 days), Traces, LLM Calls, All Runs
- Run Details:
 - Name: HuggingFacePipeline
 - Input: ["산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 뭔가 일이 벌어졌습니다. 토끼는 어디론가 사라졌습니다. 그 토끼는 어디선가 날아온 건데, 이상하게도 이상한 소리를 내며 날아가고 있었습니다. 그 소리는 바로 '꿈틀거리는 토끼'였습니다."
1942년 4월, 나는 토끼에게 달려들었다.
토끼는 내가 미처 생각하지 못했던 바로 그 소리를 냈습니다.
"아! 이것은 꿈틀거리는 토끼입니다."
"그래요! 꿈틀거리는 토끼입니다."
토끼는 나를 향해 달려가더니 나를 향해 손을 뻗었습니다.
그 순간
 - Output: [{"query": "데이터 무제...", "results": "질문: ..."}]

Right Panel (HuggingFacePipeline View):

- Panel Title: HuggingFacePipeline
- Run Details:
 - Run ID: ID
 - Playground, Add to, Copy buttons
- Prompt & Completion:
 - Text Content: 산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 산 속에 토끼 한 마리가 살고 있습니다. 그러던 어느 날 뭔가 일이 벌어졌습니다. 토끼는 어디론가 사라졌습니다. 그 토끼는 어디선가 날아온 건데, 이상하게도 이상한 소리를 내며 날아가고 있었습니다. 그 소리는 바로 '꿈틀거리는 토끼'였습니다.
1942년 4월, 나는 토끼에게 달려들었다.
토끼는 내가 미처 생각하지 못했던 바로 그 소리를 냈습니다.
"아! 이것은 꿈틀거리는 토끼입니다."
"그래요! 꿈틀거리는 톤입니다."
토끼는 나를 향해 달려가더니 나를 향해 손을 뻗었습니다.
그 순간
 - Metadata: START TIME 07/11/2025, 06:08:26 PM, END TIME 07/11/2025, 06:08:29 PM, TIME TO FIRST TOKEN, STATUS Success, TOTAL TOKENS 324 tokens, LATENCY 3.44s, TYPE LLM

RAG 평가

Section 1. LangSmith

Section 1-3. Tracing - LangGraph

Section

Tracing - LangGraph

Tracing

```
[1]: import os  
  
os.environ["LANGCHAIN_TRACING_V2"]="true"  
os.environ["LANGCHAIN_ENDPOINT"]="https://api.smith.langchain.com"  
os.environ["LANGCHAIN_API_KEY"]=[REDACTED]  
os.environ["LANGCHAIN_PROJECT"]="pr-shadowy-scow-41"
```

```
[2]: from langgraph.graph import StateGraph, END  
  
# 분류 노드  
def classification(state):  
    text = state.get("user_input", "")  
  
    if any(word in text for word in ["좋아요", "네", "좋다", "응"]):  
        label = "positive"  
    elif any(word in text for word in ["싫어요", "아니요", "별로", "싫다"]):  
        label = "negative"  
    else:  
        label = "neutral"  
    return {**state, "label": label}  
  
# 응답 노드  
def positive_answer(state):  
    return {**state, "response": "좋게 생각해주세요 감사합니다!"}  
  
def negative_answer(state):  
    return {**state, "response": "무엇이 불편하셨나요?"}  
  
def neutral_answer(state):  
    return {**state, "response": "조금 더 자세히 말씀해 주세요."}  
  
def get_label(state):  
    return state.get("label", "")
```

Section

Tracing - LangGraph

Tracing

```
# 그래프 설계
graph = StateGraph(dict)

graph.add_node("classification", classification)
graph.add_node("positive", positive_answer)
graph.add_node("negative", negative_answer)
graph.add_node("neutral", neutral_answer)

graph.set_entry_point("classification")

graph.add_conditional_edges("classification", get_label, {
    "positive": "positive",
    "negative": "negative",
    "neutral": "neutral"
})

# 각 노드 끝나면 종료
graph.add_edge("positive", END)
graph.add_edge("negative", END)
graph.add_edge("neutral", END)

# 5. 실행기 구성
app = graph.compile()

# 6. 테스트 실행 (한글 입력)
user_input = input("한글 입력: ")

final_state = app.invoke({"user_input": user_input})
print("응답:", final_state.get("response", ""))
```

한글 입력: 이 서비스 너무 좋아요
응답: 좋게 생각해주세요 감사합니다!

Section

Tracing - LangGraph

Tracing

The screenshot shows the LangChain tracing interface. On the left, the 'Runs' tab of the 'pr-shadowy-scow-41' project is selected, displaying three runs: LangGraph, HuggingFacePipeline, and QAEvalChain. The LangGraph run is highlighted with a red box around its internal components: classification (0.00s), get_label (0.00s), and positive (0.00s). The main panel shows the 'LangGraph' run details, including the trace ID, input ('이 서비스 너무 좋아요'), output ('positive'), and timestamp (07/11/2025, 06:17:06 PM). The output section also includes a 'Label' field ('positive') and a 'Response' field ('좋게 생각해주셔서 감사합니다!').

Personal > Tracing Projects > pr-shadowy-scow-41

pr-shadowy-scow-41

Runs Threads Alerts Setup

1 filter Last 7 days Traces LLM Calls All Runs

Name Input Output

LangGraph 이 서비스 너무 좋아요 positive

HuggingFacePipeline ["산 속에 토끼 한 마리가 ..." 산 속에 토끼 한 마

QAEvalChain [{"query": "데이터 무제...", "results": "질문:"}]

TRACE Waterfall LangGraph 0.01s classification 0.00s get_label 0.00s positive 0.00s

LangGraph ID

Run Feedback Metadata

Input

User Input
이 서비스 너무 좋아요

Output

Label
positive

Response
좋게 생각해주셔서 감사합니다!

User Input
이 서비스 너무 좋아요

START TIME
07/11/2025, 06:17:06 PM

END TIME
07/11/2025, 06:17:06 PM

TIME TO FIRST TOKEN

STATUS
Success

TOTAL TOKENS
0 tokens

LATENCY
0.01s

TYPE
Chain

RAG 평가

Section 1. LangSmith

Section 2-1. Monitoring

Monitoring

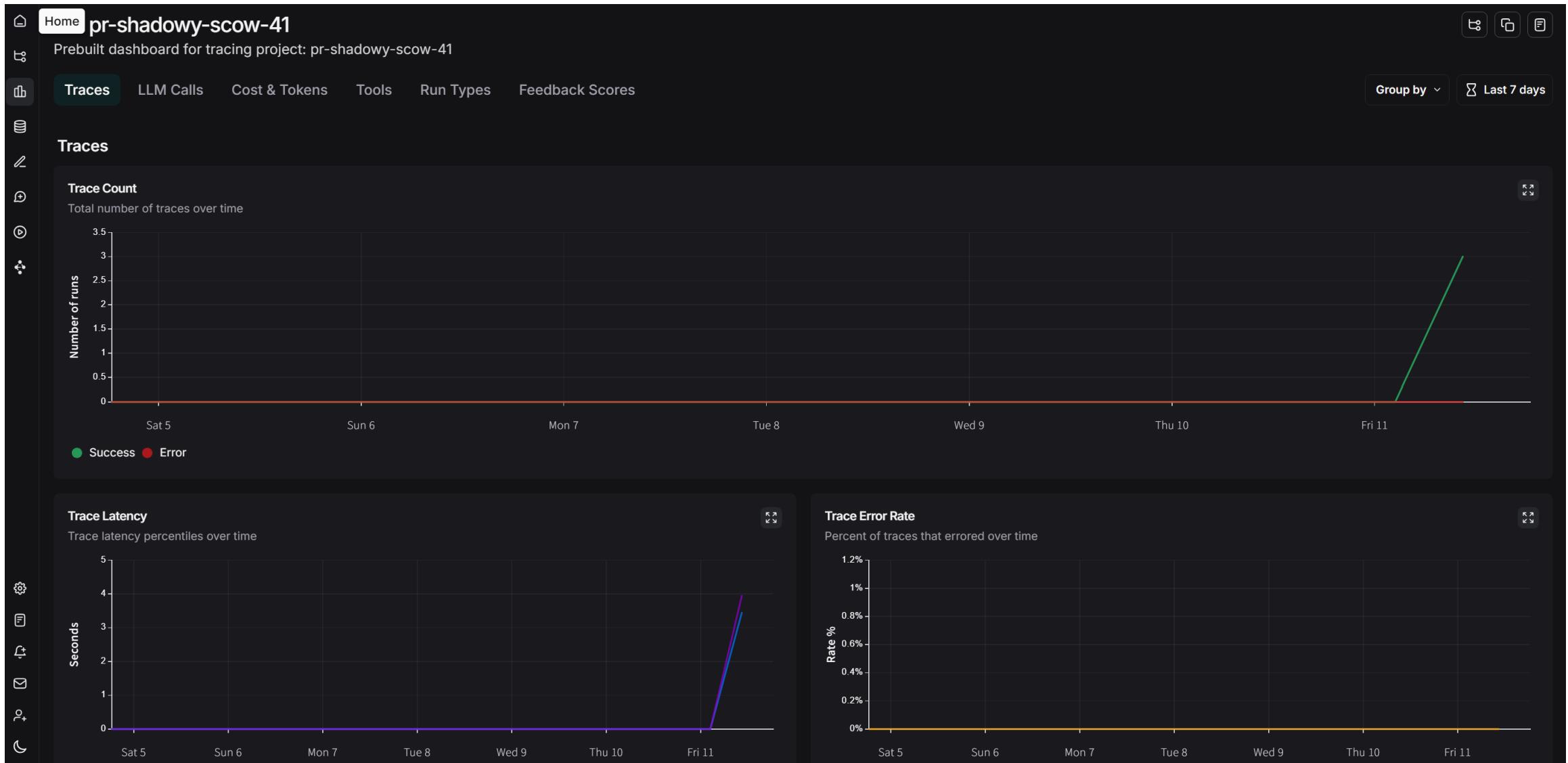
The screenshot shows a dark-themed monitoring interface. On the left, there's a vertical sidebar with icons for Personal, Monitoring, Home, and Settings. The 'Monitoring' icon is highlighted with a red box. The main area has a breadcrumb navigation bar: Personal > Monitoring. Below it, the title 'Monitoring' is displayed with the subtitle 'Monitor trends and performance in real time.' A red box highlights the 'Prebuilt Dashboards' tab, which is currently selected. Other tabs include 'Custom Dashboards' and 'Alerts'. A search bar labeled 'Search by name...' is present. The main content area displays two prebuilt dashboards: 'default' (Most Recent Run: 2025. 7. 12. 오전 7:29:20, Run Count: 1) and 'pr-shadowy-scow-41' (Most Recent Run: 2025. 7. 11. 오후 6:17:06, Run Count: 3). The 'pr-shadowy-scow-41' row is also highlighted with a red box.

Tracing Project Name	Most Recent Run (7D)	Run Count (7D)
default	2025. 7. 12. 오전 7:29:20	1
pr-shadowy-scow-41	2025. 7. 11. 오후 6:17:06	3

Section

Monitoring

Monitoring



RAG 평가

Section 1. LangSmith

Section 3-1. Evaluation

Evaluation

평가를 할 수 있긴하지만...

The screenshot shows the Cohere API interface with the following navigation path: Personal > Datasets & Experiments > kogpt2-korean-qa-demo1 > bold-interest-26. The main view displays two examples under the 'bold-interest-26' tab. The first example is "서울은 어디 있나요?" with a reference output "서울은 대한민국의 수도입니다." and a generated output starting with "Q: 서울은 어디 있나요? A: 아니죠. 그런데 저는 예. 저도 헷갈립니다. 이번에도 제가 어~ 여러분들한테". The second example is "데이터 무제한 요금제는 어떻게 되나요?" with a reference output "데이터 무제한 요금제는 24GB 이후 속도제어 중" and a generated output starting with "Q: 데이터 무제한 요금제는 어떻게 되나요? A: 데이터 무제한 요금제는 데이터 무제한 요금제와 비슷한". The interface includes a sidebar with various icons and a 'Compact' button.

Inputs	Reference Outputs	Outputs
서울은 어디 있나요?	서울은 대한민국의 수도입니다.	Q: 서울은 어디 있나요? A: 아니죠. 그런데 저는 예. 저도 헷갈립니다. 이번에도 제가 어~ 여러분들한테
데이터 무제한 요금제는 어떻게...	데이터 무제한 요금제는 24GB 이후 속도제어 중	Q: 데이터 무제한 요금제는 어떻게 되나요? A: 데이터 무제한 요금제는 데이터 무제한 요금제와 비슷한

- evaluator API가 까다롭고 불투명
- 오직 OpenAI 기반 evaluator에 맞춰져있음
- 커스텀 evaluator 사용 불편
- evaluate_strings와 같은 비직관적 추상 메소드 강제

감사합니다.

Q & A