

- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: <https://smith.langchain.com/hub/teddynote/summary-stuff-documents>

5. 05. LLM-as-Judge

2) Question-Answer Evaluator

- 가장 기본 기능을 가진 `Evaluator` = `Query` - `Answer` 평가하기
- 사용자 입력 = `input` → LLM 이 생성한 답변 → `prediction` 으로 정답 답변은 `reference` 로 정의됨
 - `Prompt` 변수
 - `query`: 질문
 - `result`: LLM 답변
 - `answer`: 정답 답변
- 새로운 가상환경 생성 - `lc_eval_env`
 - `Python-3.12`
 - `Pydantic`: ver 1.10.18

```
import pydantic
print(f"Pydantic 버전: {pydantic.__version__}")
```

- Pydantic 버전: 1.10.18 - (0.1s)**

- 환경 설정

```
# API 키를 환경변수로 관리하기 위한 설정 파일
from dotenv import load_dotenv
```

```
# API 키 정보 로드
load_dotenv() # True
```

```
from langsmith import Client
from langsmith import traceable

import os

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_V2')
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANGCHAIN_API_KEY') else "설정되지 않음" # API 키 값은 직접 출력하지 않음

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API_KEY') and langchain_project:
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{langchain_tracing_v2}')
```

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
```

```
✔ LangSmith 프로젝트: 'LangChain-prantice'
✔ LangSmith API Key: 설정됨
-> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

```
import os
from myrag import PDFRAG # local 임베딩 버전으로 수정한 myrag.py 불러오기
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_google_genai import ChatGoogleGenerativeAI

# API 키 확인
from dotenv import load_dotenv

GOOGLE_API_KEY=os.getenv("GOOGLE_API_KEY2")

if not os.getenv("GOOGLE_API_KEY2"):
    os.environ["GOOGLE_API_KEY2"] = input("Enter your GOOGLE_API_KEY2: ")

if "GOOGLE_API_KEY2" not in os.environ:
    print("❌ 경고: GOOGLE_API_KEY2 환경 변수가 설정되지 않았습니다. 반드시 설정해야 gemini LLM이 작동합니다.")
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

llm2 = ChatGoogleGenerativeAI(model="gemini-2.5-flash-lite", temperature=0, api_key=os.getenv("GOOGLE_API_KEY2"))

print("✔ gemini-2.5-flash-lite 성공!")
```

- ✔ 성공!
 - API 할당량 부족으로 두번째 계정으로 다시 시도

```
from langchain_google_genai import ChatGoogleGenerativeAI

llm2 = ChatGoogleGenerativeAI(model="gemini-2.5-flash-lite", temperature=0, api_key=os.getenv("GOOGLE_API_KEY2"))

rag = PDFRAG(
    "../15_Evaluations/data/SPRI_AI_Brief_2023년12월호_F.pdf",
    llm2,
)

print("✔ PDFRAG 생성 성공!")
```

- PDFRAG 생성 - (4.9s)

```
✔ 문서 로드 완료: 23개 페이지
✔ 문서 분할 완료: 43개 청크
✔ 임베딩 모델 로드: all-MiniLM-L6-v2
✔ 벡터스토어 생성 완료
✔ PDFRAG 생성 성공!
```

```
# 검색기(retriever) 생성
retriever = rag.create_retriever()
```

- ✔ 검색기 생성 완료 (k=4)

```
# 체인(chain) 생성
chain = rag.create_chain(retriever)
```

- ✔ RAG 체인 생성 완료

```
# 질문
answer = chain.invoke("삼성전자가 자체 개발한 생성형 AI의 이름은 무엇인가요?")
print(answer)
```

- 삼성전자가 자체 개발한 생성형 AI의 이름은 '삼성 가우스'입니다. - (2.0s)

```
# 질문에 대한 답변하는 함수를 생성
def ask_question(inputs: dict):
    return {"answer": chain.invoke(inputs["question"])}

# 사용자 질문 예시
llm_answer = ask_question(
```

```
        {"question": "삼성전자가 자체 개발한 생성형 AI의 이름은 무엇인가요?"}  
    )  
  
    llm_answer
```

- `ask_question()` - (1.2s)

```
{'answer': '삼성전자가 자체 개발한 생성형 AI의 이름은 '삼성 가우스'입니다.'}
```

```
# evaluator prompt 출력을 위한 함수  
def print_evaluator_prompt(evaluator):  
    return evaluator.evaluator.prompt.pretty_print()
```

```
from langsmith.evaluation import evaluate, LangChainStringEvaluator  
from langchain_google_genai import ChatGoogleGenerativeAI  
  
# qa 평가자 생성  
qa_evaluator = LangChainStringEvaluator(  
    "qa",  
    config={  
        "llm": ChatGoogleGenerativeAI(  
            model="gemini-2.5-flash-lite",  
            temperature=0,  
            google_api_key=os.getenv("GOOGLE_API_KEY2"))  
        }  
    )  
  
# 프롬프트 출력  
print_evaluator_prompt(qa_evaluator)
```

- `qa_evaluator` - (0.1s)

```
You are a teacher grading a quiz.  
You are given a question, the student's answer, and the true answer, and are asked to score the student answer as  
  
Example Format:  
QUESTION: question here  
STUDENT ANSWER: student's answer here  
TRUE ANSWER: true answer here  
GRADE: CORRECT or INCORRECT here  
  
Grade the student answers based ONLY on their factual accuracy. Ignore differences in punctuation and phrasing be  
  
QUESTION: *{query}*  
STUDENT ANSWER: *{result}*  
TRUE ANSWER: *{answer}*  
GRADE:
```

- 평가 진행 → 출력한 URL 이동 → 결과 확인하기

```
dataset_name = "RAG_EVAL_DATASET"  
  
# 평가 실행  
experiment_results = evaluate(  
    ask_question,  
    data=dataset_name,  
    evaluators=[qa_evaluator],  
    experiment_prefix="RAG_EVAL",  
    # 실험 메타데이터 지정  
    metadata={  
        "variant": "QA Evaluator 를 활용한 평가",  
    },  
)
```

- 결과 확인하기

- `RAG_EVAL / COMPILED` - (45s)

Personal > Datasets & Experiments > RAG_EVAL_DATASET > RAG_EVAL-eb9eee01

RAG_EVAL-eb9eee01 202507../35b940

Inputs	Reference Outputs	Outputs	Latency 1.821 p50	Completed	Tokens 8,193.00	Cost 0.0013
삼성전자가 만든 생성형 AI의 ... #7080 →	삼성전자가 만든 생성형 AI의 이름은 테디노트 일...	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	1.95s	COMPLETED 1	1,913	\$0.00
코히어의 데이터 출처 탐색기에 ... #7184 →	코히어의 데이터 출처 탐색기는 AI 모델 훈련에 사...	그에 대한 정보는 제공된 문서에 포함되어 있지 않...	1.23s	COMPLETED 1	1,261	\$0.00
미국 바이든 대통령이 안전하고... #8d34 →	2023년 10월 30일 미국 바이든 대통령이 행정명...	그에 대한 정보는 제공된 문서에 포함되어 있지 않...	2.19s	COMPLETED 1	1,711	\$0.00
구글이 테디노트에게 20억달러... #b859 →	사실이 아닙니다. 구글은 엑스로픽에 최대 20억 달...	그에 대한 정보는 제공된 문서에 포함되어 있지 않...	1.56s	COMPLETED 1	1,395	\$0.00
삼성전자가 만든 생성형 AI의 ... #e0d6 →	삼성전자가 만든 생성형 AI의 이름은 삼성 가우스...	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	1.82s	COMPLETED 1	1,913	\$0.00

- o
- o
- o RAG_EVAL / CORRECT=1, INCORRECT=4 - (7.8s)

Personal > Datasets & Experiments > RAG_EVAL_DATASET > RAG_EVAL-f6913da6

RAG_EVAL-f6913da6 202507../35b940

Compact Full Diff Default Group by Display + Compare

Inputs	Reference Outputs	Outputs	Correctness 0.20 AVG	Latency 1.703 p50	Completed
삼성전자가 만든 생성형 AI의 ... #7080 →	삼성전자가 만든 생성형 AI의 이름은 테디노트 일...	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	incorrect	4.55s	COMPLETE
코히어의 데이터 출처 탐색기에 ... #7184 →	코히어의 데이터 출처 탐색기는 AI 모델 훈련에 사...	그에 대한 정보는 제공된 문서에서 찾을 수 없습니...	incorrect	1.59s	COMPLETE
미국 바이든 대통령이 안전하고... #8d34 →	2023년 10월 30일 미국 바이든 대통령이 행정명...	그에 대한 정보는 제공된 문서에 포함되어 있지 않...	incorrect	1.93s	COMPLETE
구글이 테디노트에게 20억달러... #b859 →	사실이 아닙니다. 구글은 엑스로픽에 최대 20억 달...	그에 대한 정보는 제공된 문서에 포함되어 있지 않...	incorrect	1.70s	COMPLETE
삼성전자가 만든 생성형 AI의 ... #e0d6 →	삼성전자가 만든 생성형 AI의 이름은 삼성 가우스...	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	correct	1.70s	COMPLETE

o

- RAG 시스템 개선하기
 - o a. Retriever 개선하기
 - o b. Chunk Size 조정하기

```
# =====
# Chunk Size 조정
# =====
from myrag2 import PDFRAG
from langchain_google_genai import ChatGoogleGenerativeAI
import os
from dotenv import load_dotenv

load_dotenv()

# Chunk Size 조정
rag2 = PDFRAG(
    "../15_Evaluations/data/SPRI_AI_Brief_2023년12월호_F.pdf",
    ChatGoogleGenerativeAI(
        model="gemini-2.0-flash-lite",
        temperature=0,
        google_api_key=os.getenv("GOOGLE_API_KEY2")),
)
```



- rag2 생성 - (12.2s)

- ✓ 문서 로드 완료: 23개 페이지
- ✓ 문서 분할 완료: 72개 청크
- ✓ 임베딩 모델 로드: all-MiniLM-L6-v2
- ✓ 벡터스토어 생성 완료

```
# =====
# Top-K 증가
# =====

# 개선
retriever2 = rag2.create_retriever() # myrag2.py 불러오기
```

```
# 체인 재생성
chain2 = rag2.create_chain(retriever2)
```

-  **검색기 생성 완료**
-  **RAG 체인 생성 완료**

```
# 질문에 대한 답변 생성
chain2.invoke("삼성전자가 자체 개발한 생성형 AI의 이름은 무엇인가요?")
```

- **'삼성전자가 자체 개발한 생성형 AI의 이름은 제공된 문서에서 찾을 수 없습니다.'** - (1.4s)

```
# 질문에 대한 답변하는 함수를 생성
def ask_question(inputs: dict):
    return {"answer": chain2.invoke(inputs["question"])}

# 사용자 질문 예시
llm_answer = ask_question(
    {"question": "삼성전자가 자체 개발한 생성형 AI의 이름은 무엇인가요?"}
)

llm_answer
```

- **ask_question()** - (1.3s)

```
{'answer': '삼성전자가 자체 개발한 생성형 AI의 이름은 제공된 문서에서 찾을 수 없습니다.'}
```

```
# evaluator prompt 출력을 위한 함수
def print_evaluator_prompt(evaluator):
    return evaluator.evaluator.prompt.pretty_print()
```

```
from langsmith.evaluation import evaluate, LangChainStringEvaluator
from langchain_google_genai import ChatGoogleGenerativeAI

# qa 평가자 생성
qa_evaluator = LangChainStringEvaluator(
    "qa",
    config={
        "llm": ChatGoogleGenerativeAI(
            model="gemini-2.5-flash-lite",
            temperature=0,
            google_api_key=os.getenv("GOOGLE_API_KEY2"))
    }
)

# 프롬프트 출력
print_evaluator_prompt(qa_evaluator)
```

- **qa_evaluator**

```
You are a teacher grading a quiz.
You are given a question, the student's answer, and the true answer, and are asked to score the student answer as

Example Format:
QUESTION: question here
STUDENT ANSWER: student's answer here
TRUE ANSWER: true answer here
GRADE: CORRECT or INCORRECT here

Grade the student answers based ONLY on their factual accuracy. Ignore differences in punctuation and phrasing be

QUESTION: *{query}*
STUDENT ANSWER: *{result}*
TRUE ANSWER: *{answer}*
GRADE:
```

```
dataset_name = "RAG_EVAL_DATASET"
```

```
# 평가 실행
```

```
experiment_results = evaluate(  
    ask_question,  
    data=dataset_name,  
    evaluators=[qa_evaluator],  
    experiment_prefix="RAG_EVAL_2",  
    # 실험 메타데이터 지정  
    metadata={  
        "variant": "myrag2.py 반영",  
    },  
)
```

• 개선 시도: **RAG_EVAL_2**

View the evaluation results for experiment: 'RAG_EVAL_2-00c422bd' at:
<https://smith.langchain.com/o/2c3342d3-1170-4ffa-86fd-f621199e0b9c/datasets/420dd308-2ebd-44c9-8ce8-9aff3886dc8e/>














0it [00:00, ?it/s]

5it [00:04, 1.12it/s]

• 결과 확인하기

*

• **RAG_EVAL** / **CORRECT=2**, **INCORRECT=3** - (**7.3s**)

RAG_EVAL_2-00c422bd   202507./35b940  Compact Full Diff Default  Group by Display  + Compare								
Inputs	Reference Outputs	Outputs	Correctness 0.40 Avg 	Latency 1.453 P50 	Completed	Tokens 9,911.00	Cost 0.00	
삼성전자가 만든 생성형 AI의 ...  #7080	삼성전자가 만든 생성형 AI의 이름은 테디노트 입니	삼성전자가 자체 개발한 생성형 AI의 이름은 '삼성	incorrect	1.45s	COMPLETED	1, 2,036	\$0	
코히어의 데이터 출처 탐색기에 ...  #7184	코히어의 데이터 출처 탐색기는 AI 모델 훈련에 사	I am sorry, but I cannot find any informati	incorrect	1.45s	COMPLETED	1, 1,852	\$0	
미국 바이든 대통령이 안전하고...  #8d34	2023년 10월 30일 미국 바이든 대통령이 행정명	저는 해당 정보에 대한 문서를 찾을 수 없습니다. 	incorrect	1.54s	COMPLETED	1, 2,457	\$0	
구글이 테디노트에게 20억달러...  #b859	사실이 아닙니다. 구글은 엔스룩릭에 최대 20억 달	아니요, 구글은 테디노트가 아닌 엔스룩릭에 20억	correct	1.45s	COMPLETED	1, 1,530	\$0	
삼성전자가 만든 생성형 AI의 ...  #e0d6	삼성전자가 만든 생성형 AI의 이름은 삼성 가우스	삼성전자가 자체 개발한 생성형 AI의 이름은 '삼성	correct	1.51s	COMPLETED	1, 2,036	\$0	

• **RAG** 시스템 개선하기_2

- c. **chunk_size** 조정
- d. 찾은 커널 충돌 → **python** 파일로 실행하기

• C.

- **chunk_size** 조절 test
 - **chunk_size** 지정 **✗** or **chunk_size** = **4** → 검색 결과 없음
 - '죄송합니다. 제공된 문서에서 삼성전자가 자체 개발한 생성형 AI의 이름에 대한 정보를 찾을 수 없습니다.'
 - **chunk_size** = **7** → 검색 결과 **○**
 - "삼성전자가 자체 개발한 생성형 AI의 이름은 '삼성 가우스'입니다."
- **myrag4.py** 로 정리

• d.

- 찾은 커널 충돌로 인한 문제 → **python** 파일로 실행 → 터미널에서 직접 확인하기
- **eval_script.py** 실행 → 터미널에서 결과 바로 확인

RAG_EVAL_K7-3a35bd45 🔗 🕒 202507.../35e940 📄 🔍 Compact 🔍 Full 🔍 Diff 🔍 Default 🔍 📄 Group by 🔍 🔍 Display 🔍 🔍 Compare									
Inputs	Reference Outputs	Outputs	Correctness 0.60 AVG 📄	Latency 1.163 P50 🔍	Completed	Tokens 8,911.00	Cost 0.00		
코히어의 데이터 출처 탐색기에 ... #7184 →	코히어의 데이터 출처 탐색기는 AI 모델 훈련에 사용	코히어는 12개 기관과 함께 '데이터 출처 탐색기(D	correct	1.16s	COMPLETED 1	1,799	\$0.00		
미국 바이든 대통령이 안전하고... #8d34 →	2023년 10월 30일 미국 바이든 대통령이 행정명'	2023년 10월 30일 🔍	correct	1.07s	COMPLETED 1	1,832	\$0.00		
삼성전자가 만든 생성형 AI의 ... #e0d6 →	삼성전자가 만든 생성형 AI의 이름은 삼성 가우스'로	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	correct	0.94s	COMPLETED 1	1,751	\$0.00		
삼성전자가 만든 생성형 AI의 ... #7080 →	삼성전자가 만든 생성형 AI의 이름은 테디노트 입니	삼성전자가 만든 생성형 AI의 이름은 '삼성 가우스'	incorrect	1.63s	COMPLETED 1	1,751	\$0.00		
구글이 테디노트에게 20억달러... #b859 →	사실이 아닙니다. 구글은 엔스토크에 최대 20억 달	죄송합니다. 제공된 문서에서 해당 정보를 찾을 수	incorrect	1.25s	COMPLETED 1	1,778	\$0.00		

• 현재 상황 분석

항목	개수	정확도
CORRECT	3/5	60%
INCORRECT	2/5	40%

• 진전

- 이전: 1/5 (20%)
- 현재: 3/5 (60%)
- 향상: +200%!

• RAG 평가

- 어려움

이유	설명
초기 정확도	20-40%는 정상
반복 개선	60% → 80% → 90%로 점진적 향상
완벽은 없음	100%는 거의 불가능

*

• 현재 단계

일반적인 RAG 평가 발전:

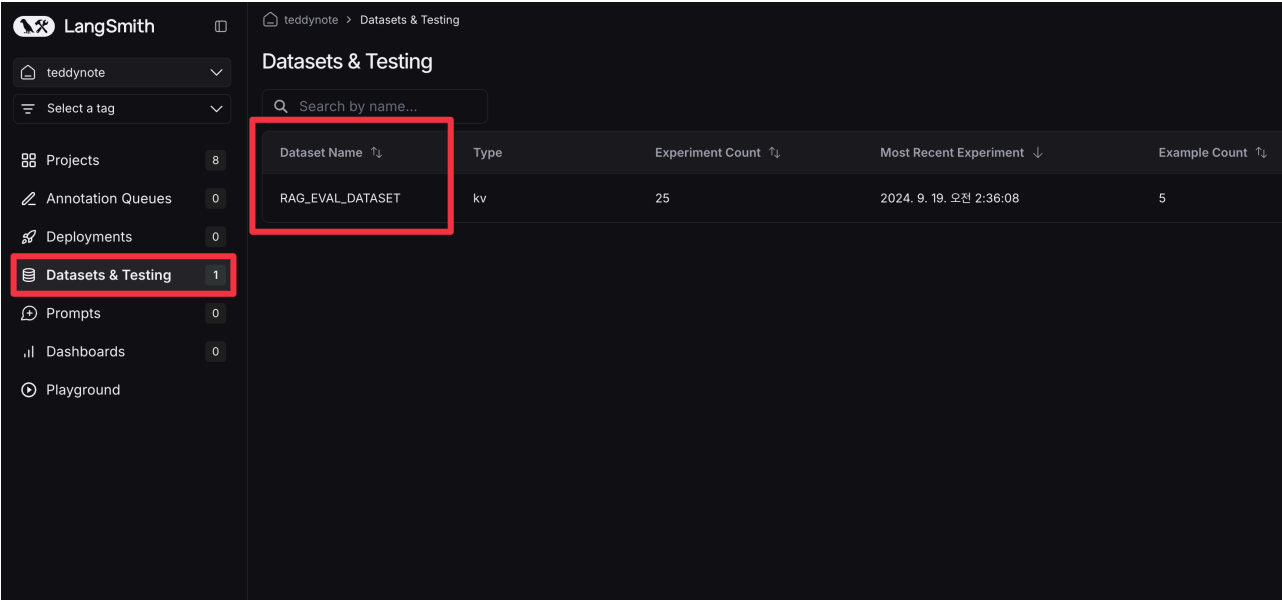
- 1단계: 20-40% (초기)
- 2단계: 50-60% (개선 후) ← 여기!
- 3단계: 70-80% (최적화)
- 4단계: 80-90% (완성)

• 향상 방법

- a. k값 증가
- b. chunk_size 조정하기
- c. 프롬프트 개선하기
- d. 다른 임베딩 모델 시도하기

3) LangSmith 테스트를 위한 Dataset 생성

• Dataset & Testing 에 새로운 데이터셋 생성하기



- csv 파일에서 LangSmith UI 사용 → 직접 데이터셋 생성 가능

- 참고: [LangSmith UI 문서](#)

```
from langsmith import Client

client = Client()
dataset_name = "RAG_EVAL_DATASET"
```

데이터셋 생성 함수

```
def create_dataset(client, dataset_name, description=None):
    for dataset in client.list_datasets():
        if dataset.name == dataset_name:
            return dataset

    dataset = client.create_dataset(
        dataset_name=dataset_name,
        description=description,
    )
    return dataset
```

```
# 데이터셋 생성
dataset = create_dataset(client, dataset_name)          # 2.9s
```

생성된 데이터셋에 예제 추가

```
client.create_examples(
    inputs=[{"question": q} for q in df["question"].tolist()],
    outputs=[{"answer": a} for a in df["answer"].tolist()],
    dataset_id=dataset.id,
)
```

- `client.create_examples()` 추가하기 - (0.1s)

```
{'example_ids': ['8d34ad4e-e3be-47ca-9a6a-9929608b60ca',
'71841ffc-cfbe-4e05-b0b0-ef85e8a6ebd4',
'e0d645a9-6d97-41da-8d45-cc5e8d6f0a37'],
'count': 3}
```

- 데이터셋에 예제 나중에 추가 가능

```
# 새로운 질문 목록
new_questions = [
    "삼성전자가 만든 생성형 AI의 이름은 무엇인가요?",
    "구글이 테디노트에게 20억달러를 투자한 것이 사실입니까?",
]
```

```
# 새로운 답변 목록
new_answers = [
    "삼성전자가 만든 생성형 AI의 이름은 테디노트 입니다.",
    "사실이 아닙니다. 구글은 앤스로픽에 최대 20억 달러를 투자하기로 합의했으며, 이 중 5억 달러를 우선 투자하고 향후 15억 달러를 추가로 투자하기로 했습니다.",
]
```

UI에서 업데이트된 버전 확인

```
client.create_examples(
    inputs=[{"question": q} for q in new_questions],
    outputs=[{"answer": a} for a in new_answers],
    dataset_id=dataset.id,
)
```

- UI에서 업데이트된 버전 확인하기 - (0.1s)

```
{'example_ids': ['7080404f-878d-4e22-9256-70a5d1b86ab3',
'b8599d26-9990-4153-8a92-0dc779087f2c'],
'count': 2}
```

- ✓ 데이터셋 준비 완료됨