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

---

02. `RAGAS`를 활용한 `평가`

1) `RAGAS를 활용한 평가`

- 참고: RAGAS
  - `API v1`: API v1
  - `API v2`: API v2

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install -qU faiss-cpu ragas
```

① 기본설정

```
import os
import logging
import warnings

warnings.filterwarnings("ignore")
os.environ["TOKENIZERS_PARALLELISM"] = "false"          # 토크나이저 병렬 처리 경고 
os.environ["TQDM_DISABLE"] = "1"                        # 진행바 비활성화
logging.disable(logging.WARNING)                        # 파이썬 로깅 레벨 조정 (WA

from dotenv import load_dotenv
load_dotenv()

print("✅ 환경 설정 완료")
```

- ✅ 환경 설정 완료

② LangSmith 설정

```
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 "설정되

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API_KEY') and langc
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{langchain_tracing_v
    print(f"✅ LangSmith 프로젝트: '{langchain_project}'")
    print(f"✅ LangSmith API Key: {langchain_api_key_status}")
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.")
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다. 다음을 확인하세요:")
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있지 않습니다 (현재: '{la
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

- 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.
```

```
# 필요한 패키지 임포트

from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import PyMuPDFLoader
from langchain_community.vectorstores import FAISS
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_core.prompts import PromptTemplate
from langchain_openai import ChatOpenAI
from langchain_ollama import ChatOllama
from langchain_huggingface import HuggingFaceEmbeddings

print("✅ 패키지 임포트 완료")
```

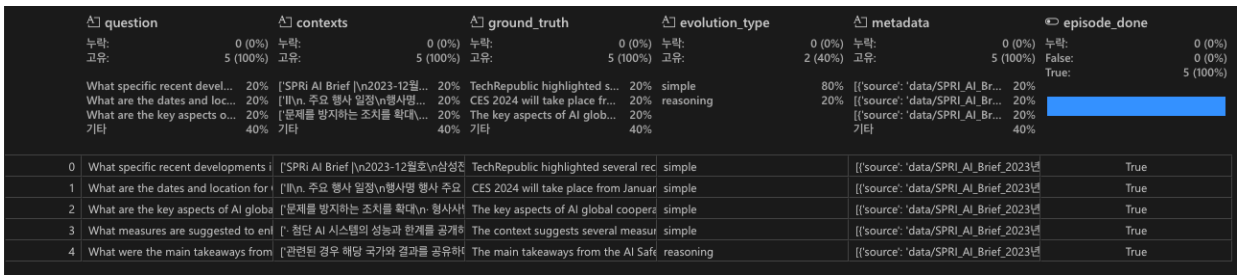
- ✅ 패키지 임포트 완료 - (`0.6s`)

---

③ 교재 원본 파일로 로드해보기

- **저장한 CSV 파일로부터 로드하기**
  - `data/ragas_synthetic_dataset_original.csv` 파일 로드하기

```
import pandas as pd

# 데이터 로드하기
df = pd.read_csv("../15_Evaluations/data/ragas_synthetic_dataset_original.csv")

# 일부 정보 읽어보기
df.head()
```

- **`df.head()`** - (`0.7s`)
  - 

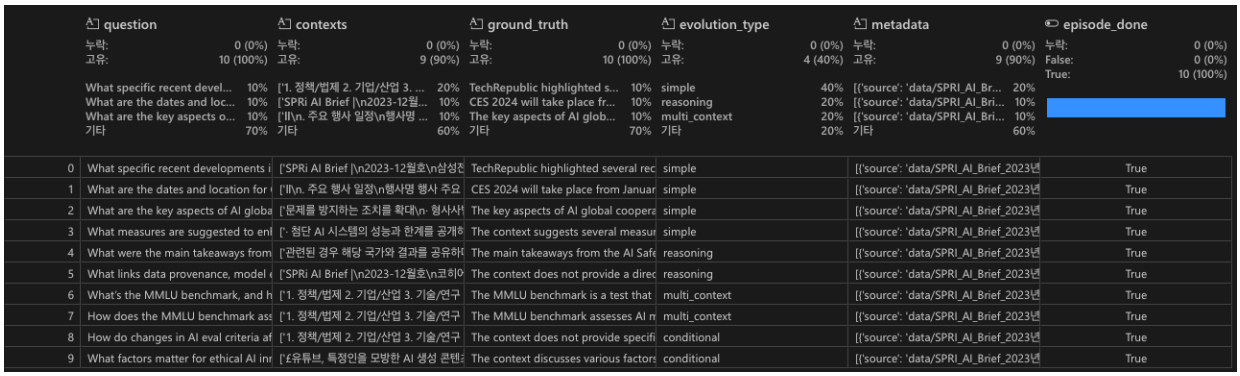
	question	contexts	ground_truth	evolution_type	metadata	episode_done
	누락: 0 (0%) 고유: 5 (100%)	누락: 0 (0%) 고유: 5 (100%)	누락: 0 (0%) 고유: 5 (100%)	누락: 0 (0%) 고유: 2 (40%)	누락: 0 (0%) 고유: 5 (100%)	누락: 0 (0%) False: 0 (0%) True: 5 (100%)
	What specific recent devel... 20% What are the dates and loc... 20% What are the key aspects o... 20% 기타 40%	['SPRi AI Brief \n2023-12월... 20% ['II\n. 주요 행사 일정\n행사명... 20% ['문제를 방지하는 조치를 확대\... 20% 기타 40%	TechRepublic highlighted s... 20% CES 2024 will take place fr... 20% The key aspects of AI glob... 20% 기타 40%	simple 80% reasoning 20%	[{'source': 'data/SPRI_AI_Br... 20% [{'source': 'data/SPRI_AI_Br... 20% [{'source': 'data/SPRI_AI_Br... 20% 기타 40%	
0	What specific recent developments i	['SPRi AI Brief \n2023-12월호\n삼성전	TechRepublic highlighted several rec	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
1	What are the dates and location for	['II\n. 주요 행사 일정\n행사명 행사 주요	CES 2024 will take place from Januar	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
2	What are the key aspects of AI globa	['문제를 방지하는 조치를 확대\n· 형사사	The key aspects of AI global cooperat	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
3	What measures are suggested to enl	['· 첨단 AI 시스템의 성능과 한계를 공개하	The context suggests several measur	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
4	What were the main takeaways from	['관련된 경우 해당 국가와 결과를 공유하도	The main takeaways from the AI Safe	reasoning	[{'source': 'data/SPRI_AI_Brief_2023년	True

```
from datasets import Dataset

test_dataset = Dataset.from_pandas(df)

test_dataset
```

- **`test_dataset`** - (`0.4s`)

```
Dataset({
    features: ['question', 'contexts', 'ground_truth', 'evolution_type',
    num_rows: 10
    })
```

- `test_dataset` 확인해보기



	question	contexts	ground_truth	evolution_type	metadata	episode_done
	누락: 0 (0%) 고유: 10 (100%)	누락: 0 (0%) 고유: 9 (90%)	누락: 0 (0%) 고유: 10 (100%)	누락: 0 (0%) 고유: 4 (40%)	누락: 0 (0%) 고유: 9 (90%)	누락: 0 (0%) False: 0 (0%) True: 10 (100%)
	What specific recent devel... 10% What are the dates and loc... 10% What are the key aspects o... 10% 기타 70%	['1. 정책/법제 2. 기업/산업 3. ... 20% ['SPRi AI Brief \n2023-12월... 10% ['II\n. 주요 행사 일정\n행사명 ... 10% 기타 60%	TechRepublic highlighted s... 10% CES 2024 will take place fr... 10% The key aspects of AI glob... 10% 기타 70%	simple 40% reasoning 20% multi_context 20% 기타 20%	[{'source': 'data/SPRI_AI_Br... 20% [{'source': 'data/SPRI_AI_Bri... 10% [{'source': 'data/SPRI_AI_Bri... 10% 기타 60%	
0	What specific recent developments i	['SPRi AI Brief \n2023-12월호\n삼성전	TechRepublic highlighted several rec	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
1	What are the dates and location for	['II\n. 주요 행사 일정\n행사명 행사 주요	CES 2024 will take place from Januar	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
2	What are the key aspects of AI globa	['문제를 방지하는 조치를 확대\n· 형사사	The key aspects of AI global coopera	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
3	What measures are suggested to enl	['· 첨단 AI 시스템의 성능과 한계를 공개하	The context suggests several measur	simple	[{'source': 'data/SPRI_AI_Brief_2023년	True
4	What were the main takeaways from	['관련된 경우 해당 국가와 결과를 공유하	The main takeaways from the AI Safe	reasoning	[{'source': 'data/SPRI_AI_Brief_2023년	True
5	What links data provenance, model	['SPRi AI Brief \n2023-12월호\n코히어	The context does not provide a direc	reasoning	[{'source': 'data/SPRI_AI_Brief_2023년	True
6	What's the MMLU benchmark, and h	['1. 정책/법제 2. 기업/산업 3. 기술/연구	The MMLU benchmark is a test that	multi_context	[{'source': 'data/SPRI_AI_Brief_2023년	True
7	How does the MMLU benchmark ass	['1. 정책/법제 2. 기업/산업 3. 기술/연구	The MMLU benchmark assesses AI m	multi_context	[{'source': 'data/SPRI_AI_Brief_2023년	True
8	How do changes in AI eval criteria af	['1. 정책/법제 2. 기업/산업 3. 기술/연구	The context does not provide specifi	conditional	[{'source': 'data/SPRI_AI_Brief_2023년	True
9	What factors matter for ethical AI in	['£유튜브, 특정인을 모방한 AI 생성 콘텐츠	The context discusses various factors	conditional	[{'source': 'data/SPRI_AI_Brief_2023년	True

```
import ast                                  # 문자열을 파이썬 리터럴로 안전하게 평가하는

# 주어진 예제 딕셔너리의 'contexts' 키에 저장된 문자열을 파이썬 리스트로 변환하는 함수
def convert_to_list(example):
    # 문자열 = 파이썬 리터럴로 변환함
    contexts = ast.literal_eval(example["contexts"])
    # 변환된 리스트 = 딕셔너리로 반환함
    return {"contexts": contexts}

# test_dataset의 각 요소에 convert_to_list 함수 적용하기
# map 함수 = 데이터셋의 각 행에 함수를 적용하는 데 사용
test_dataset = test_dataset.map(convert_to_list)

# 변환된 데이터셋을 출력하기
print(test_dataset)
```

- **변환된 데이터셋** - (`0.1s`)

```
Map:   100%|██████████████████████████████████████████| 10/10 [00:00<00:00,
```

```
Dataset({
    features: ['question', 'contexts', 'ground_truth', 'evolution_type',
    num_rows: 10
})
```

```
test_dataset[1]["contexts"]
```

```
['II\n. 주요 행사 일정\n행사명 행사 주요 개요\n– 미국 소비자기술 협회(CTA)가 주관하는 세계 최대 가전·I
```

```
# 단계 1: 문서 로드(Load Documents)
loader = PyMuPDFLoader("../15_Evaluations/data/SPRI_AI_Brief_2023년12월호_F.pdf"
docs = loader.load()                                                    # 0.3s
```

```
# 단계 2: 문서 분할(Split Documents)
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=
split_documents = text_splitter.split_documents(docs)
```

```
# 단계 3: 임베딩(Embedding) 생성
embeddings = HuggingFaceEmbeddings(
    model_name="BAAI/bge-small-en-v1.5",
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
)

print("✅ 임베딩 생성 완료")
```

- ✅ 임베딩 생성 완료 - (`8.8s`)

```
# 단계 4: DB 생성(Create DB) 및 저장
# 벡터스토어 생성하기

vectorstore = FAISS.from_documents(
    documents=split_documents,
    embedding=embeddings
    )

print("✅ vectorstore 생성 완료")
```

- ✅ vectorstore 생성 완료 - (`5.1s`)

```
# 단계 5: 검색기(Retriever) 생성
# 문서에 포함되어 있는 정보 → 검색 및 생성
retriever = vectorstore.as_retriever()

print("✅ retriever 생성 완료")
```

```
# 단계 6: 프롬프트 생성(Create Prompt)
# 프롬프트 생성하기

prompt = PromptTemplate.from_template(
    """You are an assistant for question-answering tasks.
Use the following pieces of retrieved context to answer the question.
If you don't know the answer, just say that you don't know.

#Context:
{context}

#Question:
{question}
```

```
#Answer:"""
)

print("✅ prompt 생성 완료")
```

- ✅ prompt 생성 완료

```
# 단계 7: 언어모델(LLM) 생성
llm=ChatOllama(
    model="llama3.2:3b",
    temperature=0.1,
)

print("✅ Ollama LLM 초기화 완료 (모델: llama3.2:3b)")
```

- ✅ Ollama LLM 초기화 완료 (모델: llama3.2:3b)

```
# 단계 8: 체인(Chain) 생성

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()
)

print("✅ Chain 생성 완료")
```

- ✅ Chain 생성 완료

- **배치 데이터셋** 생성하기
  - **다량의 질문을 한 번에 처리할 때 용이함**
  - 배치 : (`https://wikidocs.net/233345`)

```
# 배치 데이터셋 생성하기

batch_dataset = [question for question in test_dataset["question"]]

batch_dataset[:3]
```

- **`batch_dataset[:3]`**

```
    ['What specific recent developments in generative AI have been highlighted
    'What are the dates and location for CES 2024?',
```

```
    "What are the key aspects of AI global cooperation highlighted in the G7's
```

```
print(type(batch_dataset))              # <class 'list'>
```

```
print(batch_dataset)
```

- **`print(batch_dataset)`**

```
    ['What specific recent developments in generative AI have been highlighted
```

- before
- `batch_dataset[:3]`

```
['What specific recent developments in generative AI have been highlighted by
'What are the dates and location for CES 2024?',
"What are the key aspects of AI global cooperation highlighted in the G7's app
```

- `batch_dataset`

```
['What specific recent developments in generative AI have been highlighted by
```

- **`batch()` 호출 → 배치 데이터셋에 대한 답변 얻기**

```
answer = chain.batch(batch_dataset)

answer[:3]
```

- **`try_4`** - (`5m 49.3s`)

```
['According to the provided context, TechRepublic has highlighted that Sar
'According to the retrieved context, the dates and location for CES 2024 a
'The key aspects of AI global cooperation highlighted in the G7\'s approac
```

- **`try_1`** - (`5m 56.1s`)

```
['According to the provided context, TechRepublic has highlighted that Samsung
'According to the retrieved context, the dates and location for CES 2024 are:\
"The key aspects of AI global cooperation highlighted in the G7's approach to
```

- **`try_2`** - (`5m 36.0s`)
  - `request_timeout=None,` , `format="json"` 옵션 추가했을 때의 답변

```
['{"name": "Samsung Gauss", "text": "TechRepublic reported that Samsung Re
'{\n"기간": "2024.1.9~12",\n"장소": "미국, 라스베가스",\n"홈페이지": "https://www.
'{\n"위험 평가 및 완화": "G7은 AI 수명주기 전반에 걸쳐 위험을 평가 및 완화하는 조치를 채택하
```

- **`try_3`** - (`5m 40.2s`)

```
['According to the retrieved context, TechRepublic has highlighted that Samsur
'According to the retrieved context, the dates and location for CES 2024 are:\
'The key aspects of AI global cooperation highlighted in the G7\'s approach to
```

```
print(type(answer))                    # <class 'list'>
```

```
print(answer)
```

- **`print(answer)`**

```
['According to the provided context, TechRepublic has highlighted that Sar
```

- before
- `print(answer)`

```
['According to the retrieved context, TechRepublic has highlighted that Samsur
```

- **`LLM`이 생성한 답변 = `answer` 컬럼에 저장하기**

```
# 'answer' 컬럼 덮어쓰기 또는 추가

if "answer" in test_dataset.column_names:
    test_dataset = test_dataset.remove_columns(["answer"]).add_column("answer"

else:
    test_dataset = test_dataset.add_column("answer", answer)
```

```
print(test_dataset)
```

- **`print(test_dataset)`**

```
Dataset({
    features: ['question', 'contexts', 'ground_truth', 'evolution_type',
    num_rows: 10
})
```

```
print(type(test_dataset))                    # <class 'datasets.arrow_datas
```

---

2) 답변 평가

① Context Recall

- 검색된 context가 LLM이 생성한 답변과 얼마나 일치하는지를 측정
  - question, ground truth 및 검색된 context → 계산
  - 값 = 0 ~ 1 사이 → 높을수록 더 나은 성능
- $$\text{context recall} = \frac{|\text{GT claims that can be attributed to context}|}{|\text{Number of claims in GT}|}$$
  - Ground truth 답변에서 context recall을 추정하기 위해, ground truth 답변의 각 주장이 검색된 context에 귀속될 수 있는지 분석
  - 이상적인 시나리오: ground truth 답변의 모든 주장이 검색된 context에 귀속될 수 있어야 함

② Context Precision

- **contexts** 내의 **ground-truth** 관련 항목들이 상위 순위에 있는지를 평가하는 지표
  - 이상적: 모든 관련 chunks = 상위 순위에 나타나야 함
  - question, ground_truth, contexts → 계산
  - 값 = **0 ~ 1** 사이 → 높은 점수일수록 더 나은 정밀도
- 게산식
  - Context Precision@K의 계산식
  - $$\text{Context Precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top K results}}$$

*

- Precision@k의 계산식
- $$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$
  - k = **contexta**의 총 **chunk** 수

- `$v_k \in \{0, 1\}$` = 순위 k에서의 관련성 지표

- 정보 검색 시스템에서 검색된 컨텍스트의 품질을 평가하는 데 사용
  - 관련 정보가 얼마나 `정확하게 상위 순위`에 `배치`되었는지를 측정 → 시스템의 성능 판단

`③ Answer Relevancy`

- **`생성된 답변`이 주어진 `prompt`에 `얼마나 적절한지`를 `평가`하는 지표**
- 주요 특징
  - 목적: 생성된 `답변`의 `관련성 평가`
  - 점수 해석:
    - 낮은 점수 = `불완전`하거나 `중복 정보`를 `포함`한 답변
    - 높은 점수 = `더 나은 관련성`을 나타냄
  - 계산에 사용되는 요소: `question`, `context`, `answer`

- 게산 방법
  - 원래 `question`과 `answer`를 기반으로 생성된 `인공적인 질문들 간`의 `평균 코사인 유사도`로 정의
  - 수식 ①
    - $$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N}\cos(E_{g_i}, E_o)$$
  - 수식 ②
    - $$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N}\frac{E_{g_i}\cdot E_o}{|E_{g_i}||E_o|}$$
  - 참고
    - `$E_o$` = 생성된 질문 `i`의 임베딩
    - `$E_{g_i}$` (E_{g_i}) = 원래 질문의 임베딩
    - `$N$` = 원래 질문의 임베딩 (*기본값 = 3*)
    - 주의사항: 실제로는 점수가 대부분 0과 1 사이 → 코사인 유사도의 특성상 수학적으로 -1에서 1 사이의 값을 가질 수도 있음

- 질문-답변 시스템의 성능을 평가하는 데 유용

- 생성된 답변이 원래 질문의 의도를 얼마나 잘 반영하는지를 측정

④ Faithfulness

- **생성된 답변의 사실적 일관성을 주어진 컨텍스트와 비교하여 측정하는 지표**
- 주요 특징
  - 목적: 답변의 사실적 일관성을 컨텍스트와 비교 → 평가
  - 계산 요소: 답변과 검색된 컨텍스트 사용
  - 점수 범위: `0 ~ 1 사이`로 조정 → **높을수록 더 좋음**
- 계산 방법
  - 수식
  - $$\text{Faithfulness score} = \frac{|\text{Number of claims in the generated answer that can be inferred from given context}|}{|\text{Total number of claims in the generated answer}|}$$

*

- 계산 과정 ①: 생성된 답변에서 주장(claims)들을 식별
- 계산 과정 ②: 각 주장을 주어진 컨텍스트와 대조 검증 → 컨텍스트에서 추론 가능한지 확인
- 계산 과정 ③: 위 수식을 사용하여 점수 계산

*

- 예시
  - 질문: **"아인슈타인은 어디서, 언제 태어났나요?"**
  - 컨텍스트: **"알버트 아인슈타인(1879년 3월 14일 출생)은 독일 출신의 이론 물리학자로, 역사상 가장 위대하고 영향력 있는 과학자 중 한 명으로 여겨집니다."**
  - **높은** 충실도 답변: **"아인슈타인은 1879년 3월 14일 독일에서 태어났습니다."**
  - *낮은* 충실도 답변: **"*아인슈타인은 1879년 3월 20일 독일에서 태어났습니다*."**
- 생성된 답변이 주어진 컨텍스트에 얼마나 충실한지를 평가하는 데 유용
  - 특히 질문-답변 시스템의 정확성과 신뢰성을 측정하는 데 중요

```
from ragas import evaluate

from ragas.metrics import (
    answer_relevancy,
    faithfulness,
```

```
    context_recall,
    context_precision,
)

print("✅ 모험 평가 함수 생성 완료")


# ------------------------------------------------------------
# 2. 모든 Ragas Metrics에 LangChain LLM 객체 'llm' 직접 주입
# ------------------------------------------------------------

from langchain_google_genai import ChatGoogleGenerativeAI

from dotenv import load_dotenv
import os

# API 키 확인
if not os.getenv("GOOGLE_API_KEY"):
    os.environ["GOOGLE_API_KEY"] = input("Enter your Google API key: ")

# gemini LLM 정의
llm2 = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0,
    google_api_key=os.getenv("GOOGLE_API_KEY")
    )

print("✅ gemini Critic LLM 정의 완료. (모델: gemini-2.5-flash-lite) ")

result = evaluate(
    dataset=test_dataset,
    metrics=[
        context_precision,
        faithfulness,
        answer_relevancy,
        context_recall,
    ],
    llm=llm2,
)

print()
print("="*60)
print("✅ 평가 완료!")
print("="*60)
print()
```

- ✅ 모험 평가 함수 생성 완료 - (`0.2s`)
- ✅ Ollama Critic LLM 정의 완료. (모델: llama3.2:3b, JSON 출력 강제)
- ✅ Ragas Metrics에 LLM 주입 완료. 이제 평가를 시작할 준비가 되었습니다.

- ✅ 모험 평가 함수 생성 완료 - (`0.5s`)
- ✅ gemini Critic LLM 정의 완료. (모델: gemini-2.5-flash-lite)

- ✅ Ragas Metrics에 LLM 주입 완료. 이제 평가를 시작할 준비가 되었습니다.

```
# ------------------------------------------------------------
# 데이터셋 첫번째 로우 확인하기
# ------------------------------------------------------------

try:
    print("\n--- 1. 데이터셋 전체 컬럼 이름 확인 ---")

    # 데이터셋의 실제 컬럼 이름 출력
    actual_columns = test_dataset.column_names
    print(f"실제 컬럼 목록: {actual_columns}")

    # Ragas 필수 컬럼의 포함 여부 확인
    required_cols = ['question', 'answer', 'contexts', 'ground_truth']
    missing_cols = [col for col in required_cols if col not in actual_columns]

    if missing_cols:
        print(f"\n❌ [치명적 오류]: Ragas 필수 컬럼 중 다음이 누락되었습니다: {missing_co
    else:
        print("\n✅ [컬럼 이름 확인]: Ragas 필수 컬럼이 모두 존재합니다.")

    print("\n--- 2. 첫 번째 로우의 값 및 타입 확인 (가장 중요) ---")
    first_row = test_dataset[0]

    for col in required_cols:
        value = first_row.get(col)
        value_type = type(value)

        # contexts 컬럼에 대한 특별 검사
        if col == 'contexts':
            # contexts가 리스트인지 확인
            if value_type is list:
                # 리스트의 첫 번째 요소가 문자열인지 확인
                inner_type = type(value[0]) if value and len(value) > 0 else '
                print(f"  - {col} (타입: {value_type}, 내부 타입: {inner_type})")
                if inner_type != str:
                    print("    🚨 경고: contexts 내부 요소는 반드시 문자열(str)이어야 합
            else:
                print(f"  - {col} (타입: {value_type})")
                print("    ❌ 치명적 오류: contexts 컬럼은 반드시 리스트(list)여야 합니다
        else:
            # question, answer, ground_truth는 문자열인지 확인
            print(f"  - {col} (타입: {value_type})")
            if value_type is not str:
                print(f"    🚨 경고: {col} 컬럼은 반드시 문자열(str)이어야 합니다.")

except NameError:
    print("\n❌ 오류: 'test_dataset'이 정의되지 않았습니다. 데이터셋 로드를 먼저 실행해 주세요
except Exception as e:
    print(f"\n❌ 진단 중 예상치 못한 오류 발생:")
    print(e)
```

```
# ------------------------------------------------------------
# 3. 평가 실행
```

```
# ------------------------------------------------------------

print("평가 점수:")
for metric, score in result.items():
    print(f"  {metric}: {score:.4f}")

# DataFrame 변환
result_df = result.to_pandas()
print()
print("결과 DataFrame:")
print(result_df.head())

# 저장
result_df.to_csv("../15_Evaluations/data/ragas_evaluation_result_2.csv", inde
print()
```

- `try_1` → ❌ - (`9m 1.7s`)

```
Evaluating:   100%|██████████████████████████████████████| 40/40 [09:00<
```

```
{'context_precision': 1.0000, 'faithfulness': nan, 'answer_relevancy': nan
```

```
# 데이터프레임 형태로 확인하여 nan 발생 여부 확인
result_df = result.to_pandas()
result_df.head()
```

```
result_df.loc[:, "context_precision":"context_recall"]
```

- next: ***`03. 생성한 평가용 데이터셋 업로드 (HuggingFace Dataset)`***
