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

---

`CH15.` `평가` *(Evaluations)*

- `LLM` *(Large Language Model)* 평가: 인공지능 `언어 모델의 성능`, `정확성`, `일관성` 및 기타 중요한 측면을 `측정`하고 `분석`하는 과정
- 모델의 개선, 비교, 선택 및 응용 프로그램에 적합한 모델 결정에 필수적인 단계

- `평가 방법`
  - `자동화된 메트릭`: `BLEU`, `ROUGE`, `METEOR`, `SemScore` 등의 지표 사용
  - `인간 평가`: 전문가 or 크라우드 소싱 통한 직접적 평가 → 수행
  - `작업 기반 평가`: 특정 작업에서의 성능을 측정함
  - `LLM-as-judge`: 다른 `LLM`을 평가자로 사용하는 방법

- `LangChain에서의 Evaluation`
  - *`LangChain`은 `LLM`의 애플리케이션의 평가를 위한 다양한 도구와 프레임워크를 제공함*
    - `모듈화된 평가 컴포넌트`: 다양한 평가방법 쉽게 구현 및 조합 가능
    - `Chain 평가`: 전체 `LLM` 애플리케이션 파이프라인 평가
    - `데이터셋 기반 평가`: 사용자 정의 데이터셋 사용 → 모델 평가
    - `평가 자표`: 정확성, 일관성, 관련성 등 다양한 지료 제공

- `LLM-as-judge`
  - `LLM-as-judge`: 다른 `LLM`의 출력을 평가하기 위해 `LLM`을 사용하는 혁신적인 접근 방식
    - `자동화`: 인간의 개입 없이 `대규모 평가 수행 가능`

        - **일관성**: 평가 기준 → 일관되게 적용할 수 있음
        - **유연성**: 다양한 평가 기준, 상황에 적응 가능
        - **비용 효율성**: 인간 평가자에 비해 비용이 적게 들 수 있음

*

- **LLM-as-judge의 작동 방식**
    - **입력 제공**: 평가할 LLM의 출력 • 평가 기준 제공
    - **분석**: 평가자 LLM이 제공된 출력을 분석
    - **평가**: 정의된 기준에 따라 점수 or 피드백 생성
    - **결과 집계**: 여러 평가 결과 종합 → 최종 평가 도출

- **장단점**
    - **장점**
        - 대규모 평가 기능
        - 빠른 피드백 루프
        - 다양한 평가 기준 적용 가능
    - **단점**
        - 평가자 LLM의 편향 가능성
        - 복잡 or 미묘한 평가에 한계가 있을 수 있음
        - 평가자 LLM의 성능에 의존적

- **평가의 중요성**
    - **모델 개선**: 약점 식별 → 개선 방향 제시함
    - **신뢰성 확보**: 모델의 성능 • 한계 이해 → 도움을 줌
    - **적합한 모델 선택**: 특정 직업 or 도메인에 가장 적합한 모델을 선택 가능
    - **윤리적 고려사항**: 편향, 공정성 등의 윤리적 측면 평가 가능

---

1. 합성 테스트 데이터셋 생성 *(RAGAS)*

1) 합성 테스트 데이터셋 생성

- **왜 합성 테스트 데이터(Synthetic Test Dataset)인가?**
  - RAG *(검색 증강 생성)` 증강 파이프라인의 성능을 평가하는 것은 매우 중요
  - 어려움
    - 문서에서 수백 개의 QA (질문-문맥-응답) 샘플을 수동으로 생성하는 것 = **시간과 노동력이 많이 소요**
    - 사람이 만든 질문 = 철저한 평가에 필요한 복잡성 수준에 도달하기 어려움 = 궁극적으로 평가의 품질에 영향을 미칠 수 있음
  - **합성 데이터 생성을 사용** → 데이터 집계 프로세스에서 개발자의 시간을 **90%** 까지 **감소 가능**
  - 참고: RAGAS

- 사전에 VS Code 터미널에 설치할 것

```
pip install -qU ragas
```

- 오류 발생 → 커널 및 패키지 충돌
- 트러블 슈팅 문서 참고
  - RAGAS_Synthetic_Dataset_Generation_Version_Conflicts_Troubleshooting

---

- **환경설정**

```
# 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

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

```
# ========================================
# ragas 없이 Gemini로 직접 생성
# ========================================
import os
os.environ['TOKENIZERS_PARALLELISM'] = 'false'

from dotenv import load_dotenv
load_dotenv()

from langchain_community.document_loaders import PDFPlumberLoader
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import JsonOutputParser
import pandas as pd
print("✅ 임포트 완료")
```

- **✅ 임포트 완료**

---

2) 문서 전처리

- 실습에 활용할 문서
  - 소프트웨어정책연구소 (SPRi) - 2023년 12월호

    - 저자: 유재흥(AI정책연구실 책임연구원), 이지수(AI정책연구실 위촉연구원)
    - 참고: 링크
    - 파일명: SPRI_AI_Brief_2023년12월호_F.pdf

- **문서 로드하기**

```
# ========================================
# 1. 문서 로드
# ========================================
loader = PDFPlumberLoader("../15_Evaluations/data/SPRI_AI_Brief_2023년12월호_F.pdf"
docs = loader.load()[3:-1]
docs = docs[:3]                          # 3개만 로드하기

print(f"✅ 문서 로드: {len(docs)}개")
```

- ✅ **문서 로드: 3개** - (`4.4.s`)

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

```
# 필터링
docs = [doc for doc in docs if doc.page_content.strip()]
print(f"✅ 유효 문서 수: {len(docs)}")
```

- ✅ **유효 문서 수: 3**

```
# --- docs 로드 여부(존재 유무) 검사 및 내용 필터링 로직 추가 ---

if not docs:
    # docs 리스트가 비어있다면(로드에 실패했다면) 치명적 오류 메시지 출력
    print("❌ 치명적 오류: 'docs' 리스트에 로드된 문서가 0개입니다. 이전 로드/저장 단계를 확인하세요

else:
    # --- [핵심 수정: 문서 내용 필터링 복구] ---
    original_doc_count = len(docs)
    # page_content가 비어 있지 않은 문서만 필터링합니다.
    docs = [doc for doc in docs if doc.page_content and doc.page_content.strip()]
    filtered_doc_count = len(docs)

    if original_doc_count > filtered_doc_count:
        print(f"⚠️ 경고: {original_doc_count - filtered_doc_count}개의 빈 문서가 제거되

    if filtered_doc_count == 0:
        print("❌ 치명적 오류: 유효한 텍스트를 가진 문서가 0개입니다. 웹 크롤링(`loader.py`) 단
    else:
        print(f"✅ docs 유효성 검사 완료: {filtered_doc_count}개의 문서가 사용 준비되었습니다
```

- ✅ docs 유효성 검사 완료: 3개의 문서가 사용 준비되었습니다.

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

```
# ========================================
# 2. Gemini 설정
# ========================================
llm = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0.7
)

print("✅ Gemini 설정 완료")
```

- ✅ **Gemini 설정 완료**

```
E0000 00:00:1760339825.688791 6315053 alts_credentials.cc:93] ALTS creds ignore
```

```
# ========================================
# 3. 프롬프트 (ragas 스타일)
# ========================================

prompt = ChatPromptTemplate.from_template(
    """당신은 AI 데이터셋 생성 전문가입니다.

다음 문서를 읽고 3개의 고품질 질문-답변 쌍을 생성하세요.

질문 유형:
1. simple: 문서에서 직접 답을 찾을 수 있는 간단한 질문
2. reasoning: 여러 정보를 종합해야 하는 추론 질문
3. multi_context: 전체 맥락을 이해해야 하는 복합 질문

문서:
{document}

JSON 형식으로 출력:
{{
  "questions": [
    {{
      "user_input": "질문 내용",
      "reference": "답변 내용 (문서 기반)",
      "reference_contexts": ["관련 문맥"],
      "synthesizer_name": "simple"
    }},
    {{
      "user_input": "질문 내용",
      "reference": "답변 내용 (문서 기반)",
      "reference_contexts": ["관련 문맥"],
      "synthesizer_name": "reasoning"
    }},
    {{
      "user_input": "질문 내용",
```

```
      "reference": "답변 내용 (문서 기반)",
      "reference_contexts": ["관련 문맥"],
      "synthesizer_name": "multi_context"
    }}
  ]
}}

중요: 반드시 유효한 JSON 형식으로만 출력하세요."""
)

chain = prompt | llm | JsonOutputParser()

print("✅ 프롬프트 설정 완료")
```

- ✅ 프롬프트 설정 완료

```
# ========================================
# 4. 질문 생성
# ========================================

print("⏳ 테스트셋 생성 중... (약 2~3분)")

all_questions = []

for i, doc in enumerate(docs):
    print(f"  문서 {i+1}/{len(docs)} 처리 중...")
    try:
        result = chain.invoke({"document": doc.page_content[:3000]})  # 3000자로 제
        if isinstance(result, dict) and "questions" in result:
            all_questions.extend(result["questions"])
        print(f"  ✅ 문서 {i+1} 완료 ({len(result.get('questions', []))}개 생성)")
    except Exception as e:
        print(f"  ⚠️ 문서 {i+1} 실패: {e}")
        continue

print(f"✅ 총 {len(all_questions)}개 질문 생성!")
```

- 테스트셋 생성 - (7.8s)

```
⏳ 테스트셋 생성 중... (약 2~3분)

문서 1/3 처리 중...

✅ 문서 1 완료 (3개 생성)

문서 2/3 처리 중...

✅ 문서 2 완료 (3개 생성)

문서 3/3 처리 중...
```

```
✅ 문서 3 완료 (3개 생성)

✅ 총 9개 질문 생성!
```

```python
# ========================================
# 5. DataFrame 생성 & 저장
# ========================================

test_df = pd.DataFrame(all_questions)

# 필수 컬럼 확인
if not test_df.empty:
    # reference_contexts가 리스트인지 확인
    test_df['reference_contexts'] = test_df['reference_contexts'].apply(
        lambda x: x if isinstance(x, list) else [str(x)]
    )

    # CSV 저장
    test_df.to_csv("data/ragas_synthetic_dataset.csv", index=False)

    print("✅ 저장 완료!")
    print("\n생성된 데이터셋:")
    print(test_df[['user_input', 'synthesizer_name']])
    print(f"\n✅ 총 {len(test_df)}개 질문 생성!")

else:
    print("⚠️ 데이터 생성 실패")
```

- **생성된 데이터셋** - (`0.2s`)

```
✅ 저장 완료!

생성된 데이터셋:
                                          user_input synthesizer_name
0  미국 바이든 대통령은 안전하고 신뢰할 수 있는 AI 개발과 사용에 관한 행정명령을 ...           sin
1  AI의 안전과 보안 기준 마련과 관련하여, 미국 정부에 정보 공유를 요구받는 컴퓨팅...        reasoni
2  미국 행정명령은 AI가 사회 전반에 미치는 영향을 고려하여 어떤 분야에서 형평성과 ...    multi_con
3            G7이 AI 기업을 대상으로 마련한 국제 행동강령의 명칭은 무엇인가요?           simpl
4    AI 국제 행동강령에서 AI 수명주기 전반에 걸쳐 요구하는 주요 조치들은 무엇인가요?        reasoni
5  G7의 히로시마 AI 프로세스가 출범하게 된 배경과 이 프로세스를 통해 궁극적으로 ...    multi_cont
6      영국 AI 안전성 정상회의에서 발표된 '블레츨리 선언'은 무엇을 강조하고 있나요?           sim
7  영국 AI 안전 연구소는 첨단 AI 모델의 안전 테스트에서 어떤 역할을 수행할 예정...        reason
8  AI 안전성 정상회의에서 발표된 '블레츨리 선언'과 영국 정부의 AI 안전 테스트 ...    multi_conte

✅ 총 9개 질문 생성!
```