

/\* elice \*/

Enterprise

# 출처 표시

RAG가 답변할 때, 근거를 함께 보여주는 방법

11:30 ~ 12:30

# 학습 범위 정리

## 여태까지 학습한 내용

- RAG 5단계 코드 구조
- 각 파라미터의 역할 이해
- 커스텀 프롬프트로 답변 품질 변화
- 프롬프트 규칙 아이디어 생성

## 학습할 내용

- 출처 표시가 왜 필수인가 (제조업)
- metadata가 파이프라인을 타는 원리
- return\_source\_documents 활용법
- 출처 정확성 검증 (라이브 데모)

# 왜 출처 표시가 필수인가?

제조업에서 생기는 일

"이 냉장고 정격 소비전력이 얼마예요?"

AI: 36W입니다.

→ 이 숫자가 사양서에서 온 건지,  
시험성적서에서 온 건지,

아니면 AI가 지어낸 건지 알 수 없습니다.



## 검증 가능성

답변 근거를 원문에서 직접 확인 가능



## 신뢰도 확보

R&D 보고서에 AI 답변을 인용할 수 있음



## 오류 추적

잘못된 답변의 원인을 역추적 가능

# 출처 없는 답변 vs 출처 있는 답변

---

## 출처 없는 답변

Q: RF9000 냉장실 용량은?

A: 냉장실 용량은 524L입니다.

**이 숫자가 맞는지 확인하려면?**

- 사양서를 직접 열어서 찾아봐야 함
- 문서가 10종이면? 100종이면?

## 출처 있는 답변

Q: RF9000 냉장실 용량은?

A: 냉장실 용량은 524L입니다.

**출처: 제품사양서\_RF9000.pdf**

페이지 2 | 청크 #3

 **바로 해당 페이지에서 확인 가능!**

# metadata가 파이프라인을 타는 원리



**핵심: Document 객체 안에 metadata가 함께 다닌다**

# Day2에서 만든 Document 객체를 다시 보면...

```
Document(  
    page_content = "RF9000의 냉장실 용량은 524L...",  
    metadata = {"source": "제품사양서_RF9000.pdf", "page": 1}  
)
```

# 출처 표시

## 자동 생성 (PyPDFLoader)

**source**

→ PDF 파일 경로

**page**

→ 페이지 번호 (0부터 시작)

## 직접 추가 가능 (커스텀)

**doc\_type**

→ "사양서" / "시험성적서"

**product**

→ "RF9000" / "냉장고"

```
# 커스텀 metadata 추가 ~ 오후 실습에서 직접 해봅니다
for doc in documents:
    doc.metadata["doc_type"] = "사양서"
    doc.metadata["product"] = "RF9000"
    doc.metadata["version"] = "v2.1"
```

# 출처를 돌려받는 코드

오전 2에서 본 RAG 함수에 한 줄만 추가하면 됩니다

## 코드 (출처 없음)

```
def qa_invoke(query):  
    ...  
    return {"result": answer}
```

## 코드 (출처 포함!)

```
def qa_invoke(query):  
    ...  
    return {"result": answer,  
            "source_documents": docs}
```

"source\_documents": docs ← 이 한 줄이 전부!

```
# 이제 결과에 source_documents가 포함됩니다  
result = qa_invoke("냉장실 용량은?")  
result["result"]      # → "524L입니다."  
result["source_documents"] # → [Document(...), ...]
```

# 출처 정보 추출하기

source\_documents 리스트에서 필요한 정보를 꺼내는 코드

```
# 출처 정보 추출 ~ 반복문으로 하나씩 꺼냅니다

for i, doc in enumerate(result["source_documents"]):
    source = doc.metadata["source"] # 파일명
    page = doc.metadata["page"] # 페이지
    text = doc.page_content[:100] # 미리보기

    print(f"[출처 {i+1}]")
    print(f" 문서: {source}")
    print(f" 페이지: {page + 1}")
```



**page + 1**

Python은 0부터  
세지만 사람은  
1부터 세니까요!

(Day1 인덱싱 복습)

# 사용자 친화적 출처 포맷팅

```
def format_response_with_sources(result):
    """답변 + 출처를 보기 좋게 포맷팅"""

    answer = result["result"]
    sources = result["source_documents"]
    output = f"답변:\n{answer}\n"
    output += "\n참고 문서:\n"
    for i, doc in enumerate(sources):
        name = doc.metadata["source"]
        pg = doc.metadata["page"] + 1
        output += f" [{i+1}] {name} (p.{pg})\n"
    return output
```

## 출력 결과



### 답변:

RF9000 냉장실 용량은 524L이며,  
냉동실 용량은 344L입니다.



### 참고 문서:

- [1] 제품사양서\_RF9000.pdf (p.2)
- [2] 제품사양서\_RF9000.pdf (p.3)



이 함수를 한 번 만들어 두면, 모든 질의에서 자동으로 출처가 표시됩니다

# PDF vs TXT – 페이지 정보의 차이

Day2에서 PDF를 선택한 이유가 여기 있습니다

## PDF (PyPDFLoader)

- ✓ source: "제품사양서\_RF9000.pdf"
- ✓ page: 1 (페이지 번호 자동!)

→ 출처에 페이지까지 표시 가능  
→ "사양서 2페이지에 있습니다"

## TXT (TextLoader)

- ✓ source: "문서.txt"
- ✗ page: 없음!

→ 파일명만 나오고 위치 모름  
→ "어디에 있는지는 직접 찾으세요"

제조업 문서는 PDF가 대부분 → PyPDFLoader로 페이지 정보를 자동 확보합니다

# 실전: 문서 2종을 함께 검색할 때

오후 실습에서 사양서 + 시험성적서를 함께 넣고 검색합니다

## 제품사양서

RF9000 (4페이지)  
용량, 크기, 소비전력,  
기능, 에러코드 등

+

## 시험성적서

RF9000 (3페이지)  
에너지효율, 소음,  
냉각 성능, 안전시험 등



## 답변+출처

출처에 어떤 문서에서  
왔는지가 구분되어  
표시됩니다

# 문서 2종을 넣고 질문하면 ~ 출처가 문서별로 구분됩니다

Q: "이 냉장고의 에너지 효율 등급과 소비전력을 알려주세요"

참고 문서:

[1] 제품사양서\_RF9000.pdf (p.2) ← 소비전력 정보

[2] 시험성적서\_RF9000.pdf (p.1) ← 에너지효율 등급

# 출처 표시 실무 팁 3가지

---

1

## 파일명을 사람이 읽기 좋게 변환

제품사양서\_스마트냉장고\_RF9000.pdf → '제품사양서 (스마트냉장고 RF9000)'

2

## 중복 출처 제거

같은 문서의 같은 페이지가 여러 번 나올 수 있으므로 set()으로 중복 제거

3

## 관련 텍스트 미리보기 제공

출처 아래에 page\_content 앞 100자를 보여주면 검증이 더 쉬움

## PREVIEW

---

# 출처 정확성 검증

라이브로 Jupyter에서 질문을 실행합니다

PDF를 열어서 출처를 대조해주세요

정말 p. 2에 이 내용이 있는지 함께 확인합니다

# 출처 표시 실무 팁 3가지

Q: "RF9000의 소비전력은 얼마인가요?"

A: RF9000의 소비전력은 36W입니다.

참고 문서:

- [1] 제품사양서\_스마트냉장고\_RF9000.pdf (p.2)
- [2] 제품사양서\_스마트냉장고\_RF9000.pdf (p.2)

## PDF 대조

사양서 p.2를 열어보면

**소비전력: 36W**

 정확합니다!

## 추가 질문 시도

"E3 에러코드는 무엇인가요?" → 에러코드표(p.4)에서 확인

"냉장실과 냉동실 용량 차이는?" → 계산이 필요한 질문 → 사양서(p.2)

## 데모 ②: 시험성적서 + 교차 참조

Q: "에너지효율 등급은?"

A: RF9000의 에너지소비효율 등급은 1등급입니다.

참고 문서:

- [1] 시험성적서\_RF9000.pdf (p.2) ← 측정값
- [2] 제품사양서\_RF9000.pdf (p.2) ← 사양표

교차 확인!

두 문서에서 동시에 찾음

시험성적서: 측정값 확인

사양서: 스펙 확인

→ 벡터DB에 두 문서 모두  
저장했기 때문에 가능!

RAG가 벡터DB에 있는 두 문서를 모두 검색하기 때문에 교차 확인이 가능합니다

# 데모 ③: 문서에 없는 질문

Q: "RF9000의 가격은 얼마인가요?"

A: 해당 정보를 제공된 문서에서 찾을 수 없습니다.  
제품사양서와 시험성적서에는 가격 정보가  
포함되어 있지 않습니다.

LLM만 사용하면?

"대략 200만원 정도 합니다"  
→ 근거 없이 지어냄 → 할루시네이션!

RAG를 사용하면?

"해당 정보를 찾을 수 없습니다"  
→ 문서에 없으면 솔직하게 → 방지!

할루시네이션 대응은 내일 Day4에서 더 깊이 다룹니다

# 전체 흐름: 출처 표시 RAG 코드 요약

```
# === 전체 흐름 (오후 실습에서 직접 작성할 코드) ===  
docs = PyPDFLoader("제품사양서_RF9000.pdf").load() # 1단계: 문서 로딩 (Day2 복습)  
chunks = text_splitter.split_documents(docs) # 2단계: 청킹 (Day2 복습)  
vectorstore = Chroma.from_documents(chunks, embeddings) # 3단계: 벡터DB 저장 (Day2 복습)  
  
# 4단계: RAG 체인 생성 (블록2 복습 + 블록3 NEW!)  
def qa_invoke(query):  
    docs = vectorstore.similarity_search(query, k=3)  
    context = "\n\n".join(doc.page_content for doc in docs)  
    prompt = f"문서 내용:\n{context}\n질문: {query}\n답변:"  
    answer = gemini_client.models.generate_content(  
        model=GEMINI_MODEL, contents=prompt).text # ← 오늘의 핵심!  
    return{"result": answer, "source_documents": docs}  
  
# 5단계: 질의 + 출처 확인 (블록3 NEW!)  
result = qa_invoke("냉장실 용량은?")  
ask_with_sources("냉장실 용량은?")
```

# 오전 정리

---

## 1 체험

RAG 완성품 데모 확인  
활용 아이디어 토론

## 2 이해

RAG 5단계 코드  
커스텀 프롬프트 효과

## 3 심화

출처 표시 원리 + 코드  
라이브 데모 검증

오전에 이해한 것을 오후에 직접 만듭니다

# 오후 실습 안내

Section 0	환경설정 (pip install)	빠르게	Day2 복습
Section 1	PDF 로딩 + 벡터DB 구축	빠르게	Day2 복습
Section 2	유사도 검색 테스트	빠르게	Day2 복습
Section 3	RAG 체인 구축	핵심	오후1
Section 4	커스텀 프롬프트 적용	핵심	오후2
Section 5	출처 표시 구현	핵심	오후2
Section 6	연습 과제	도전	오후3

"Section 0~2는 이미 학습한 내용으로 셀 실행만 진행"

## PREVIEW

---

# 점심시간

돌아오시면 한 가지만 부탁드립니다

노트북을 열고 Section 0의 첫 셀을 실행해주세요

(pip install — 설치에 2~3분 걸립니다)

12:30 ~ 13:30