

# **Day 2**

## **Advanced RAG Systems**

Retrieval-Augmented Generation 심화 과정

• • •

9개 실습으로 완성하는 RAG 마스터링

# 학습 로드맵

## 1 RAG 기초

- Naive RAG 구현
- 실패 패턴 분석
- 개선 방향 도출

## 2 쿼리 개선

- Multi-Query
- HyDE
- Step-Back

## 3 Hybrid Search

- Vector + BM25
- RRF Fusion
- Reranking

## 4 평가 및 튜닝

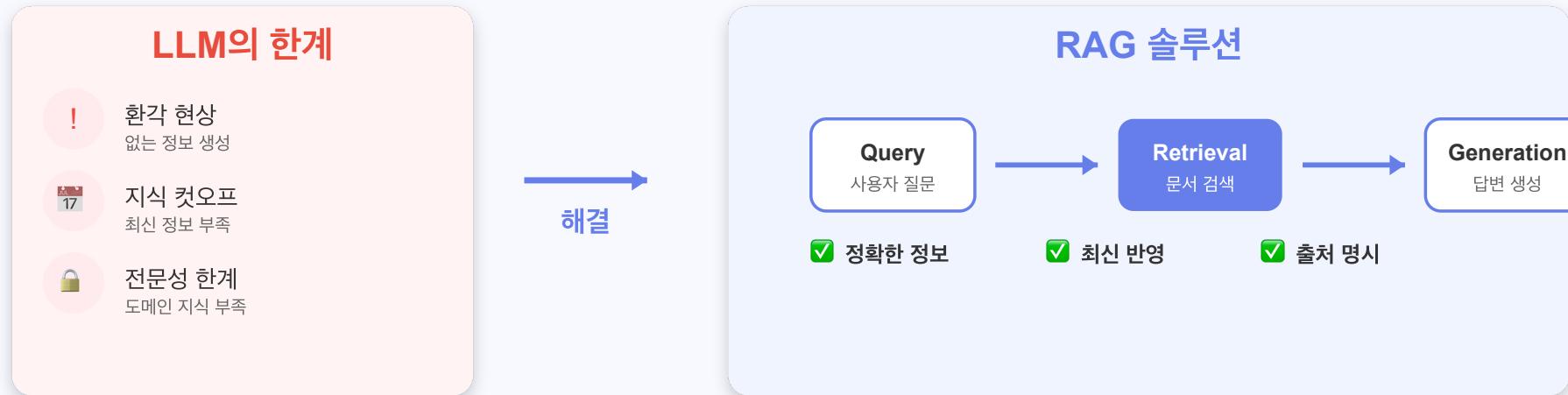
- CRAG
- RAGAS 평가
- 파라미터 튜닝

총 9개 실습으로 완성하는 RAG

실전 프로덕션 RAG 완전 정복

# RAG란 무엇인가?

Retrieval-Augmented Generation



## 핵심 개념

"LLM의 생성 능력 + 외부 지식 베이스 = 신뢰할 수 있는 AI"

검색된 문서를 컨텍스트로 제공하여 더 정확하고 최신의 답변 생성

# 김치찌개 예시로 이해하기

Naive RAG 작동 흐름

## 벡터 DB (13개 문서)

### 김치찌개 레시피

배추김치, 돼지고기, 두부, 파...

### 된장찌개 레시피

된장, 두부, 애호박, 감자...

### 불고기 레시피

쇠고기, 간장, 설탕, 마늘...

... (나머지 10개 문서)

백종원 김치찌개, 흑백요리사 레시피  
조리 용어 사전, 칼질 기법  
보관법, 영양 정보 등

### 총 13개 Chunk

(임베딩 벡터로 변환)

## 사용자 질문

"김치찌개 레시피 알려줘"

### Cosine Similarity 계산

#### 1 김치찌개 레시피

유사도: 0.92 (매우 높음)

#### 2 백종원 김치찌개

유사도: 0.88 (높음)

#### 3 된장찌개 레시피

유사도: 0.75 (중간)

LLM이 Top-3 문서 기반으로

답변 생성

# Naive RAG 아키텍처

가장 기본적인 RAG 구현



## ⚠ 주요 문제점

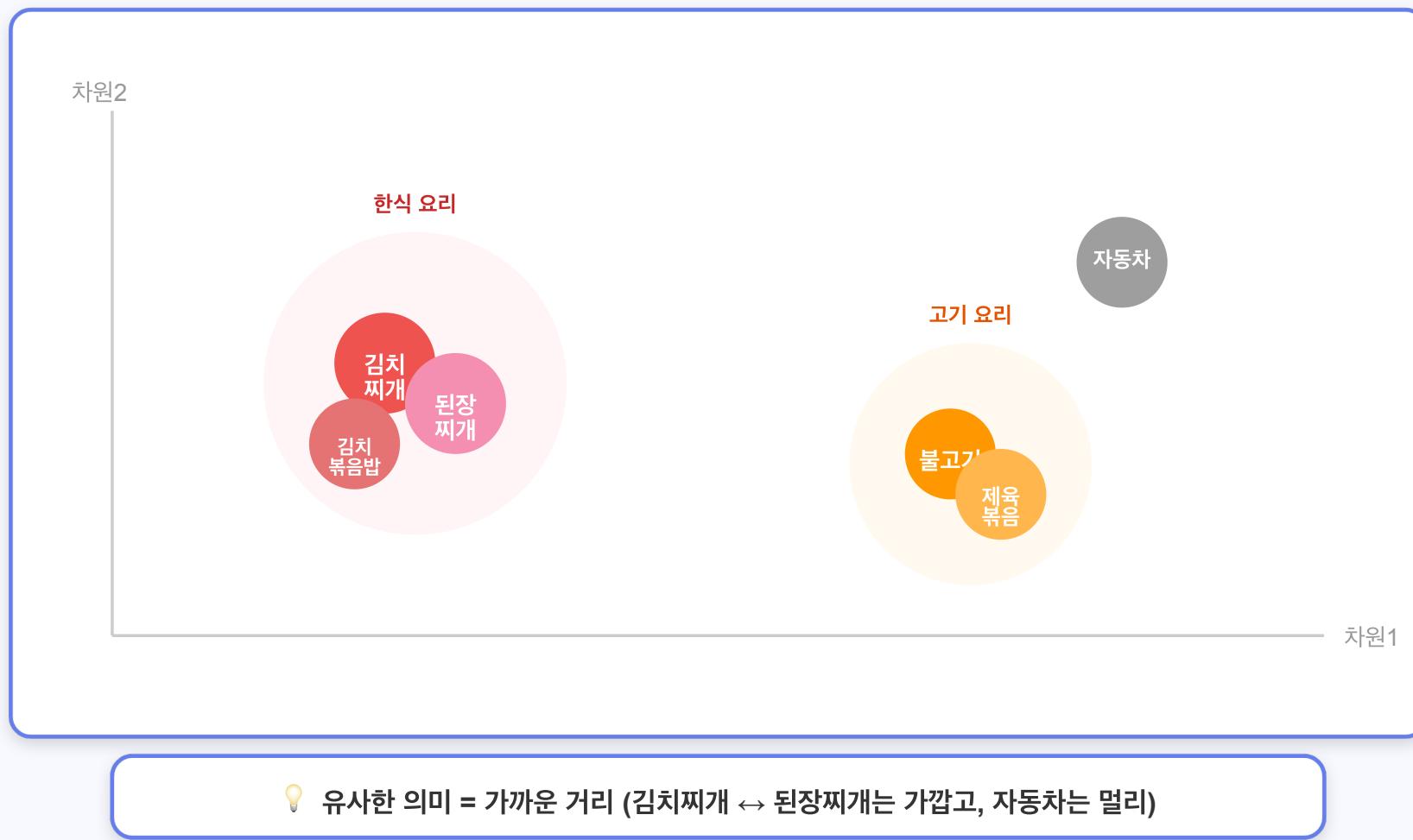
1. 낮은 정확도  
단순 유사도 매칭으로 부정확
2. 컨텍스트 손실  
문서 간 연관성 무시
3. 품질 검증 없음  
검색 문서의 관련성 검증 부재

## 핵심 포인트

- ✓ 구현은 단순  
5단계로 빠르게 구축 가능
- ✗ 성능은 낮음  
정확도 약 25-35% 수준
- ⌚ 개선 필요  
Query, Retrieval, Generation  
각 단계별 최적화 필요

# Vector Space

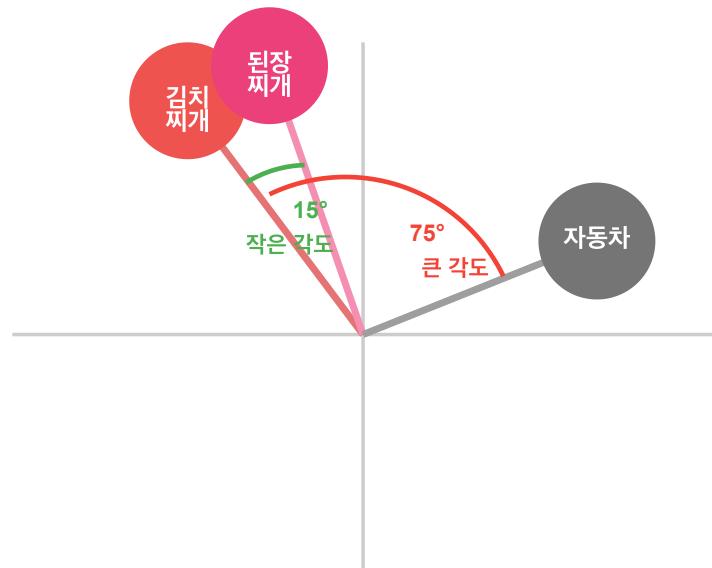
단어를 지도에 찍어보기



# Cosine Similarity

각도로 유사도 측정하기

각도로 유사도 측정



유사도 점수

$$\cos(\theta) = (\mathbf{A} \cdot \mathbf{B}) / (\|\mathbf{A}\| \times \|\mathbf{B}\|)$$

-1 ~ 1 범위 | 각도 작을수록 유사

김치찌개 vs 된장찌개

0.95

각도: 15° → 매우 비슷 (둘 다 찌개)

김치찌개 vs 불고기

0.65

각도: 45° → 약간 비슷 (둘 다 한식)

김치찌개 vs 자동차

0.05

각도: 85° → 완전 다른 (음식 vs 물건)

# Chunking 전략

왜 텍스트를 잘라야 하나?

## ✗ 전체 문서 한 번에

김치찌개는 한국의 대표적인찌개 요리입니다. 배추김치를 주재료로 사용하며, 돼지고기, 두부, 파 등을 함께 넣어 끓입니다. 김치의 신맛과 고기의 감칠맛이 어우러져 깊은 맛을 납니다. 찌개를 끓일 때는 물 대신 멸치 육수나 쌀뜨물을 사용하면 더욱 맛있습니다... (5000자 계속)

🚫 임베딩 크기 제한  
(512 토큰)

🚫 의미 뒤섞임  
(여러 주제)

## ✓ 적절한 크기로 분할

Chunk 1 (200자)  
김치찌개는 한국의 대표적인찌개...

Chunk 2 (200자)  
배추김치를 주재료로 사용하며...

Chunk 3 (200자)  
찌개를 끓일 때는 물 대신 육수를...

✓ 크기 제한 준수  
(512 이하)

✓ 의미 명확  
(주제 집중)

## 💡 왜 Chunking이 필요한가?

- ➊ 모델 제한: 임베딩 모델은 보통 512 토큰까지만 처리 가능
- ➋ 검색 정확도: 작은 청크일수록 의미가 명확해서 관련성 높은 내용만 검색
- ➌ 맥락 유지: 너무 작으면 문맥 손실, 너무 크면 노이즈 증가

# Overlap의 중요성

문맥을 잃지 않기 위한 핵심

## ✗ Overlap 없음

Chunk 1: 김치찌개는 배추김치와  
돼지고기를 넣고 끓인다.

Chunk 2: 국물을 자작하게 하려면  
물을 적게 넣는다.

### 문제: 연결이 끊김!

"국물"이 무엇을 말하는지 모름

## ✓ Overlap 있음

Chunk 1: 김치찌개는 배추김치와  
돼지고기를 넣고 끓인다. **국물을**

**국물을** 자작하게 하려면  
물을 적게 넣는다.

### 문맥 유지!

"국물"이 김치찌개 국물임을 파악

## 📌 권장 사항

### ✓ Overlap 크기: Chunk 크기의 10-20%

예: Chunk 500자 → Overlap 50-100자

### ✓ 문장 단위로 Overlap

단어 중간에서 자르지 말고 완전한 문장으로

### ✓ 성능 vs 저장 공간 트레이드오프

Overlap 클수록 정확도↑, 저장 공간↑, 비용↑

# Vector DB 비교

어떤 벡터 데이터베이스를 선택할까?

## Vector DB

### Milvus

오픈소스  
클라우드 네이티브

## 특징

- 초고속 검색 (HNSW, IVF)
- 대규모 확장성
- GPU 가속 지원

## 장점

- 엔터프라이즈급
- 고성능
- 자체 호스팅 가능

## 사용 사례

대용량 프로덕션  
실시간 검색

### FAISS

Facebook AI  
로컬 라이브러리

- 빠른 검색 알고리즘
- 메모리 효율적
- Python 통합 쉬움

- 무료
- 가볍고 빠름
- 학습 및 프로토타입

### Pinecone

SaaS (클라우드)  
완전 관리형

- 관리 불필요
- 자동 스케일링
- API 기반

- 즉시 사용 가능
- 운영 부담 제로
- 빠른 시작

개발/테스트  
소규모 프로젝트

스타트업  
빠른 MVP

## 실무 선택 가이드

개발: FAISS | 프로덕션: Milvus | 빠른 시작: Pinecone

# 5가지 실패 패턴

Naive RAG가 실패하는 이유

1

## Semantic Gap (의미 간극)

사용자 질문과 문서의 표현 방식이 달라 검색 실패  
예: "최신 동향" vs "트렌드", "방법" vs "how-to"

2

## Context Fragmentation

문서를 청크로 나누면서 맥락이 분절됨  
예: "이것", "그것" 같은 대명사 처리 불가

3

## Ambiguous Query (모호한 질문)

질문이 모호하거나 다의적이어서 정확한 검색 어려움  
예: "그거" "저번에 말한 것" 같은 애매한 표현

4

## Stale Information (오래된 정보)

DB 업데이트가 안 되어 구식 정보 반환  
예: 2023년 데이터로 2024년 질문에 답변

5

## Low Relevance (낮은 관련성)

유사도만으로 판단해 관련 없는 문서 반환  
예: 키워드만 비슷하고 내용은 무관한 문서

### 해결 방향

- Query Refinement (쿼리 개선)
- Hybrid Search + Reranking (검색 개선)

### 핵심 인사이트

이 5가지 문제를 해결하는 것이 Advanced RAG의 목표!

다음 슬라이드부터 각 문제의 해결 방법을 배웁니다

# Metadata Filtering

메타데이터로 검색 범위 좁히기



유사도 검색 전에 먼저 필터링!

```
date >= "2024-01-01"  
category = "AI"  
author = "전문가"
```



질문: "최근 AI 뉴스는?"

- 시간 필터: 최근 1개월
- 카테고리: "AI", "Tech"
- 출처 신뢰도: 4.0 이상



장점

- 1 검색 속도 향상 (범위 축소)
- 3 비용 절감 (처리 문서 감소)

- 2 정확도 증가 (관련 문서만)
- 4 시간 민감 쿼리 처리 가능



실전 팁: Vector Search와 조합하면 최고 효과! (Hybrid Filtering)

# Metadata Filtering 실전

Before/After 비교

✗ 필터 없음

질문: "최신 AI 뉴스는?"

🔍 전체 1000개 문서 검색

⌚ 검색 시간: 3.2초

📘 관련 문서: 12개 (1.2%)

✗ 2019년 오래된 기사 포함

비효율 + 부정확

✓ 필터 적용

질문: "최신 AI 뉴스는?"

📅 date >= 2024-06-01

🔍 50개 문서만 검색

⌚ 검색 시간: 0.3초 (10배 빠름)

✓ 관련 문서: 48개 (96%)

빠르고 정확!

## 실전 적용 사례

1 시간 필터: "최근", "올해", "작년" → **date 범위**

2 카테고리 필터: "AI 뉴스" → **category = ["AI", "Tech", "ML"]**

3 출처 필터: 신뢰도 4.0 이상만 검색

# 프롬프트 엔지니어링

LLM과의 효과적인 대화법

## 1 Few-shot Learning

예시로 배우기

질문: "AI란?"  
답변: "인공지능은..."  
질문: "RAG란?"  
답변: \_\_\_\_\_

## 2 Chain-of-Thought

단계별 사고

"Let's think step by step"  
1. 먼저 핵심 개념 파악  
2. 관련 정보 수집  
3. 논리적으로 결론 도출

## 3 ReAct Pattern

추론 + 행동

Thought: 정보가 부족해  
Action: 검색("최신 데이터")  
Observation: 결과 확인  
Answer: 최종 답변

### 💡 왜 중요한가?

- ✓ 더 정확한 답변: LLM이 문제를 더 잘 이해
- ✓ 일관성 향상: 매번 비슷한 품질의 답변 생성
- ✓ 환각 감소: 근거 기반 답변 유도
- ✓ 투명성: 추론 과정 추적 가능

#### 💡 실전 팁

✗ 나쁜 예: "답변해줘" → ✓ 좋은 예: "단계별로 생각하며 근거와 함께 답변해줘"

프롬프트가 구체적일수록 답변 품질이 높아집니다!

# 프롬프트 엔지니어링 실전

RAG 프롬프트 Before/After

## ✗ 애매한 프롬프트

### User Query:

"김치찌개 만드는 법 알려줘"

### LLM Prompt:

"김치찌개 만드는 법 알려줘"

### LLM 응답:

"김치찌개는 한국의 전통 음식입니다.  
일반적으로 배추김치와 돼지고기를  
넣고 끓입니다..."

### 문제점:

- 검색된 문서 무시 (환각 발생)
- 일반적인 답변만 제공
- 출처 명시 없음

## ✓ 구조화된 프롬프트

### LLM Prompt:

당신은 레시피 전문가입니다.

아래 검색된 문서를 참고하여  
질문에 답하세요.

### 검색된 문서:

- 김치찌개 레시피 (유사도: 0.92)  
"배추김치 200g, 돼지고기..."

질문: "김치찌개 만드는 법 알려줘"

문서에 없는 내용은 "모르겠습니다"  
라고 답하세요.

### LLM 응답:

"검색된 레시피에 따르면,  
배추김치 200g과 돼지고기를  
준비합니다... [출처: 문서 1]"

✓ 문서 기반 답변 + 출처 명시

# Query Refinement 기법

쿼리를 개선하는 3가지 핵심 방법

## 1 Multi-Query

하나의 쿼리 → 여러 각도

원본 쿼리:

"김치찌개 만드는 법 알려줘"



생성된 쿼리들:

1. "김치찌개 재료는 무엇인가?"
2. "김치찌개 조리 순서는?"
3. "김치찌개 맛있게 끓이는 팁"
4. "김치찌개 국물 내는 방법"

✓ 장점: 다양한 관점 커버

검색 범위 확대

## 2 Sub-Question

복잡한 질문 → 단계별 분해

원본 쿼리:

"김치찌개와 된장찌개의 재료 차이는?"



분해된 질문들:

- Step 1:  
"김치찌개 재료는?"  
Step 2:  
"된장찌개 재료는?"  
Step 3:  
"두 재료 비교"

✓ 장점: 복잡한 질문 처리

## 3 HyDE

가상 답변 생성 → 검색

원본 쿼리:

"김치찌개 만드는 법?"



가상 답변 생성 (LLM):

"김치찌개는 배추김치 200g,  
돼지고기 150g, 두부 1/2모,  
대파, 마늘을 넣고 끓입니다.  
먼저 돼지고기를 볶다가  
김치를 넣고 물을 부어  
15분간 끓이면 됩니다."

↓ 이 답변으로 검색

✓ 장점: 의미적으로 가까운 검색

# Query Refinement 실전

Before/After 비교

## ✗ 단일 쿼리

질문: "김치찌개 맛있게 만드는 법?"

🔍 검색 결과 (Top 3):

- 김치찌개 레시피 (유사도: 0.75)
- 찌개 종류 (유사도: 0.52)
- 된장찌개 만들기 (유사도: 0.48)

문제: 관련 문서 1개만 검색됨

## ✓ Multi-Query

3개 쿼리로 확장:

Q1: "김치찌개 재료와 계량"

→ 김치 200g, 돼지고기 150g...

Q2: "김치찌개 조리 순서"

→ 고기볶고 → 김치볶고 → 끓이기

Q3: "김치찌개 맛내는 비법"

→ 신김치 사용, 고춧가루 추가...

관련 문서 9개 검색 (3배 증가!)

## 성능 비교

단일 쿼리  
**33%**

관련 문서 발견율

Multi-Query  
**87%**

관련 문서 발견율

**+54%**

개선

# Hybrid Search

벡터 검색 + 키워드 검색의 조합

## 벡터 검색

(Semantic Search)

### 특징:

- 의미 기반 검색
- 유사한 표현 찾기
- 문맥 이해

## BM25 검색

(Keyword Search)

### 특징:

- 키워드 매칭
- 정확한 용어 검색
- TF-IDF 기반

## RRF 융합

(Reciprocal Rank Fusion)

### 결합 방식:

- 두 검색 결과 합침
- 순위 기반 점수화
- 최상의 문서 선택

## RRF 계산 예시

질문: "김치찌개 만드는 법"

### 벡터 검색 결과:

- 김치찌개 레시피 (순위: 1)
- 찌개 종류 (순위: 2)
- 김치 보관법 (순위: 3)

### BM25 검색 결과:

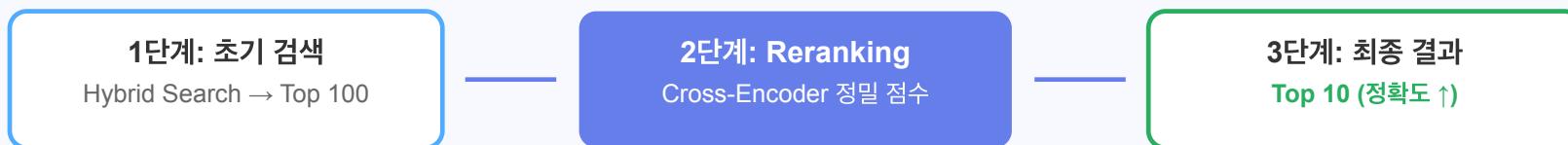
- 김치 보관법 (순위: 1)
- 김치찌개 레시피 (순위: 2)
- 된장찌개 만들기 (순위: 3)

### RRF 최종 결과:

1. 김치찌개 레시피 ( $1/61 + 1/62 = 0.0329$ ) ✓ 최고 점수!

# Reranking

Cross-Encoder로 정밀 재정렬



## ⌚ Cross-Encoder의 원리

### Bi-Encoder (기존)

Query와 Document를 따로 인코딩  
→ 빠르지만 상호작용 없음

### Cross-Encoder (개선)

Query + Document 함께 인코딩  
→ 느리지만 정확도 훨씬 높음

## ⟳ Reranking 실제 예시

### Hybrid Search 결과 (Top 5)

- 1위: 김치 보관법 (0.78)
- 2위: 백종원 김치찌개 (0.76)
- 3위: 김치찌개 레시피 (0.74)
- 4위: 김치찌개 고기 (0.72)
- 5위: 김치찌개 베터 유사도만 고려

### Reranking 후 결과

- 1위: 백종원 김치찌개 (0.94)
- 2위: 김치찌개 레시피 (0.91)
- 3위: 김치 보관법 (0.62)
- 4위: 김치찌개 의미까지 정밀 분석

# CRAG (Corrective RAG)

검색 결과를 자동으로 검증하고 수정



# CRAG 실전 예시

LLM Evaluator가 문서 품질을 판단

사용자 질문:

"김치찌개에 두부는 꼭 넣어야 하나?"

✓ Correct

검색된 문서:

"김치찌개에 두부는 선택사항입니다. 개인 취향에 따라 넣거나 생략할 수 있습니다."



LLM Evaluator 판단:

"관련성 높음!"



그대로 사용하여 답변 생성

✗ Incorrect

검색된 문서:

"된장찌개는 된장을 기본으로 하여 끓이며, 바지락이나 조개를 넣으면 좋습니다."



LLM Evaluator 판단:

"관련 없음!"



웹 검색으로 재시도

⚠ Ambiguous

검색된 문서:

"찌개 종류에는 김치찌개, 된장찌개, 순두부찌개 등이 있습니다."



LLM Evaluator 판단:

"애매함!"



Knowledge Strips 추출 +  
웹 검색 보완

# RAGAS 평가 프레임워크

RAG 시스템을 과학적으로 평가하기

## 1 Faithfulness

사실성 / 환각 감지

답변이 문서에 근거?

지원되는 문장 수 /  
전체 문장 수

## 2 Answer Relevancy

답변 관련성

질문에 직접 답변?

질문과 답변의  
유사도 점수

## 3 Context Precision

맥락 정밀도

검색 문서가 관련?

관련 문서 수 /  
전체 검색 문서 수

## 사용법

- ✓ 자동 평가
  - ✓ Ground Truth 필요
  - ✓ 4가지 지표 종합
- 약점 파악!

## 4 Context Recall

맥락 재현율

필요 문서 모두 검색?

검색된 필수 문서 /  
실제 필요 문서 (GT)

## 핵심

4가지 지표로

약점 파악

→ 이해하고

→ 적절히 사용!

# RAGAS 평가 실전

실제 점수로 보는 RAG 성능 측정

테스트 질문:

"김치찌개에 들어가는 주재료는 무엇인가요?"

## Faithfulness

답변의 사실성

**0.92**

의미:

답변이 검색된 문서의 내용과 일치하는가?

평가:

✓ 매우 높음  
답변이 문서에 근거함  
환각(hallucination)  
거의 없음

## Answer

답변 관련성

**0.88**

의미:

답변이 질문에 직접적으로 대답하는가?

평가:

✓ 높음  
질문에 잘 대응  
불필요한 정보  
거의 없음

## Context

문맥 정밀도

**0.67**

의미:

검색된 문서가 얼마나 관련있는가?

평가:

⚠️ 중간  
관련 없는 문서  
일부 포함됨  
→ 개선 필요!

## Context

문맥 재현율

**0.75**

의미:

필요한 정보를 얼마나 찾았는가?

평가:

✓ 양호  
주요 정보 대부분  
검색됨  
일부 누락 가능



개선 방향

Context Precision 향상을 위해 Reranking 적용 추천

# 파라미터 튜닝

최적의 설정값 찾기

## 1 Chunk Size

문서 쪼갤 크기

작게 (256)

→ 정확하지만 맥락 손실

크게 (1024)

→ 맥락 좋지만 노이즈

권장: 512 (균형점)

## 2 Top-K

검색할 문서 개수

적게 ( $k=1$ )

→ 빠르지만 정보 부족

많이 ( $k=10$ )

→ 정보 많지만 비용↑

권장:  $k=5$  (균형점)

## 3 Threshold

유사도 임계값

낮게 (0.1)

→ 관련성 낮은 문서 포함

높게 (0.7)

→ 너무 엄격, 누락 위험

권장: 0.3 (균형점)

## 튜닝 프로세스

### 1 Baseline 설정

→ 기본값으로 시작 (512 /  $k=5$  / 0.3)

### 2 하나씩 변경

→ Chunk Size 실험 (256 vs 512 vs 1024)

### 3 RAGAS 평가

→ 4가지 메트릭으로 성능 측정

### 4 최적값 선택

→ 가장 높은 점수의 파라미터 조합!

# 파라미터 튜닝

최적의 설정값 찾기

## 1 Chunk Size

문서 쪼갤 크기

작게 (256)

→ 정확하지만 맥락 손실

크게 (1024)

→ 맥락 좋지만 노이즈

권장: 512 (균형점)

## 2 Top-K

검색할 문서 개수

적게 ( $k=1$ )

→ 빠르지만 정보 부족

많이 ( $k=10$ )

→ 정보 많지만 비용↑

권장:  $k=5$  (균형점)

## 3 Threshold

유사도 임계값

낮게 (0.1)

→ 관련성 낮은 문서 포함

높게 (0.7)

→ 너무 엄격, 누락 위험

권장: 0.3 (균형점)

## 튜닝 프로세스

### 1 Baseline 설정

→ 기본값으로 시작 (512 /  $k=5$  / 0.3)

### 2 하나씩 변경

→ Chunk Size 실험 (256 vs 512 vs 1024)

### 3 RAGAS 평가

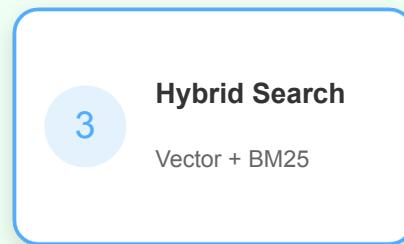
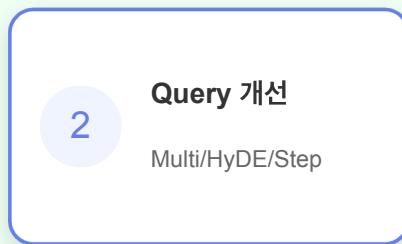
→ 4가지 메트릭으로 성능 측정

### 4 최적값 선택

→ 가장 높은 점수의 파라미터 조합!

# 전체 요약

Naive → Advanced RAG 여정



## 실전 구현 팁

- ✓ 단계별 적용: Naive → Query 개선 → Hybrid → Modular
- ✓ RAGAS로 각 단계 측정하며 점진적 개선
- ✓ 도메인 특성에 맞는 Chunking과 Metadata 설계
- ✓ 비용/속도/정확도 트레이드오프 고려

## 핵심 교훈

- 단일 기법보다 조합이 중요
- RAGAS로 측정하며 과학적 개선
- 각 메트릭을 이해하고 적절히 활용

# 축하합니다!



## Day 2: Advanced RAG Systems 완료

### 🎓 학습한 내용

- ✓ 임베딩과 벡터 DB: 의미 기반 검색의 기초
- ✓ Naive RAG의 5가지 실패 패턴 분석
- ✓ 프롬프트 엔지니어링으로 답변 품질 개선
- ✓ Metadata Filtering: 조건 기반 사전 필터링
- ✓ Hybrid Search: Vector + BM25 + Reranking
- ✓ CRAG: 검색 품질 자가 평가 및 보완
- ✓ RAGAS: 과학적 평가 프레임워크