생성형 AI 기반 RAG 개발자 과정

임베딩과 유사도 검색 직접 구현

Session 7 / 33 — Day 1

벡터 DB 없이, 코사인 유사도만으로 직접 구현

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

출처: Day 1 임베딩·유사도

이번 시간을 마치면

🔢 임베딩을 만든다

- 문장 → **벡터**(숫자 배열)로
- **의미 공간** 직관 확인
- OpenAI 임베딩 API 호출

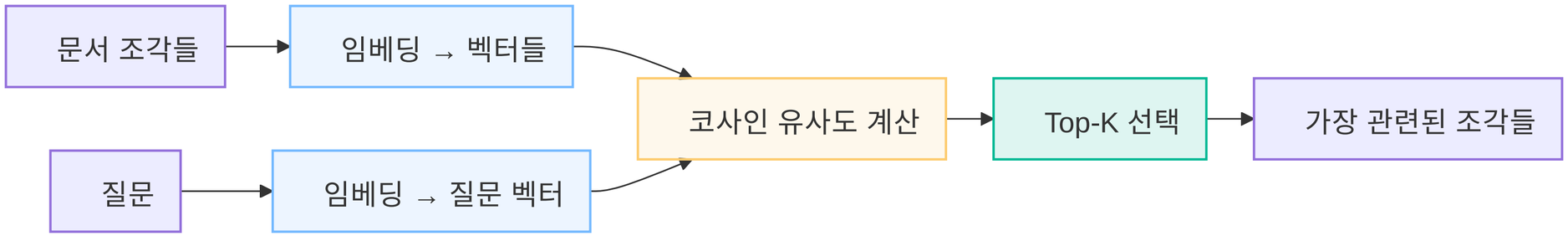
📐 검색을 구현한다

- **Cosine 유사도** 직접 계산
- **Top-K** 가장 가까운 조각 찾기
- 가장 단순한 **검색 함수** 완성

🎯 목표: "임베딩 + 코사인 = 의미 검색"을 **손으로** 체득. 코드는 스니펫, 전체는 실습 때.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

벡터 DB 없이 만드는 의미 검색

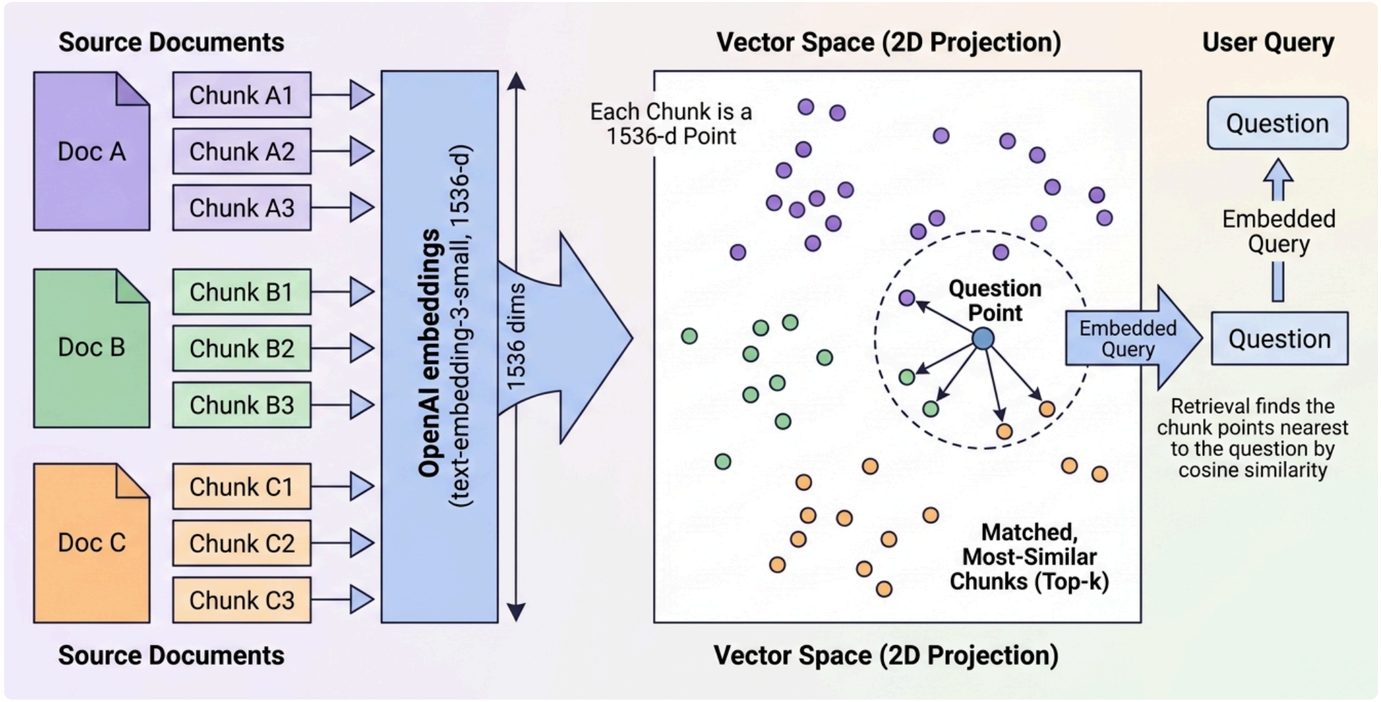


💡 위 흐름이 **모든 벡터 DB 의 내부**. 청크와 질문을 **같은 1536차원**으로 임베딩해 가장 가까운 K개를 찾는 것 — Day2 는 이걸 빠르고 영속적으로 만든 것.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

출처: Day 1 임베딩·유사도

청크·질문을 같은 차원으로 임베딩해 매칭



💡 청크와 질문을 **같은 1536차원**으로 임베딩해 가장 가까운 K개를 찾습니다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

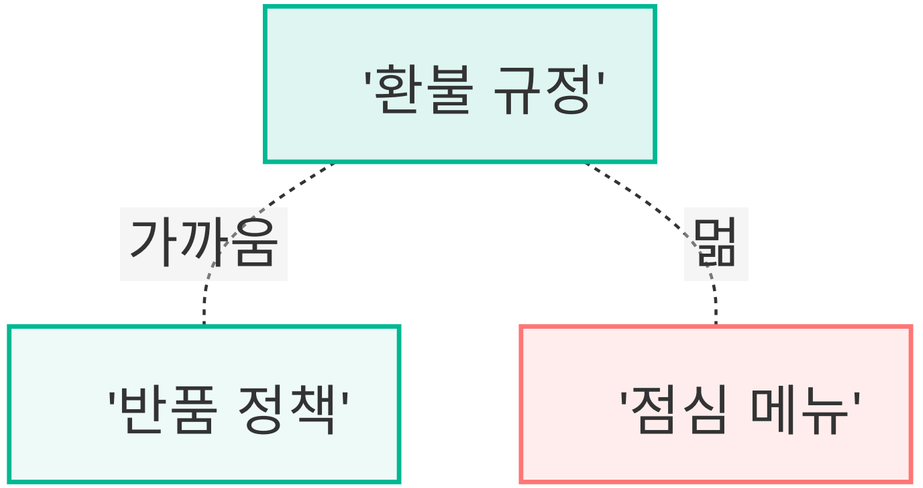
벡터 = 의미의 좌표

직관

- 임베딩 = 문장을 **여러 숫자**로 표현
- 예: `[0.12, -0.04, 0.88, ...]` (1536개)
- 각 숫자는 의미의 **한 축**
- 비슷한 의미 → **가까운 좌표**

의미 공간

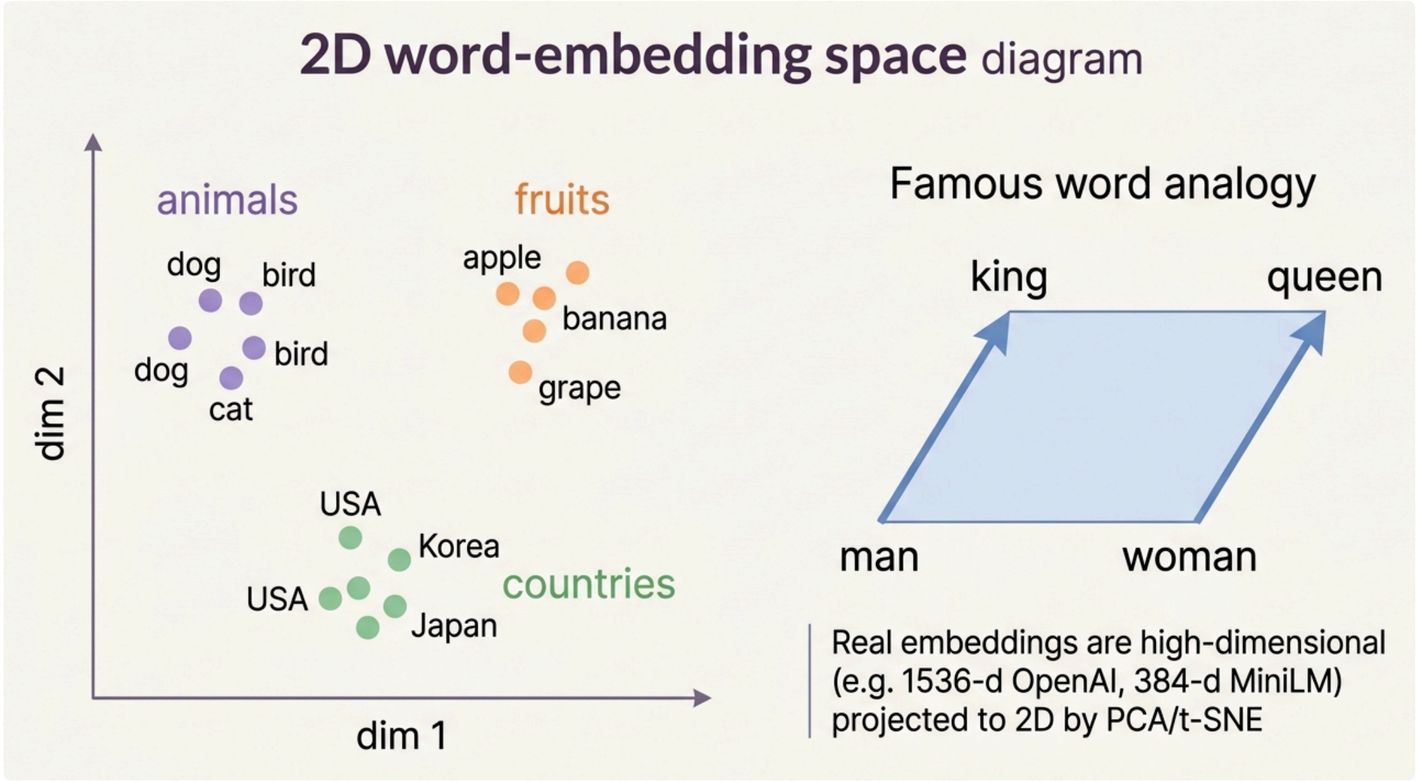
- 모든 문장이 분포하는 **다차원 공간**
- "가깝다 = 비슷하다"



> 🔑 검색이란 = 질문 벡터에서 **가장 가까운 문서 벡터**를 찾는 일. 의미 관계는 평행 벡터로(`king-man+woman≈queen`).

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

의미 공간 한눈에 보기



🔑 비슷한 의미의 문장은 **군집**을 이루고, 의미 관계는 **평행 벡터**로 나타난다(`king-man+woman≈queen`). 검색이란 곧 질문 벡터에서 **가장 가까운 문서 벡터**를 찾는 일.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

문장을 벡터로 (스니펫)

```
from openai import OpenAI
client = OpenAI()

resp = client.embeddings.create(
    model="text-embedding-3-small",
    input="환불은 7일 이내 가능합니다",
)
vec = resp.data[0].embedding   # 길이 1536 의 float 리스트
print(len(vec), vec[:3])       # 1536 [0.01, -0.02, ...]
```

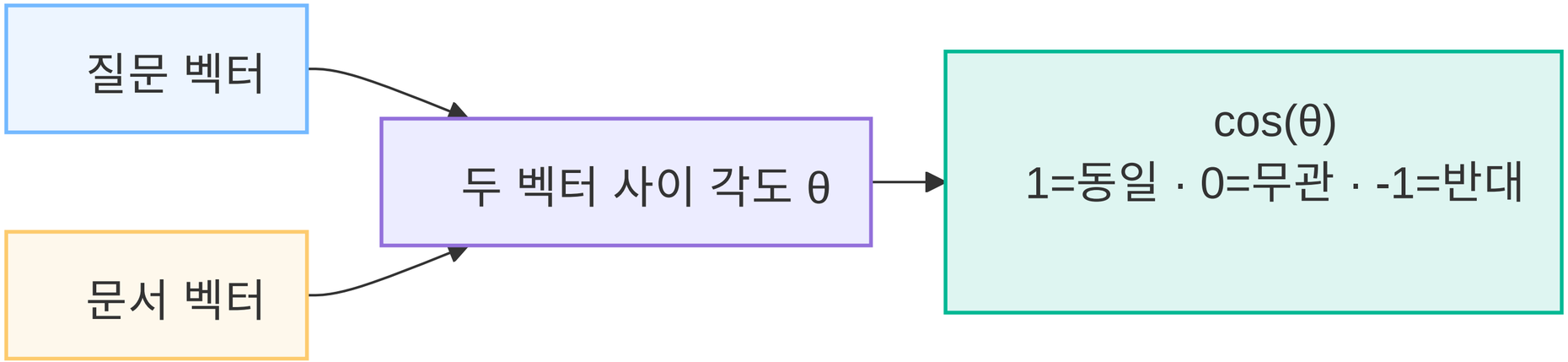
- 한 번 호출로 문장 → **1536차원 벡터**
- 매우 저렴 (`text-embedding-3-small`)
- 문서 조각마다 한 번씩 임베딩 → 저장해두고 재사용

⚠️ **한 가지 철칙**: 문서와 질문은 **같은 임베딩 모델**로. 다른 모델은 다른 의미 공간 → 거리 비교가 무의미. 모델을 바꾸면 차원도 달라져 인덱스를 **재구축**해야 함.

> 📁 전체 실행 코드: `2.langchain/7.RAG/1.basics/1.1_embeddings_intro.py`

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

코사인 유사도 — "방향이 같은가"



왜 각도인가

- 길이(크기)가 아니라 **방향**이 의미를 담음
- 각도가 작을수록(같은 방향) **더 유사**

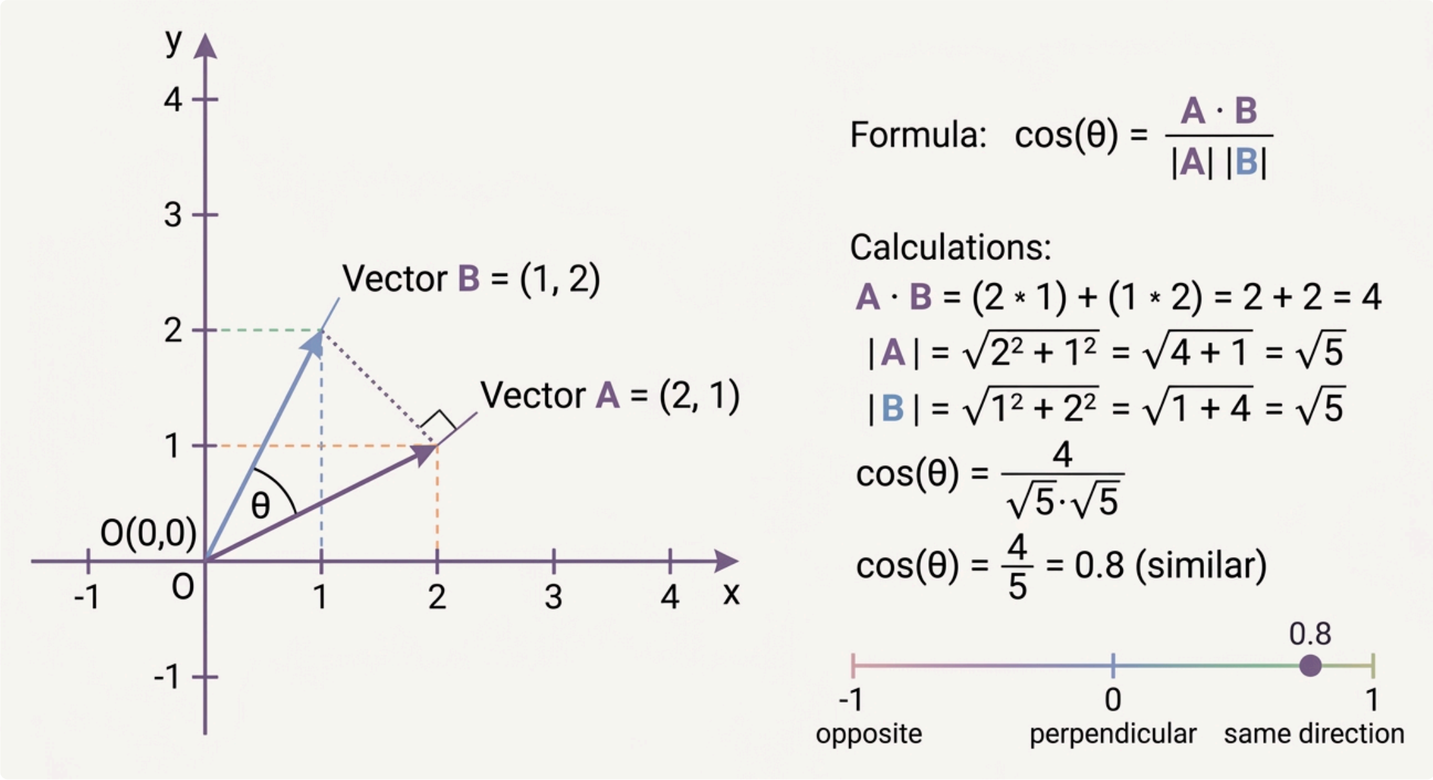
값의 해석

- **1.0** — 거의 같은 의미
- **0.7~0.9** — 관련 높음

💡 척도는 여럿(L2·내적·맨해튼)이지만, **벡터를 정규화하면 내적 = 코사인** — 실무 검색 엔진이 내부에서 쓰는 방식. 텍스트 검색의 사실상 표준은 **코사인 유사도**.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

cos(θ) = (A·B) / (‖A‖‖B‖)



💡 분자는 **내적**, 분모는 **두 벡터의 크기**. 크기를 나눠 없애므로 **방향(각도)만** 남는다 — 그래서 문서 길이에 둔감하고 값은 항상 **−1~1**.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

라이브러리 없이 (스니펫)

```
import math

def cosine(a, b):
    dot = sum(x*y for x, y in zip(a, b))          # 내적
    na  = math.sqrt(sum(x*x for x in a))          # a 크기
    nb  = math.sqrt(sum(y*y for y in b))          # b 크기
    return dot / (na * nb)                        # 코사인값

print(cosine(q_vec, doc_vec))   # 예: 0.83 → 관련 높음
```

- 단 4줄 — **내적 / (크기×크기)**
- 실무에선 `numpy` 로 더 빠르게: `np.dot(a,b)/(norm(a)*norm(b))`

> 🔑 벡터 DB 가 내부에서 하는 핵심 연산이 바로 이것. 우린 그 원리를 손으로 본 것.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

가장 가까운 K개 고르기 (스니펫)

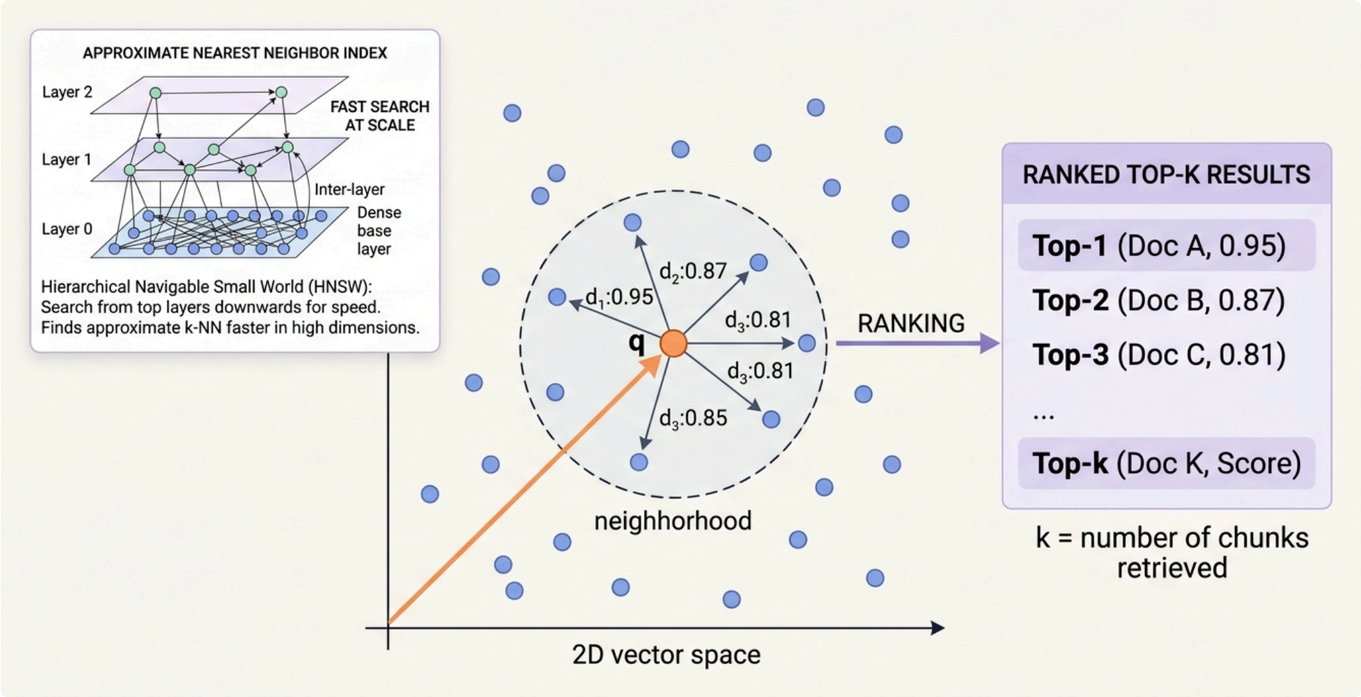
```
def search(query, chunks, chunk_vecs, k=3):
    q = embed(query)                              # 질문 임베딩
    scored = [(cosine(q, v), c)                   # 각 조각과 유사도
              for c, v in zip(chunks, chunk_vecs)]
    scored.sort(reverse=True)                     # 높은 순 정렬
    return scored[:k]                             # 상위 K개

for score, chunk in search("환불 며칠?", chunks, vecs):
    print(round(score, 3), chunk[:40])
```

- 미리 임베딩해 둔 `chunk_vecs` 와 질문을 비교 → 정렬 → 상위 K
- 이게 곧 **retriever** — Day2 에서 `vectorstore.as_retriever()` 로 대체됨

> 📁 더 발전된 형태: `2.langchain/7.RAG/1.basics/1.2_inmemory_vectorstore.py`
> 🔑 문서가 수백만 개로 늘면 전체 비교는 느림 → 대규모는 **ANN/HNSW** 근사 검색으로 대체.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

Top-K 검색 — 한 그림으로



🔑 질의 벡터와 문서 벡터의 유사도를 계산해 가장 가까운 K개를 반환합니다.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

오늘 손으로 한 것이 Day2 의 자동화

오늘 (손으로)	Day2 (벡터 DB)
리스트에 벡터 보관	FAISS / ChromaDB 인덱스
`cosine()` 직접	DB 내장 유사도
`sort()[:k]`	`as_retriever(k=3)`
매번 전체 비교 (느림)	**HNSW** 근사 검색 (빠름)
껐다 켜면 사라짐	**영속화** (재실행 비용 0)

💡 벡터 DB = 오늘 만든 검색을 **빠르고·영속적이고·메타데이터까지** 되게 한 것.
"마법이 아니라 최적화"임을 기억.

⚠️ 검색은 "가장 가까운 것"을 줄 뿐 "정말 관련 있는 것"을 보장하지 않음 → **유사도 점수가 임계값 아래면 "관련 문서 없음"** 처리. 환각 방지의 마지막 안전장치.

🔎 **임계값은 척도마다 다름**: **코사인**은 −1~1 범위라 보통 0.7~0.8 정도를 기준으로 삼고, **L2 거리**를 `1/(1+거리)`로 바꾼 **0~1 점수**에서는 0.2 같은 낮은 값이 컷오프. 같은 "0.2"라도 어느 척도의 점수인지 반드시 구분할 것.

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현

핵심 5가지

1. ✅ 임베딩 = 문장을 **의미 좌표(벡터)** 로 — `text-embedding-3-small`
2. ✅ 의미 공간: **가까움 = 비슷함**
3. ✅ **코사인 유사도** = 두 벡터의 방향(각도) 비교, 4줄로 구현
4. ✅ **검색 = 질문 벡터에 가장 가까운 Top-K 조각** 찾기
5. ✅ 벡터 DB 는 이 과정을 **빠르고 영속적으로** 자동화한 것

© 2026 젠아이랩스 (GenAI Labs) · 강사 박수현