LangChain 9주차

Date	@2026년 1월 11일
Category	GDGoC
Study	LangChain 스터디
day of the week	Sun
form	스터디
done	☐

CH09 벡터저장소(VectorStore)

벡터스토어(VectorStore)는 **Retrieval-Augmented Generation(RAG)** 파이프라인에서 네 번째 단계에 해당하며, 앞선 단계에서 생성된 **임베딩 벡터(embedding vectors)** 를 체계적으로 저장·관리하는 역할을 수행한다. 이 단계의 목적은 단순한 데이터 보관이 아니라, **의미 기반 검색을 빠르고 안정적으로 수행할 수 있는 검색 인프라를 구축하는 것**이다.

RAG 시스템에서 생성된 임베딩은 일반적인 관계형 데이터베이스나 키–값 저장소에 그대로 저장할 수도 있지만, 이러한 방식은 고차원 벡터 간의 유사도를 효율적으로 계산하기 어렵다. 벡터스토어는 **고차원 벡터 공간에서의 거리 계산(예: cosine similarity, inner product, L2 distance)** 을 전제로 설계된 저장 구조와 인덱싱 방식을 제공함으로써, 대규모 문서 집합에서도 실시간에 가까운 검색 성능을 가능하게 한다.

벡터스토어 저장의 필요성

1. 대규모 데이터에서의 빠른 검색 성능

임베딩 벡터는 일반적으로 수백~수천 차원의 실수 벡터로 구성되며, 문서 수가 증가할수록 단순한 선형 탐색 방식은 실용적인 성능을 보장하기 어렵다. 벡터스토어는 **근사 최근접 이웃 탐색(ANN, Approximate Nearest Neighbor)** 알고리즘과 전용 인덱스를 활용하여, 대량의 벡터 중에서도 질의 벡터와 가장 유사한 벡터들을 매우 빠르게 탐색할 수 있도록 설계되어 있다.

이로 인해 사용자는 수만~수백만 개의 문서가 저장된 환경에서도, 검색 지연 없이 관련 문서를 조회할 수 있다.

2. 확장성(Scalability)과 운영 안정성

실제 서비스 환경에서는 데이터가 지속적으로 추가되며, 임베딩 또한 반복적으로 생성된다. 벡터스토어는 이러한 상황을 고려하여 **수평적 확장** 또는 **샤딩(sharding)** 이 가능하도록 설계된 경우가 많으며, 일부 시스템(Pinecone 등)은 완전 관리형 서비스로서 자동 확장을 지원한다.

적절한 벡터스토어를 사용하면 데이터 규모 증가에 따라 성능이 급격히 저하되는 문제를 방지할 수 있으며, RAG 시스템 전체의 응답 지연(latency)과 안정성에 직접적인 영향을 미친다.

3. 의미 검색(Semantic Search)의 핵심 인프라

텍스트 기반 데이터베이스는 기본적으로 키워드 매칭에 의존한다. 이 방식은 사용자가 문서에 포함된 정확한 표현을 사용하지 않으면 원하는 결과를 얻기 어렵다는 한계를 가진다.

반면 벡터스토어는 **문장의 의미적 표현 자체를 벡터로 저장**하기 때문에, 질문과 문서 간의 표현 차이가 존재하더라도 의미적으로 유사한 문서를 검색할 수 있다. 이는 자연어 질의 기반 인터페이스에서 매우 중요한 특성이다.

예를 들어 다음과 같은 질문을 고려할 수 있다.

> 질문: “모바일 디바이스 상에서 동작하는 인공지능 기술을 소개한 기업은 어디인가?”

이 질문에 대해 키워드 검색은 “모바일”, “인공지능”, “기업”이라는 단어가 명시적으로 포함된 문서에만 반응하지만, 벡터 검색은 **“온디바이스 AI”, “엣지 AI”, “스마트폰용 AI 칩”** 과 같이 표현이 다른 문서들도 의미적으로 연관된 결과로 반환할 수 있다.

벡터스토어의 시스템적 중요성

벡터스토어 저장 단계는 RAG 시스템에서 **검색(Retrieval) 단계의 품질을 사실상 결정하는 핵심 구성 요소**이다. 동일한 임베딩 모델과 동일한 LLM을 사용하더라도, 벡터스토어의 종류, 인덱싱 방식, 검색 전략(k, MMR, score threshold 등)에 따라 최종 응답의 정확도와 일관성은 크게 달라질 수 있다.

즉, 벡터스토어는 단순한 저장소가 아니라

- 검색 품질
- 응답 시간
- 시스템 확장성
- 운영 비용

모두에 영향을 미치는 **RAG 시스템의 기반 인프라**라고 볼 수 있다.

벡터스토어는 단일한 기술 선택지가 아니라, **시스템의 성숙도와 운영 환경에 따라 달라지는 선택의 연속선** 위에 놓여 있다.

Chroma, FAISS, Pinecone은 서로 경쟁 관계라기보다는, **서로 다른 문제 단계에 대응하는 해법**에 가깝다.

1. Chroma: “RAG를 가장 빨리 시작하기 위한 벡터스토어”

Chroma는 벡터스토어를 **개발 친화적(local-first)** 관점에서 재해석한 도구이다.

복잡한 인덱싱 설정이나 인프라 구성 없이, **RAG 파이프라인을 빠르게 실험하고 검증하는 것**을 최우선 목표로 설계되었다.

```
from langchain.vectorstores import Chroma

vectorstore = Chroma.from_documents(
    documents=docs,
    embedding=embeddings
)

results = vectorstore.similarity_search("온디바이스 AI 기술")
```

핵심 특징

- 로컬 환경에서 바로 사용 가능
- 문서, 메타데이터, 임베딩을 함께 관리
- LangChain과의 높은 통합성
- “데이터베이스처럼 쓰는 벡터스토어”에 가까운 추상화

Chroma의 장점은 **“생각하지 않아도 되는 것”이 많다는 점**이다.

인덱스 구조, 검색 알고리즘, 샤딩 전략 등을 사용자가 직접 고민하지 않아도, RAG 실험에 필요한 최소 기능을 즉시 제공한다.

사용 맥락

- 개인 프로젝트
- 프로토타입 개발
- 연구·실험 단계
- 소규모 문서 기반 QA 시스템

하지만 Chroma는 의도적으로 **대규모 분산 시스템 문제를 다루지 않는다**.

즉, "지금 당장 써보기에는 최고지만", 시스템이 커질수록 한계가 명확해진다.

이 지점에서 자연스럽게 다음 선택지가 등장한다.

2. FAISS: "벡터 검색 그 자체에 집중한 엔진"

FAISS는 Facebook AI Research에서 개발한 라이브러리로, 벡터스토어라기보다는 **고성능 벡터 검색 엔진**에 가깝다.

Chroma가 "편의성 중심의 저장소"라면, FAISS는 **"검색 성능 중심의 알고리즘 집합"**이다.

```
from langchain.vectorstores import FAISS

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

results = vectorstore.similarity_search("엣지 AI 기업")
```

핵심 특징

- 다양한 근사 최근접 이웃(ANN) 알고리즘 제공
- IVF, HNSW, PQ 등 세밀한 인덱스 설계 가능
- CPU/GPU 기반 고성능 검색
- 매우 큰 벡터 집합에서도 빠른 검색 가능

FAISS는 "어떻게 저장할 것인가"보다는 **"어떻게 가장 빠르고 정확하게 찾을 것인가"**에 집중한다.

그 결과, FAISS는 다음과 같은 특성을 가진다.

- 메타데이터 관리 기능이 제한적
- 영속성, 분산, 장애 복구는 직접 구현해야 함
- 운영 환경에서는 추가 인프라 설계가 필수

사용 맥락

- 대규모 임베딩 검색 실험
- 검색 성능 튜닝이 중요한 연구 환경
- 커스텀 RAG 파이프라인
- 내부 시스템에 직접 통합하는 경우

즉, FAISS는 **"엔진"** 이지 **"서비스"** 는 아니다. 이 점이 다음 단계로 이어지는 중요한 연결 고리다.

3. Pinecone: "운영 가능한 벡터 검색 인프라"

Pinecone은 FAISS가 다루지 않는 문제를 정면으로 해결한다. 즉, 벡터 검색을 **서비스 수준에서 안정적으로 운영하는 것**이 목표이다.

```
from langchain.vectorstores import Pinecone as PineconeStore

vectorstore = PineconeStore.from_existing_index(
    index_name="rag-index",
    embedding=embeddings
)

results = vectorstore.similarity_search("모바일 AI 스타트업")
```

핵심 특징

- 완전 관리형(Managed) 벡터 데이터베이스
- 자동 인덱싱, 스케일링, 복제
- 높은 가용성과 낮은 지연 시간
- 메타데이터 필터링과 하이브리드 검색 지원

Pinecone은 더 이상 “라이브러리”가 아니라, **RAG 시스템의 검색 계층을 외주화하는 선택지**에 가깝다.

사용자는 다음과 같은 질문을 하지 않아도 된다.

- 인덱스를 언제 다시 빌드해야 할까?
- 데이터가 늘어나면 어떻게 샤딩할까?
- 장애가 나면 어떻게 복구할까?

이 모든 질문을 Pinecone이 대신 처리한다.

사용 맥락

- 프로덕션 RAG 서비스
- 다수의 사용자 요청을 처리해야 하는 환경
- SLA, 안정성, 운영 비용이 중요한 경우
- 엔터프라이즈급 시스템

하나의 연속선으로 보면

이 세 가지를 나란히 비교하기보다, **하나의 성장 경로**로 보면 훨씬 자연스럽다.

- **Chroma**

  → “RAG를 이해하고 빠르게 실험한다”
- **FAISS**

  → “검색 품질과 성능을 직접 통제한다”
- **Pinecone**

  → “검색을 인프라로서 운영한다”

이 흐름은 단순히 기술의 우열이 아니라, **시스템의 성숙도와 요구 사항이 변화하는 과정**을 반영한다.

중요한 관점 정리

- Chroma는 **개발 경험(DX)** 을 최적화한다.
- FAISS는 **검색 알고리즘과 성능**을 최적화한다.
- Pinecone은 **운영과 확장성**을 최적화한다.

따라서 "어떤 것이 더 좋은가?"보다는 **"지금 내 시스템은 어느 단계에 있는가?"** 가 더 중요한 질문이다.