덕성여자대학교
2025학년도 2학기

인공지능 플랫폼 실습 및 설계

11주차 : 대규모 언어모델(LLM)을 사용한 애플리케이션 이해(4)

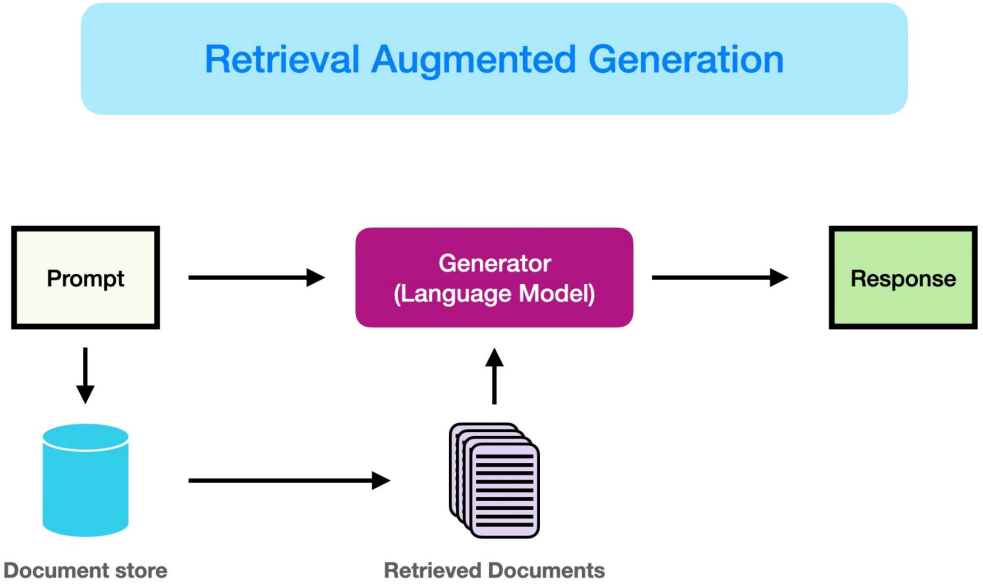
검색 증강 생성(RAG, Retrieval Augmented Generation)

하 윤 종 (hayunjong83@gmail.com)

지난 시간 실습 사항

- 인공지능플랫폼실습및설계 11주차 실습
- 인공지능플랫폼실습및설계 11주차 실습(결과)

검색 증강 생성(Retrieval Augmented Generation)



훈련된 딥러닝 모델로서 대규모 언어모델(LLM)은

- 학습한 데이터에만 의존하여 답변을 생성하므로
  학습되지 않은 최신 정보나
  LLM을 사용하는 조직의 지식 및 데이터는 알지 못한다.

- 이에 따라 LLM의 답변이
  근거나 출처가 불명확하고, 부정확한 정보로부터
  답변을 출력하는 환각현상(Hallucination)이 존재한다.

검색 증강 생성(RAG, Retrieval Augmented Generation)

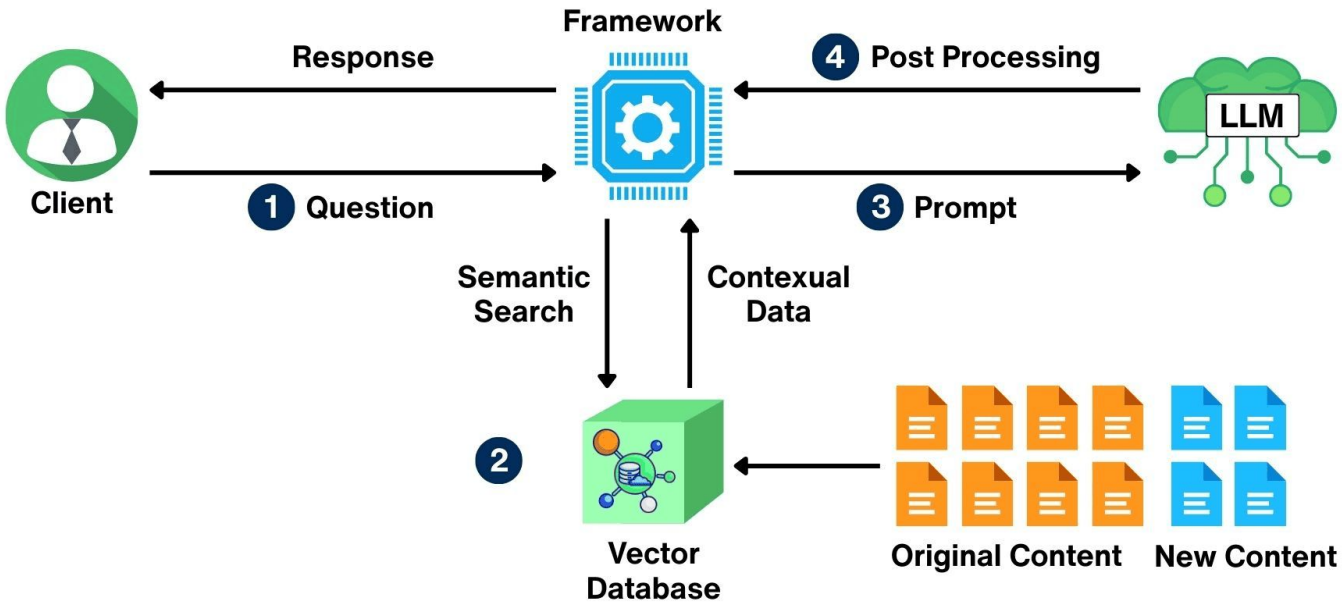
- LLM에게 단순히 질문이나 요청만 전달하여
  답변을 생성하는 것이 아니라

- 답변에 필요한 충분한 정보와 맥락(context)를
  제공하여 답변하도록 요청하는 방법

- 필요한 정보와 맥락은 "검색(Retrieval)"을 통해
  선택되고 제공 ⇒ 검색을 통하여 답변을 보충, 생성성

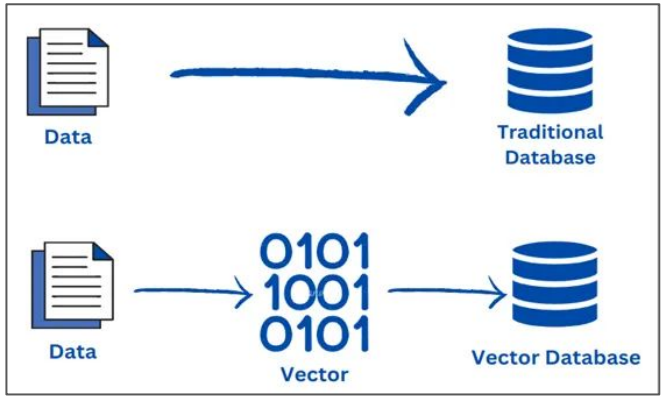
K21Academy
Learn Cloud, Data & AI from Experts

RAG ARCHITECTURE MODEL

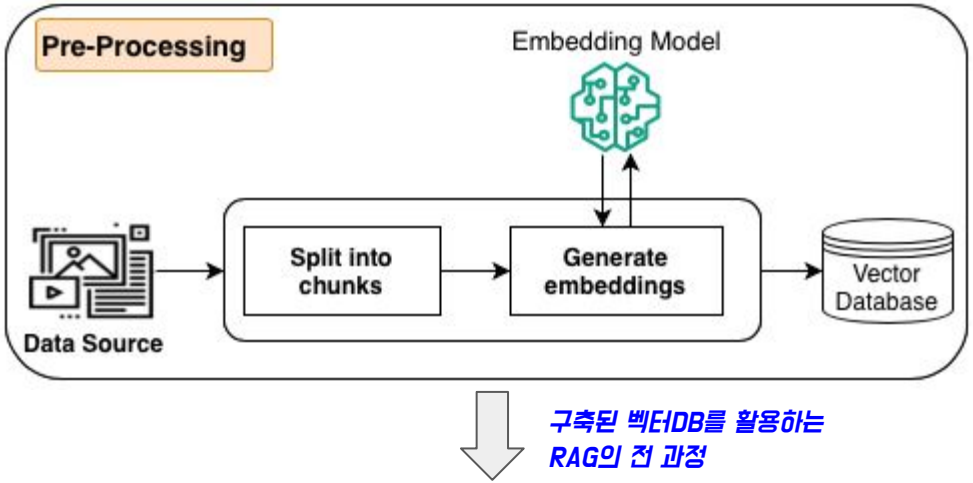


참고) [An Overview of Retrieval-Augmented Generation(RAG) and RAGOps]

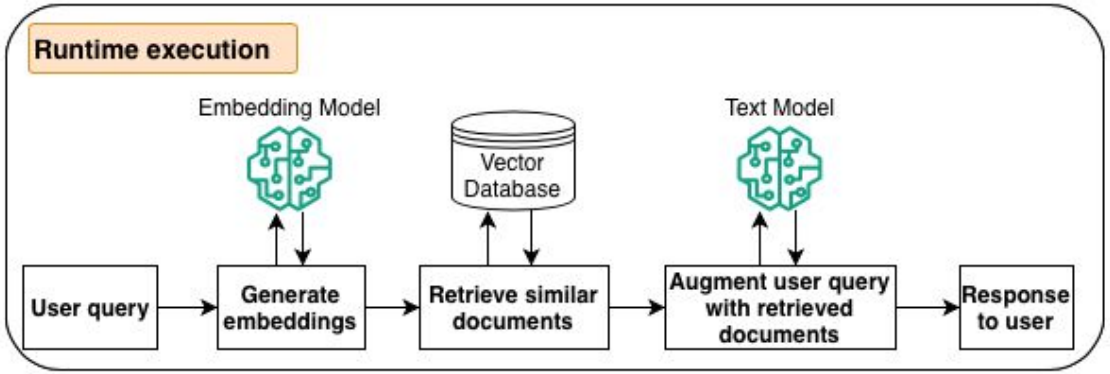
※ 벡터 데이터베이스(Vector Database)



▼ 벡터 데이터베이스에 문서와 정보를 저장하는 과정



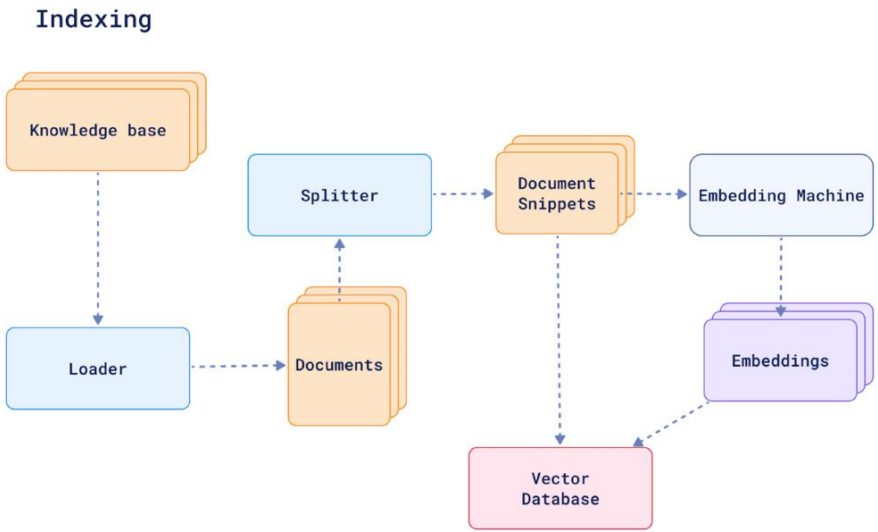
구축된 벡터DB를 활용하는 RAG의 전 과정



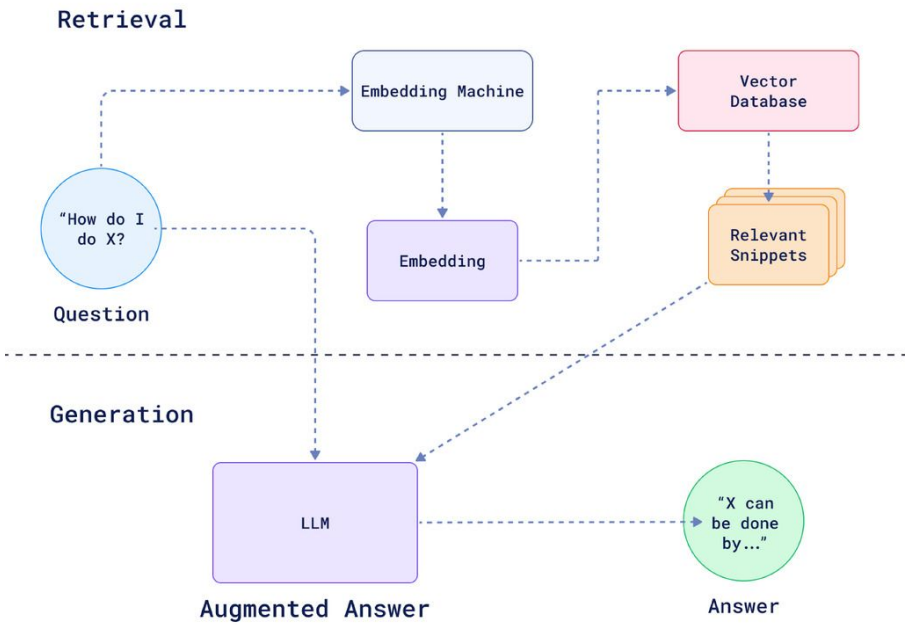
- 벡터 데이터베이스에 **지식과 정보를 저장**할 때 사용하는 임베딩 모델(Embedding Model)과 최초의 **사용자 질문**(Query)을 검색하기 위해 **임베딩 벡터로 변환**하는 **임베딩 모델**은 **반드시 같은 모델**이어야 한다.

- 검색된 유사문서와 최초의 사용자 질문은 합쳐져 답변 생성을 위한 LLM에 다시 입력된다.
  - 이 때. 함께 제공하는 문서가 맥락(context)정보
  - 답변생성을 위한 LLM과 벡터DB 검색을 위한 임베딩모델은 같은 모델이 아니다.

지식기반(Knowledge base) 저장 과정 - Indexing



연관된 문서 검색(Retrieval)과 최종 답변 생성(Generation) 과정



※ 벡터 데이터베이스의 의미 검색(semantic search)을 위한 대표적인 방식인 ANN(Approximate Nearest Neighbor)

- 전통적인 데이터베이스의 검색(search)은 정확히 일치하는 값을 찾는다(exact match)

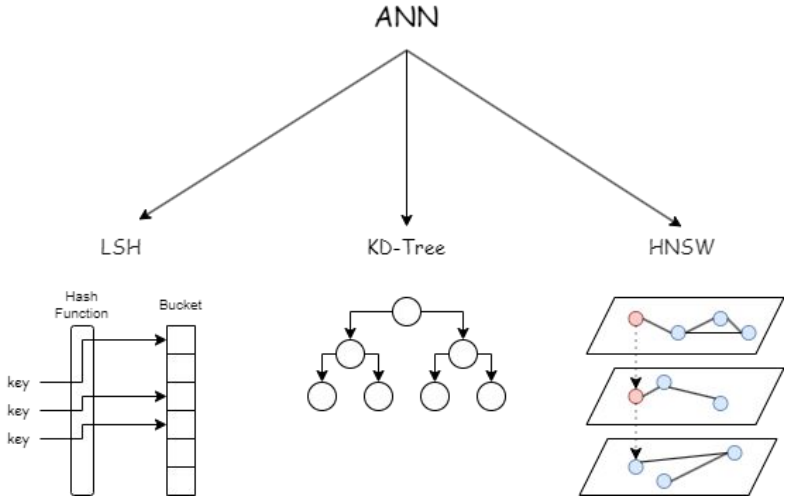
- 자연어 질의-응답, 유사 이미지의 검색을 위해서는 "정확한 일치"가 아닌
  의미적으로 유사한 대상을 찾는 것이 필요하다.
  ⇒ **벡터 검색**(vector search)
  : 검색하려는(query) 텍스트, 이미지 등을 고차원 (임베딩) 벡터로 바꾼 뒤에
  거리 기반의 유사도를 비교하는 방식

- 벡터 검색이 곧 **의미 기반 검색**(semantic search)를 뜻한다.

k-최근접 이웃(KNN, K-Nearest Neighbor)는 대규모 데이터베이스에서 한계를 가진다.
- 계산 복잡도와 메모리 사용량 문제

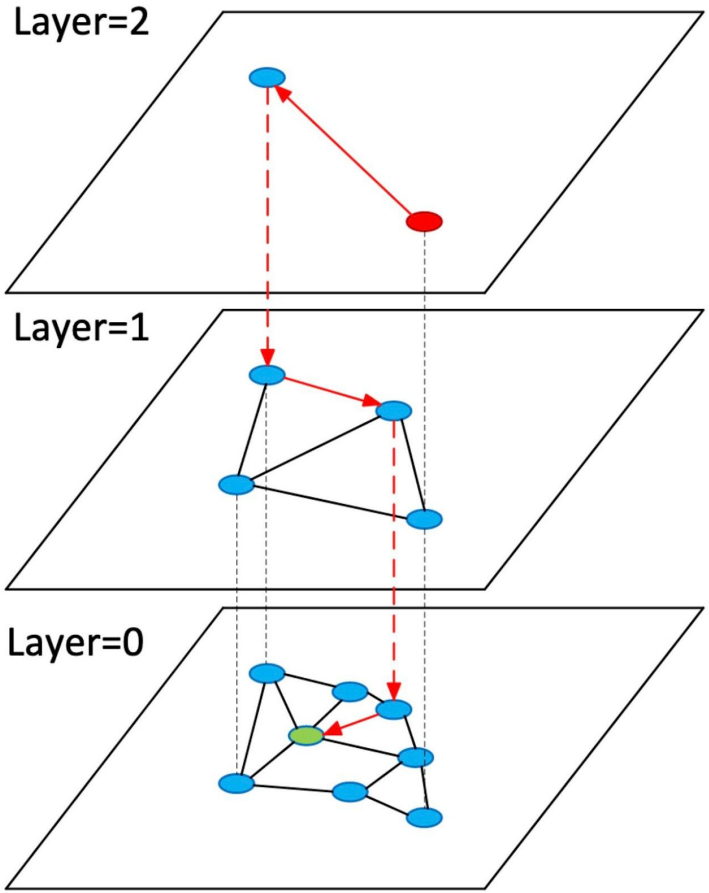
이를 극복하기 위해서 정확도를 약간 희생하지만, 빠른 검색속도를 제공하는
근사 최근접 이웃(ANN, Approximate Nearest Neighbor) 알고리즘이 사용된다.
- 빠른 검색 속도
- 대용량 데이터셋에서도 효율적인 작동

- 정확도의 일부분 포기
- KNN에 비하여 더 복잡한 데이터 구조와 알고리즘 ⇒ 복잡도 증가
- 일부 ANN 알고리즘에서는 메모리 사용량이 증가할 수도 있다.
- 최적성능을 위한 매개변수 설정 하는 최적화 필요



참고) *벡터 검색 엔진 최적화 방법 - HNSW*

▼ 현재 가장 널리 쓰이는 ANN 알고리즘인
HNSW(Hierarchical Navigable Small World)



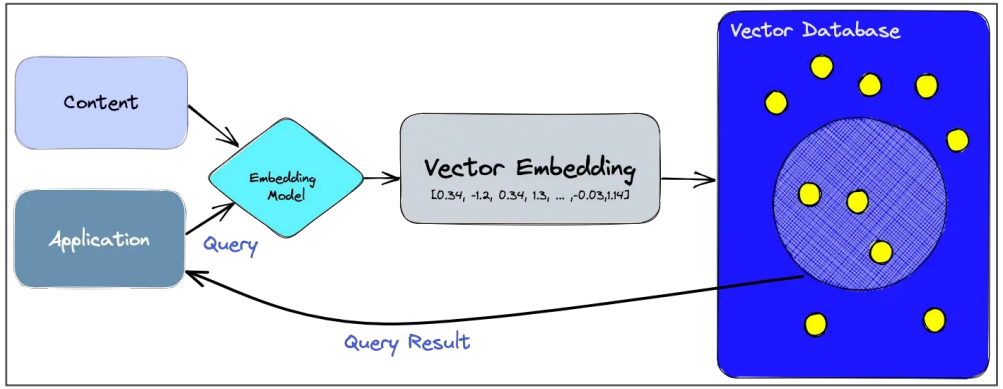
Decreasing characteristic radius

HNSW

- 그래프 기반
  : 각 벡터를 하나의 노드로 보고, 작게 연결된 세계(Small World Group)로 구성

- 상위 레벨에서 대략적인 검색방향을 잡고
  하위 레벨로 내려가면서 정밀한 탐색을 하는 방식

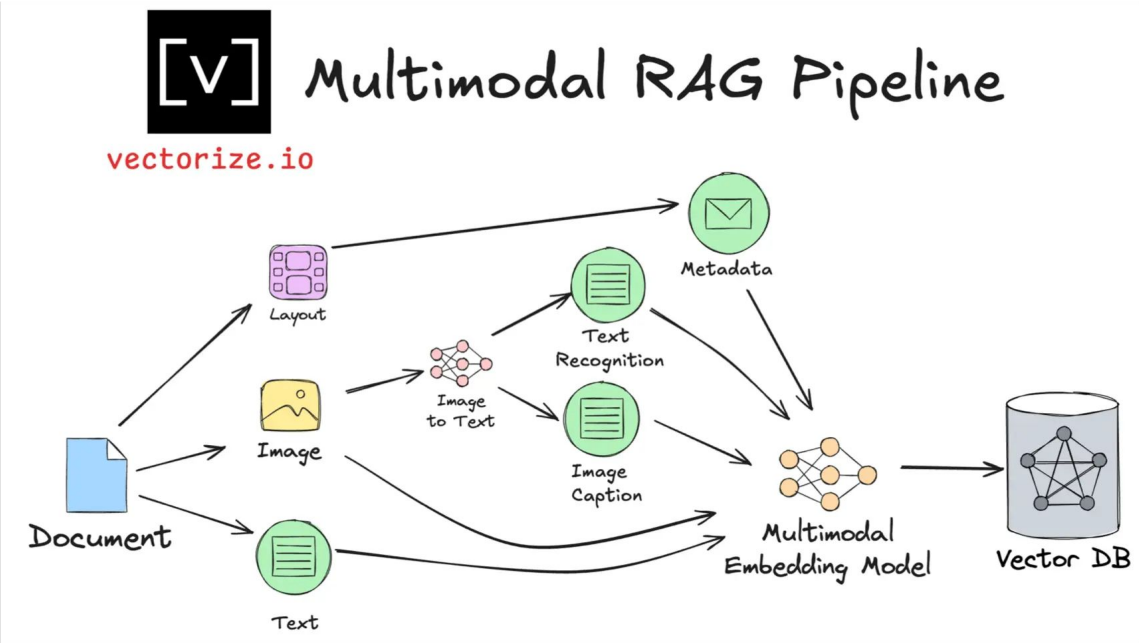
- 매우 빠르며, 높은 정확도를 보인다.
- 메모리 기반 검색에 강력한 성능을 보인다.
  : 그래프 → random access,
  연결 리스트 자료구조 형식의 그래프 : 디스크가 아닌 RAM에서 빠르게 동작
  적은 계산량에 비해 많은 메모리 탐색색



참고) *What is a Vector Database & How Does it Work?*

참고) *Understanding HNSW for Vector Search - Milvus*

참고) 멀티모달 RAG를 구성하는 패턴

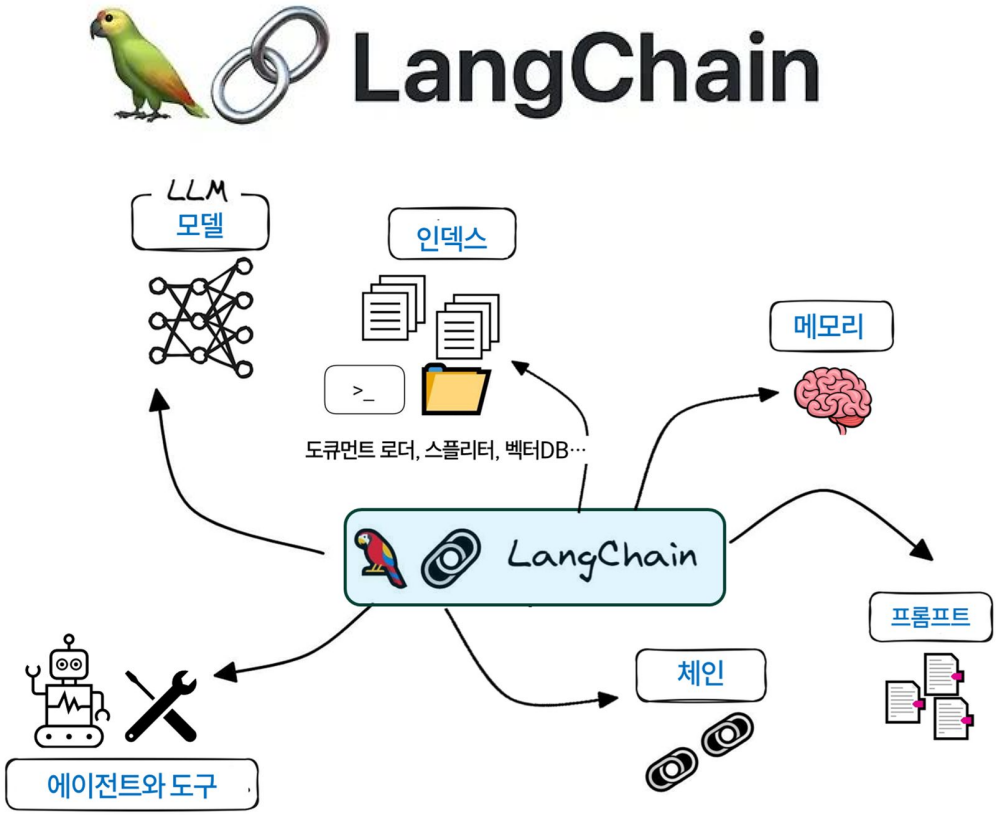


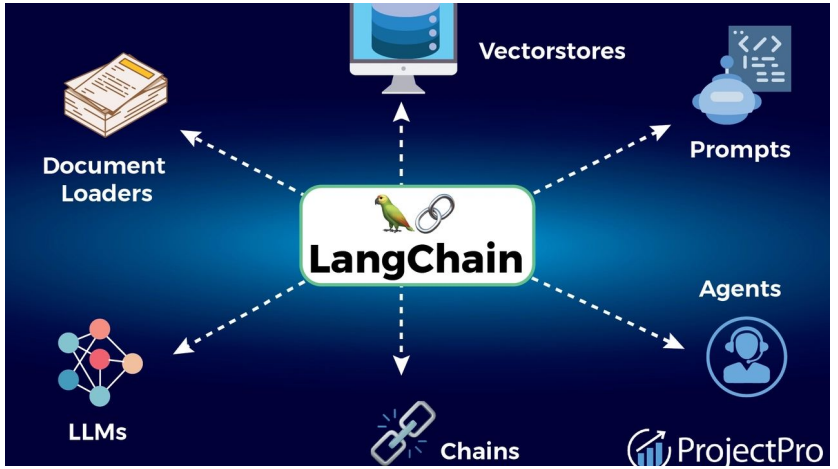
멀티모달 RAG 시스템은
여러 입력 유형(modality)을 처리한다는 점을 제외하면
일반적인 RAG와 유사하다.

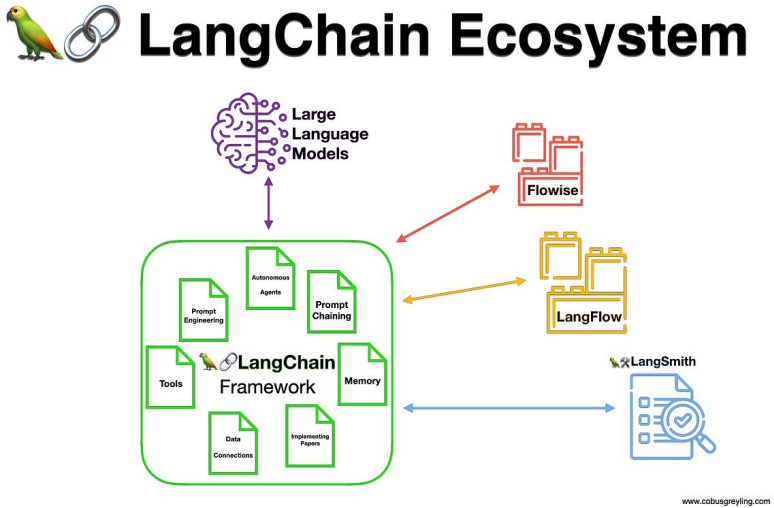
- 단. 이후 답변 생성과정에서
  원본 유형 데이터를 사용할지 여부에 대하여
  고려할 필요가 있다.

참고) *Multimodal RAG Patterns Every AI Developer Should Know*

※ LangChain을 통해 알아보는 RAG 구현의 구성요소







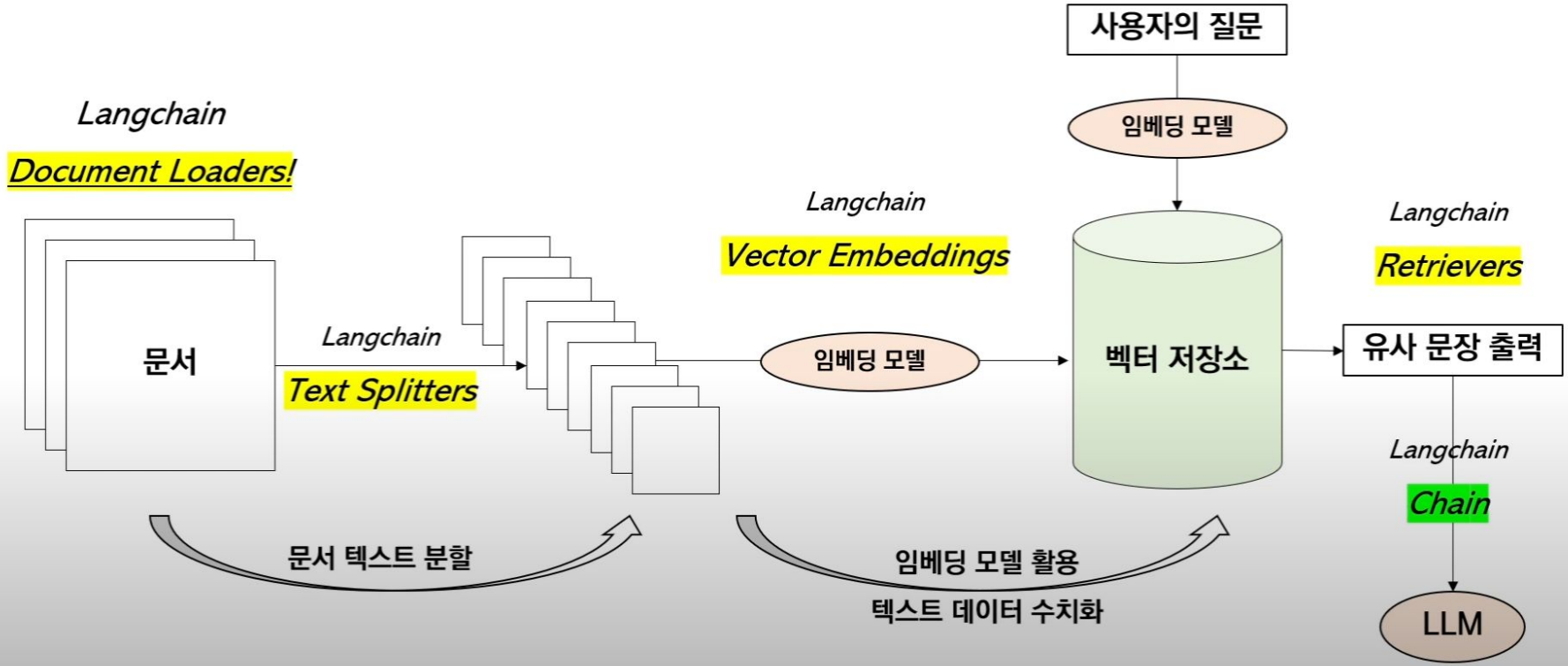
참고) [LangChain 한국어 튜토리얼 - RAG 기능별 모듈]

RAG 프로세스

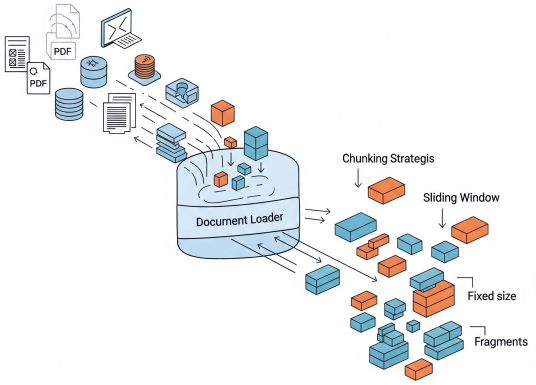
문서 로드 및 Vector DB 저장

1	문서 로드(Load)	문서(pdf, word), RAW DATA, 웹페이지, Notion 등의 데이터를 읽기
2	분할(Split)	불러온 문서를 chunk 단위로 분할
3	임베딩(Embedding)	문서를 벡터 표현으로 변환
4	벡터DB(VectorStore)	변환된 벡터를 DB에 저장

랭체인의 **Retrieval**은 RAG의 대부분의 구성 요소를 아우르며, 구성 요소 하나하나가 RAG의 품질을 좌우



LangChain



Document Loaders Chunking Strategis

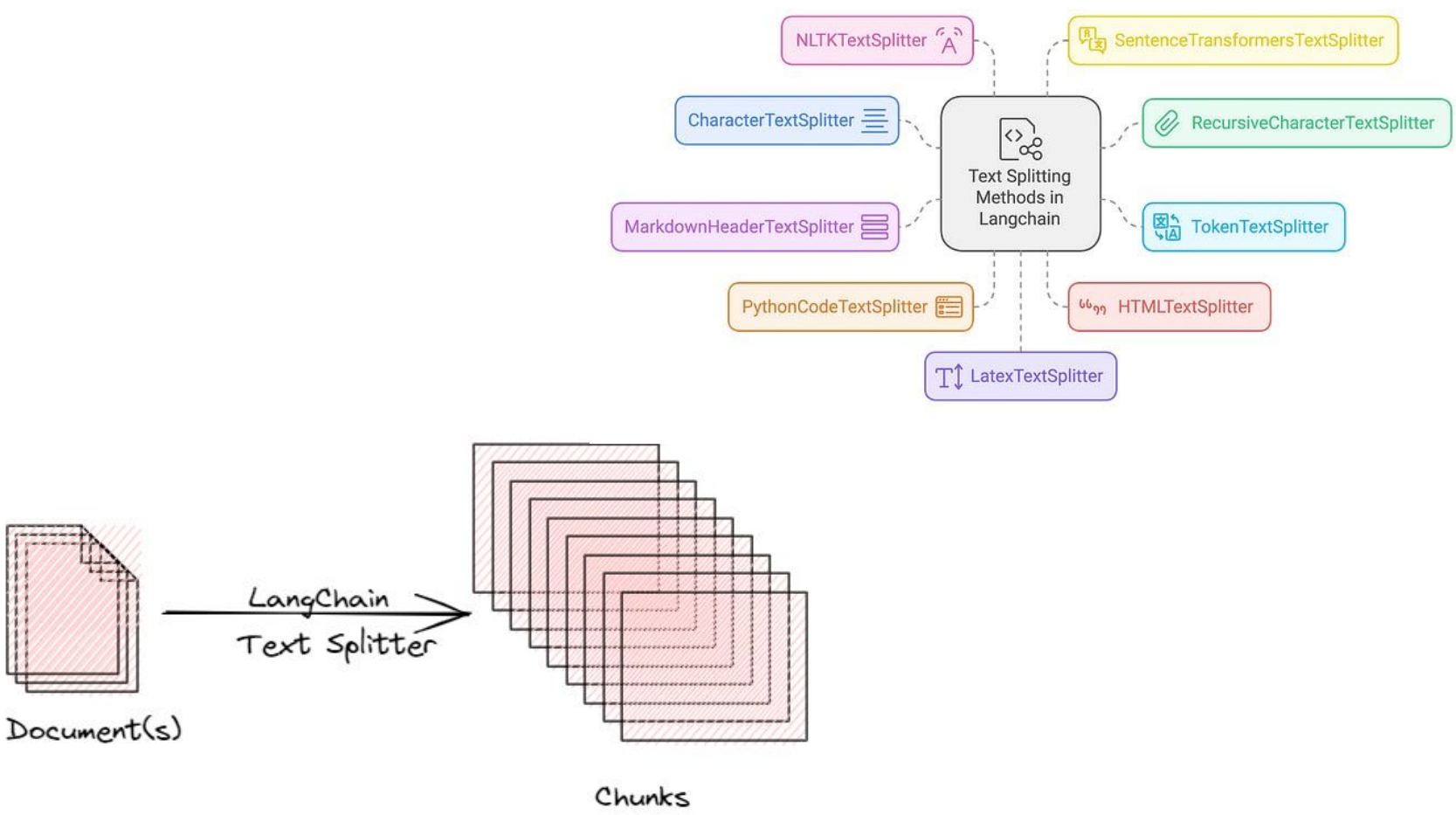
Document Loaders는 다양한 형태의 문서를 RAG 전용 객체로 불러들이는 모듈

Document Loader

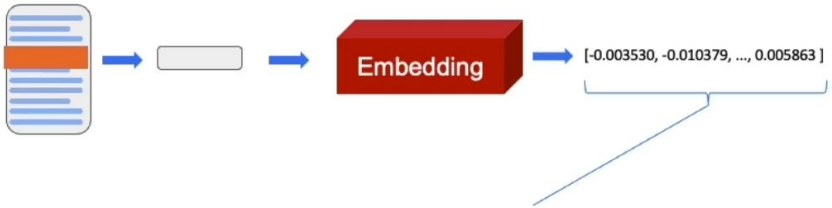
Page_content: 문서의 내용

Metadata: 문서의 위치, 제목, 페이지 넘버 등

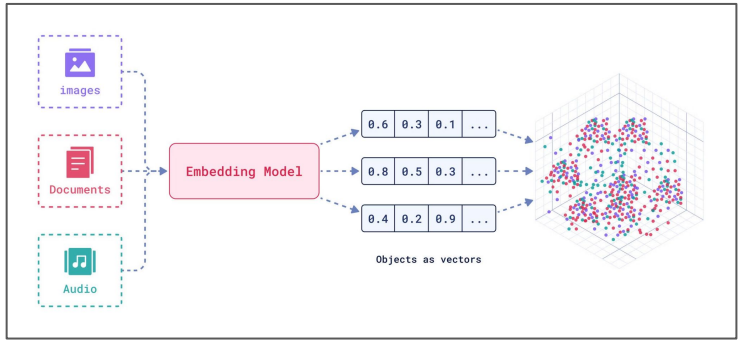
```
{'answer': ' The president honored Justice Breyer for his service and mentioned his legacy of excellence.\n',
 'sources': '31-pl'}
```


NLTKTextSplitter
SentenceTransformersTextSplitter
CharacterTextSplitter
Text Splitting Methods in Langchain
RecursiveCharacterTextSplitter
MarkdownHeaderTextSplitter
TokenTextSplitter
PythonCodeTextSplitter
HTMLTextSplitter
LatexTextSplitter
LangChain
Text Splitter
Document(s)
Chunks

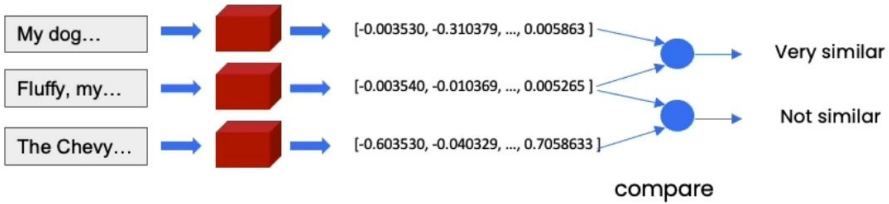
Embeddings



- Embedding vector captures content/meaning
- Text with similar content will have similar vectors



1) My dog Rover likes to chase squirrels.
2) Fluffy, my cat, refuses to eat from a can.
3) The Chevy Bolt accelerates to 60 mph in 6.7 seconds.



RAG 프로세스

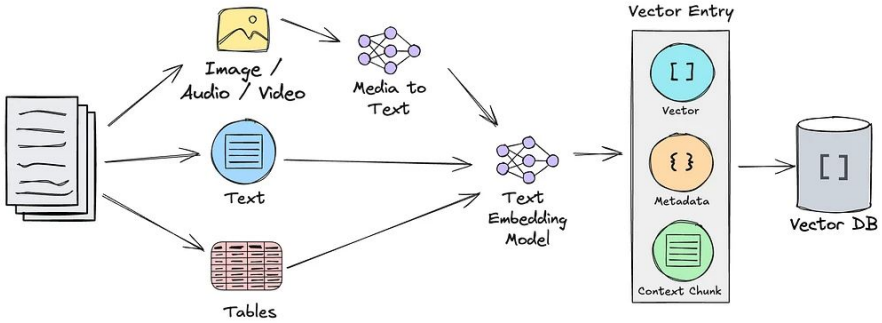
문서 검색 및 결과 도출

5	**검색(Retrieval)**	유사도 검색(similarity, mmr), Multi-Query, Multi-Retriever
6	**프롬프트(Prompt)**	검색된 결과를 바탕으로 원하는 결과를 도출하기 위한 프롬프트
7	**모델(LLM)**	모델 선택(GPT-3.5, GPT-4, etc)
8	**결과(Output)**	텍스트, JSON, 마크다운

vectorize.io

Multimodal RAG Pattern 1

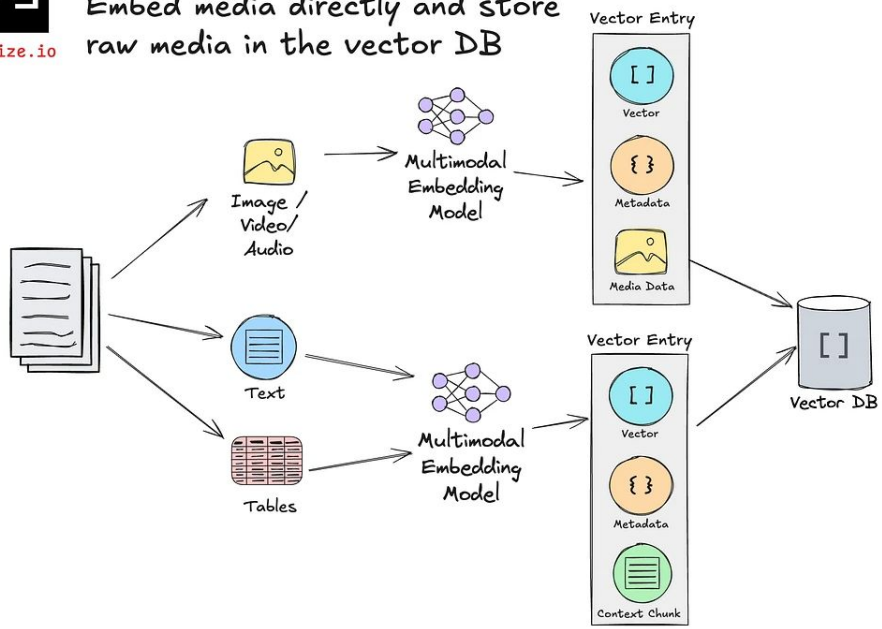
Embed media descriptions as text



vectorize.io

Multimodal RAG Pattern 2

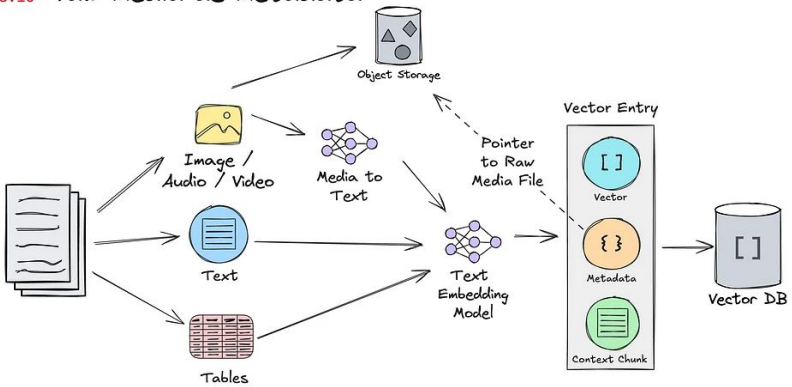
Embed media directly and store raw media in the vector DB

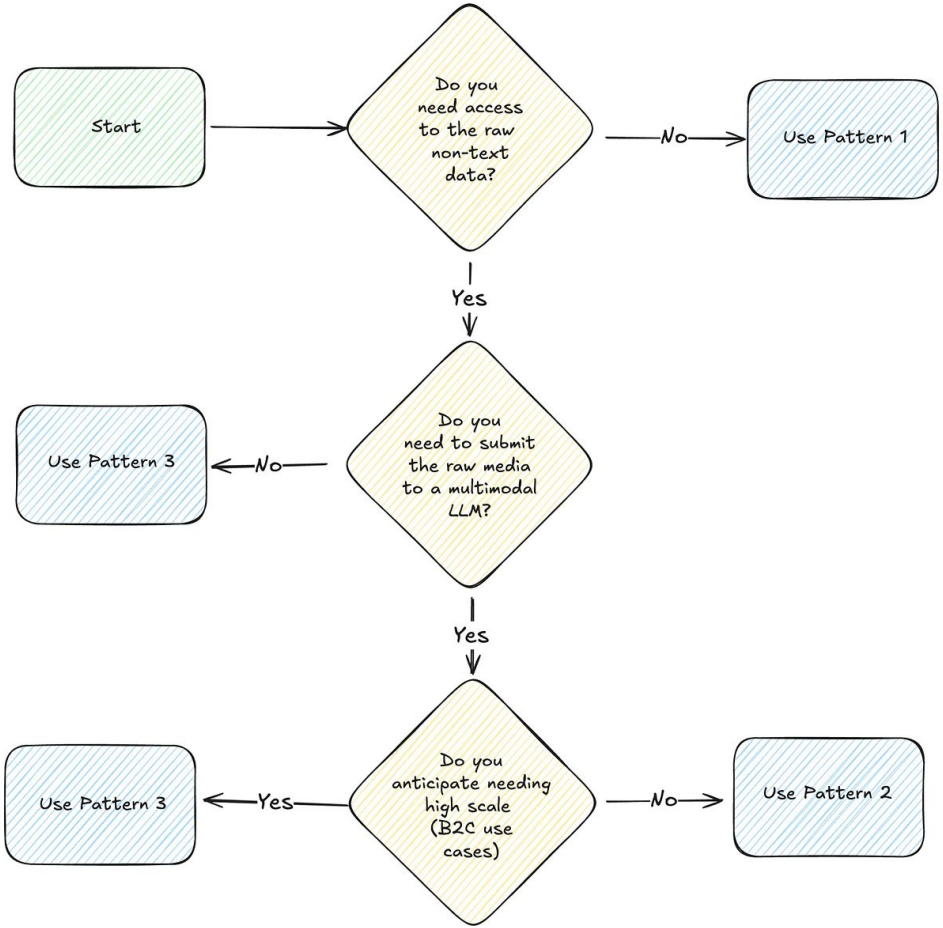


vectorize.io

Multimodal RAG Pattern 3

Embed media description as text with pointer to raw media as metadata





참고) *Mastering RAG : How to Select an Embedding Model*

Types Of Embedding Models For RAG

Model type	Embeddings	Example model	Compute	Retrieval performance	Granularity/input
Sparse	0.3 0.8	SPLADE	Low	Good	Sentence, paragraph
Dense	0.3 0.1 0.8	Sentence transformers	Medium	Good	Sentence, paragraph
Multivector	0.4 0.3 0.2 x tokens	COLBERT	High	Best	Sentence, paragraph
Long context dense	0.3 0.1 0.8	text-embedding-3-small	Medium	Good	Paragraphs, chapters
Variable dimension	0.3 0.1 0.8 Dimensions can change	text-embedding-3-small	Medium	Good	Sentence, paragraph
Code (dense)	0.3 0.1 0.8	text-embedding-3-small	Medium	Good	Functions, classes

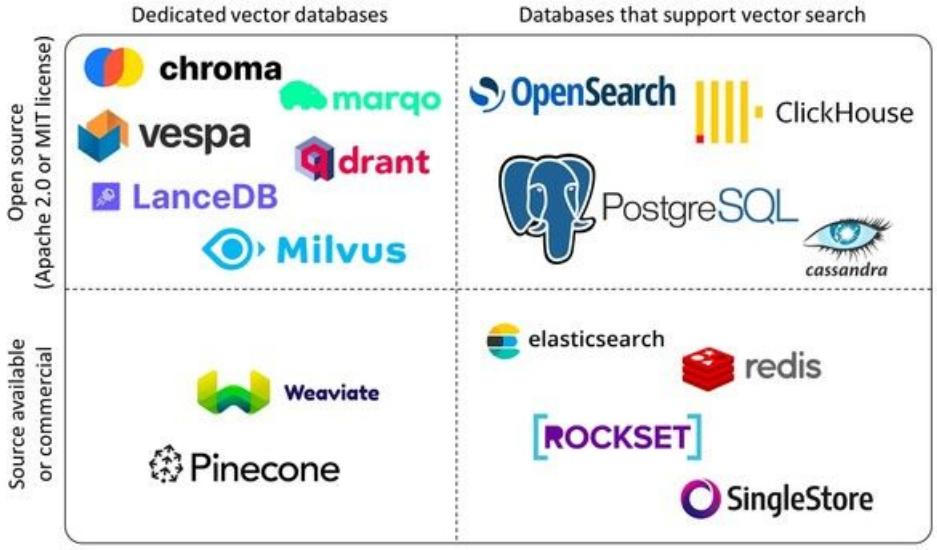
Galileo

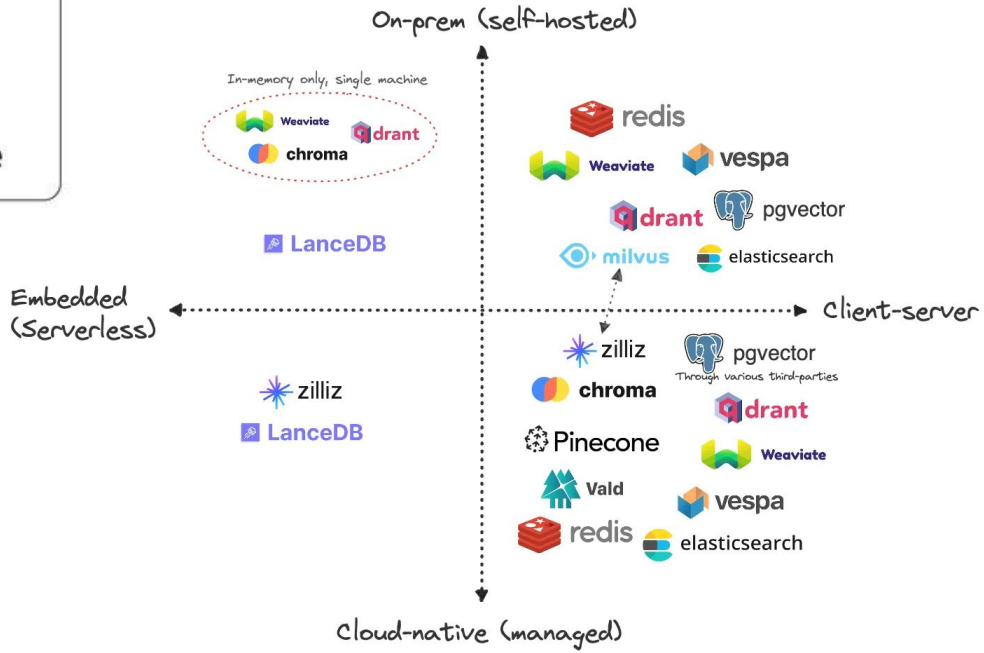
Embedding Price Comparison

Provider	Model	input price per 1K Token	input price per 1M Token	Compared to Open AI Ada	Compared to Cohere
Amazon Bedrock	Titan Embeddigs	$0.00010	$0.10000	0.00%	0.00%
(Azure) OpenAI	Ada Embeddings	$0.00010	$0.10000	0.00%	0.00%
	Embedding 3 small	$0.00002	$0.02000	-80.00%	-80.00%
	Embedding 3 Large	$0.00013	$0.13000	30.00%	30.00%
Cohere	Embed	$0.00010	$0.10000	0.00%	0.00%
Google Vertex AI 1 token ~= 4 chars	Text Embeddings	$0.00040	$0.40000	300.00%	300.00%
Together	BGE-Base	$0.00003	$0.02800	-72.00%	-72.00%
Anyscale	thenlper-gte-large	$0.00005	$0.05000	-50.00%	-50.00%
MosaicML	Instructor-Large	$0.00010	$0.10000	0.00%	0.00%
	Instructor-XL	$0.00020	$0.20000	100.00%	100.00%

Galileo

Source: https://docs.google.com/spreadsheets/d/1NX8ZW9Jnfpy88PC2d6Bwfa87JRiv3GTeqwXoB4mKU_s


Dedicated vector databases
Databases that support vector search
Open source (Apache 2.0 or MIT license)
Source available or commercial
chroma
marqo
vespa
qdrant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
elasticsearch
redis
Weaviate
ROCKSET
Pinecone
SingleStore


On-prem (self-hosted)
In-memory only, single machine
Weaviate
qdrant
chroma
LanceDB
redis
Weaviate
vespa
qdrant
pgvector
milvus
elasticsearch
Embedded (Serverless)
Client-server
zilliz
LanceDB
zilliz
pgvector
Through various third-parties
chroma
qdrant
Pinecone
Weaviate
Vald
vespa
redis
elasticsearch
Cloud-native (managed)

(계속...)

Q & A