

RAG 검색 증강 생성

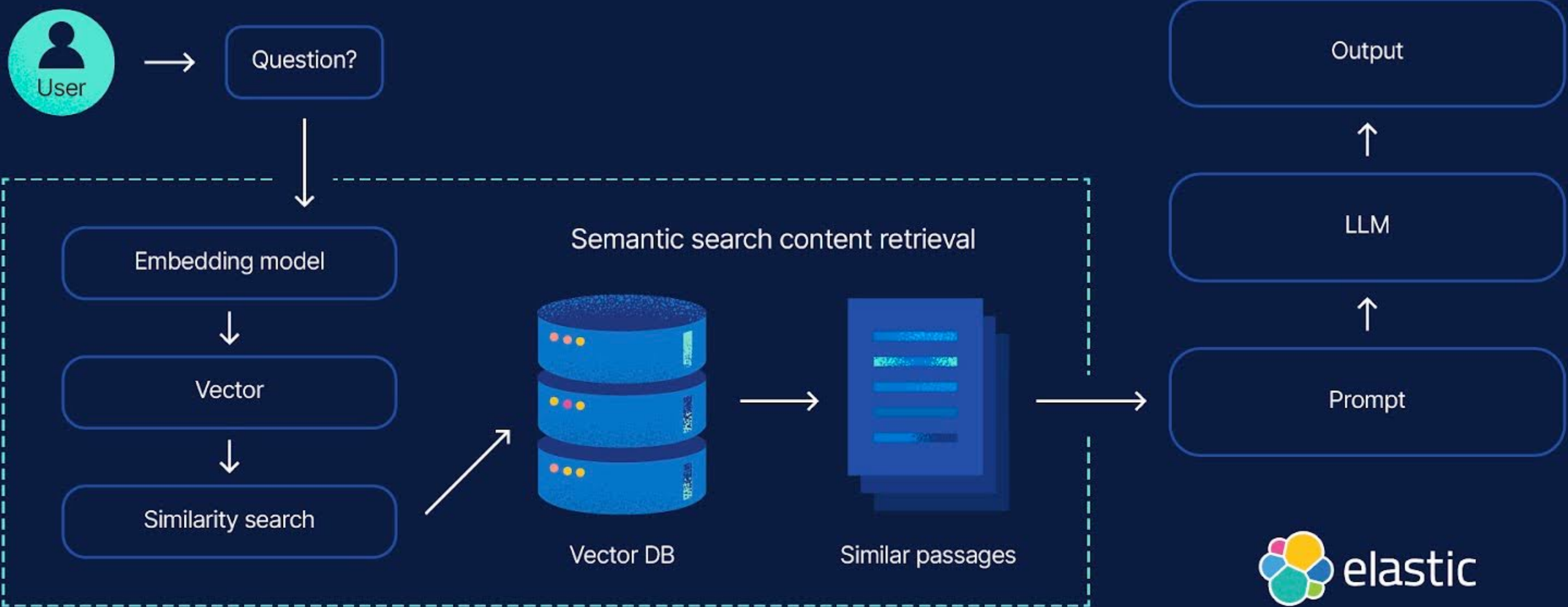
RAG란

RAG

Retrieval Augmented Generation

검색 증강 생성

Retrieval augmented generation (RAG) in action

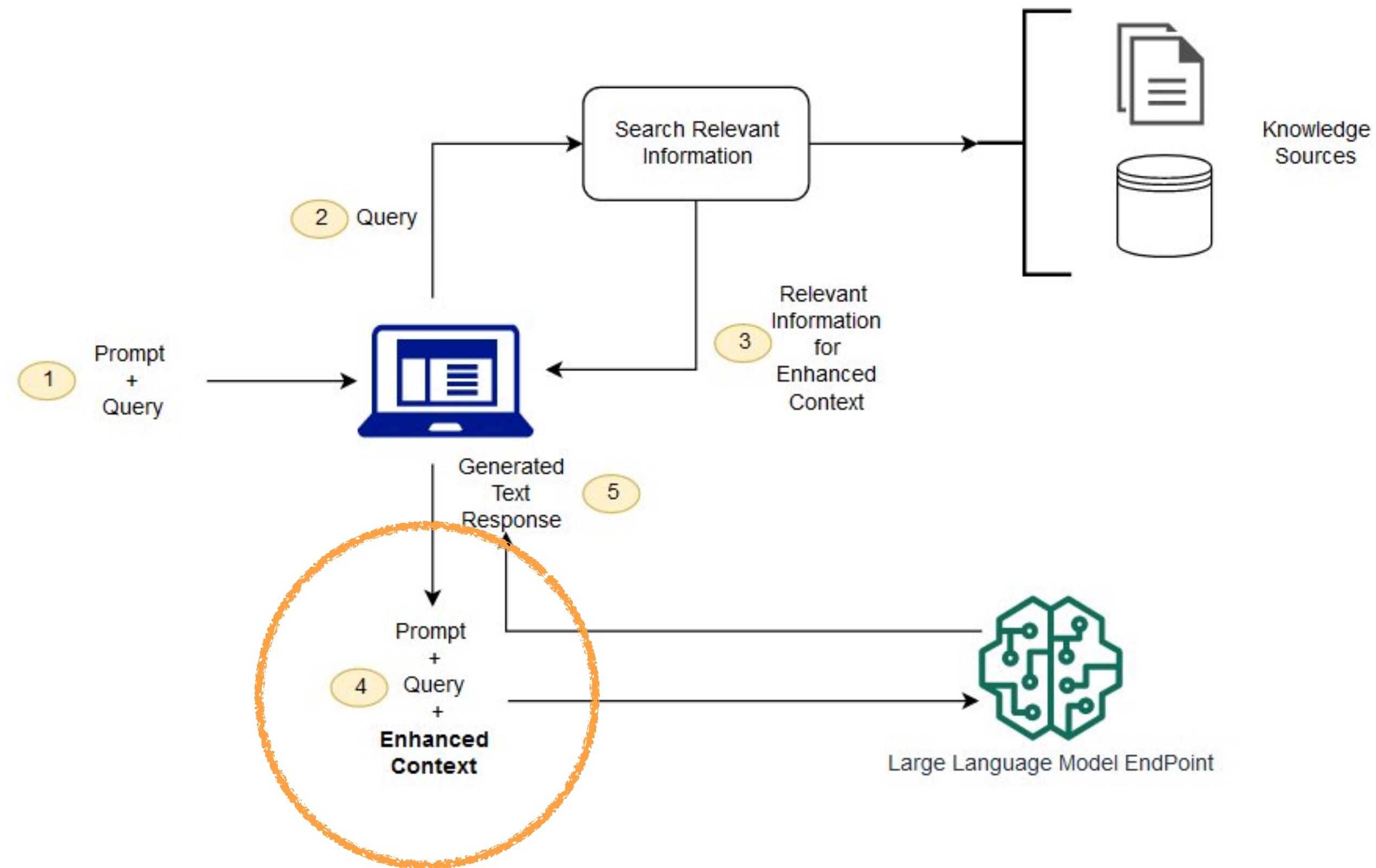


Retrieval



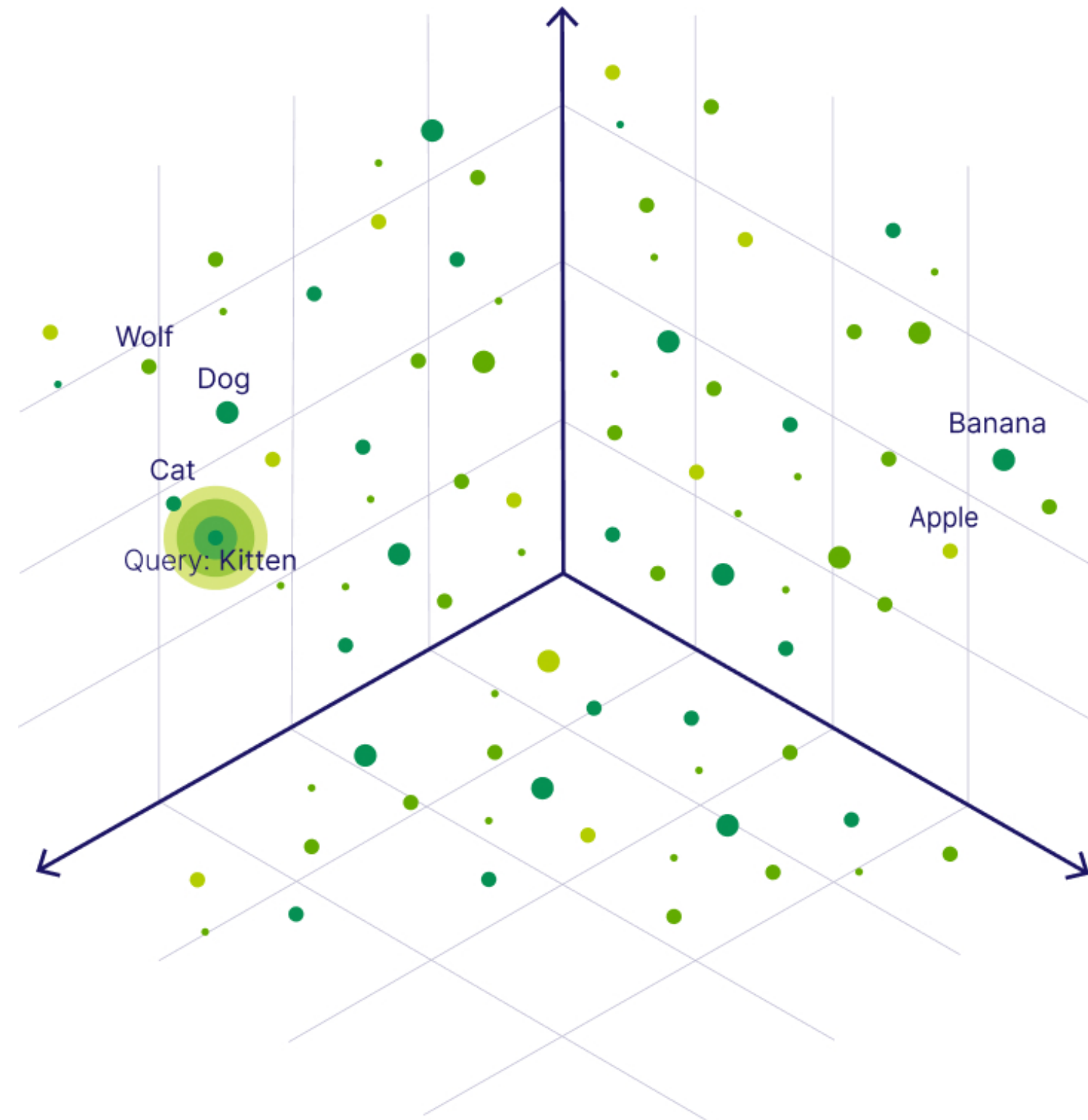
언어 모델(LLM)이 가지고 있지 않은, 아웃풋을 만드는 데에 필요한 정보를 제공

Augmented

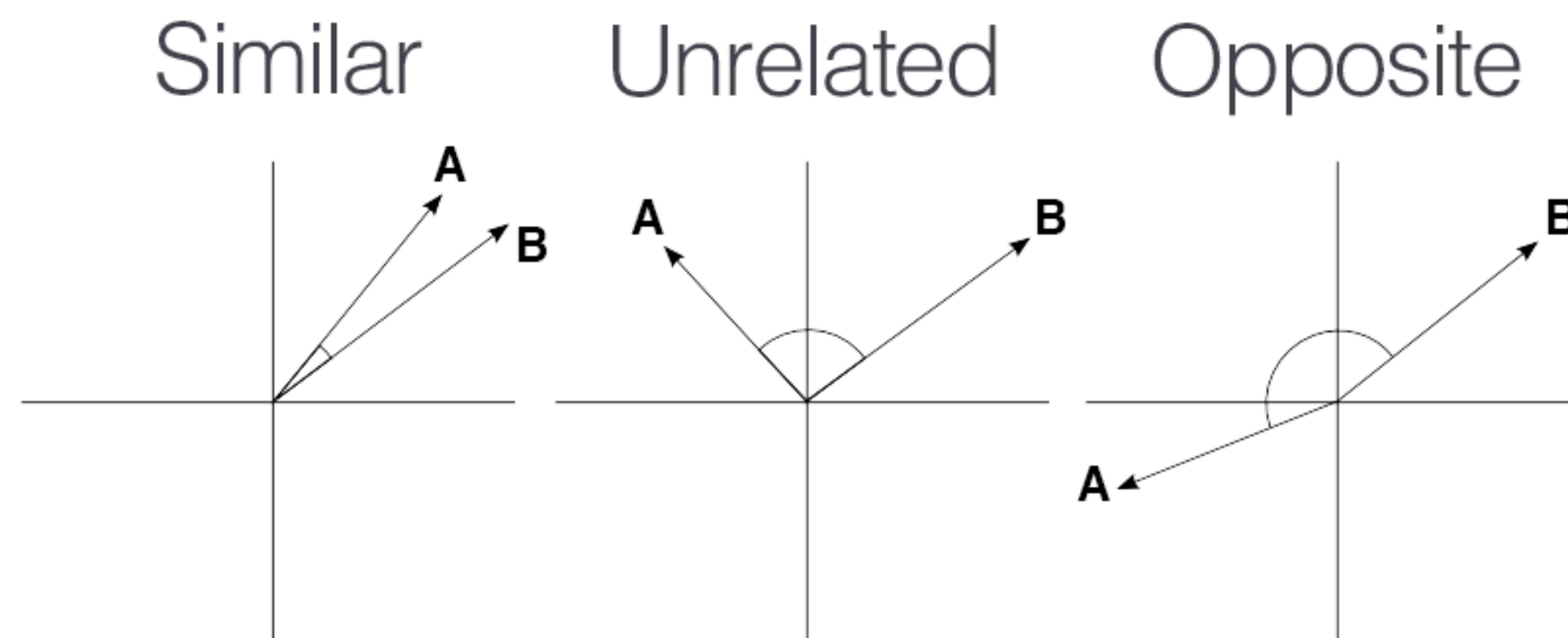


Retrieval된 데이터를 LLM에게 전달

Vector Database



Vector Database



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

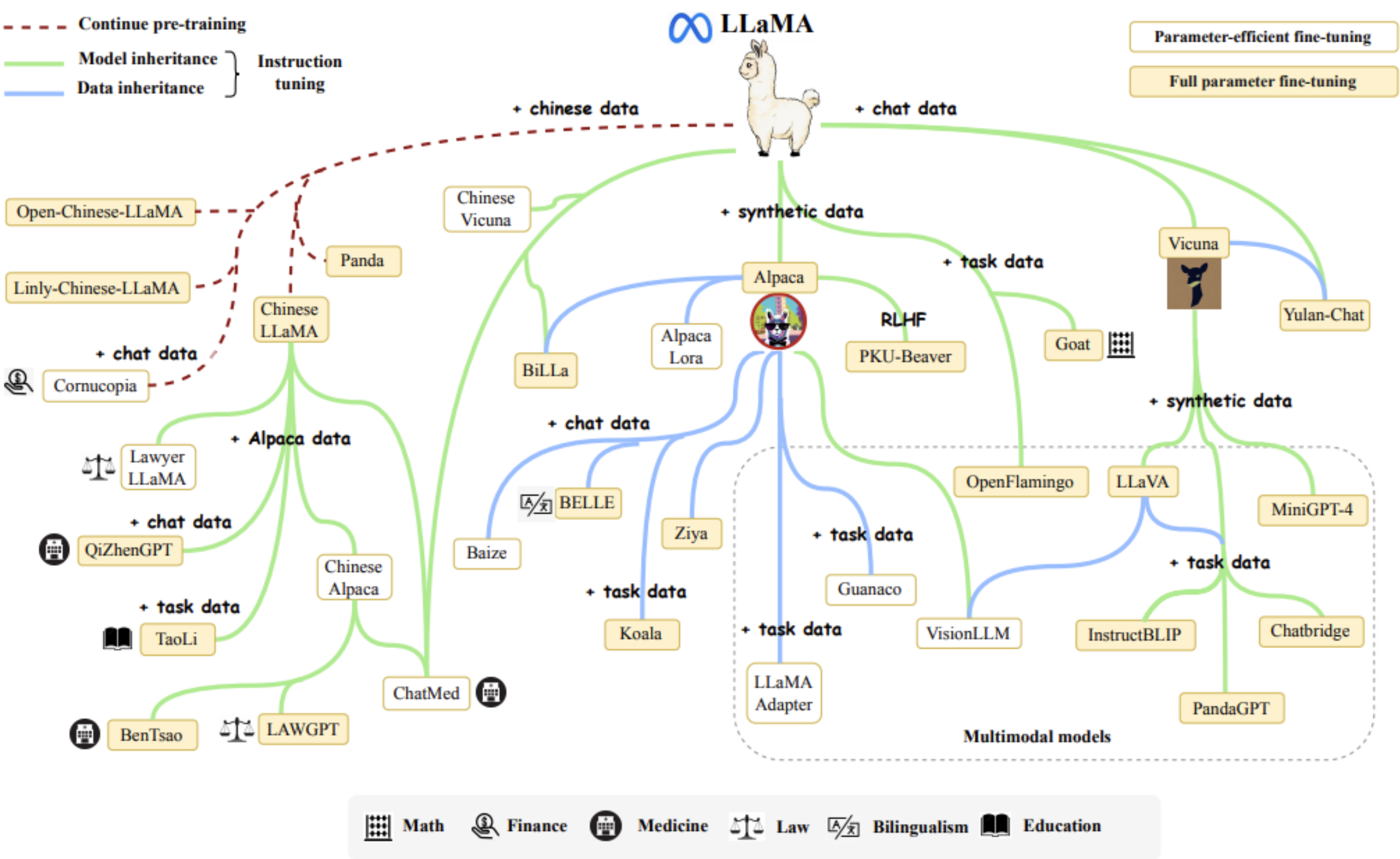
LangChain이란?



LangChain

추상화하여 LLM과 원활하게 작동하도록 설계된 SW
“프로세스 단순화”

LLM Model



NLP

허프만 코드 이해가 안 가는데 $\pi\pi$
고정 길이 코드는 모두 8bit를 가져야 하는데,
가변 길이 코드는 1bit를 가져도 되고 nbit를 가져도 돼서 이게 더 가성비있다는
건가?



네, 맞아요! 허프만 코드(Huffman coding)는 가변 길이 코드의 예로, 고정 길이 코드와 비교했을 때 더 효율적으로 데이터를 압축할 수 있습니다.

고정 길이 코드는 모든 문자가 고정된 비트 수를 갖습니다. 예를 들어, 8비트로 표현된다면, 모든 문자는 8비트로 인코딩됩니다. 이렇게 되면, 문자들의 출현 빈도에 관계없이 모두 동일한 크기의 공간을 차지하게 되죠.

반면, 허프만 코드는 빈도가 높은 문자일수록 짧은 코드를, 빈도가 낮은 문자일수록 긴 코드를 부여합니다. 예를 들어, "a"라는 문자가 자주 나온다면 "a"는 1비트나 2비트로 표현될 수 있고, 드물게 나오는 "z"는 더 긴 비트 수로 표현될 수 있습니다.

이렇게 가변 길이 코드를 사용하면, 자주 등장하는 문자들은 짧은 비트로 압축되어 전체적으로 더 적은 비트 수를 사용하게 되고, 덜 자주 등장하는 문자들은 더 많은 비트를 사용하지만 전체적으로 더 효율적인 압축을 할 수 있게 됩니다.

즉, 허프만 코드는 가변 길이 코드를 사용함으로써 자주 등장하는 데이터는 더 적은 비트로 압축하고, 덜 자주 등장하는 데이터는 더 많은 비트로 압축하여 전체적인 데이터 크기를 줄이는 방식입니다. 이로 인해 가성비가 더 뛰어나게 됩니다!

🔊 📄 🌟 🔒 ↺ ↻

NLP

NLU 자연어 이해 / NLG 자연어 생성

Langchain 간단한 짭밥..

```
[1] #환경변수 불러오기
    from dotenv import load_dotenv

    load_dotenv()
    ✓ 0.0s Python

... True

[6] from langchain_upstage import ChatUpstage

    llm = ChatUpstage()
    ai_message = llm.invoke("두통이랑 카페인이랑 상관관계가 있어?")
    ✓ 1.5s Python

▶ ▼ ai_message.content
[7] ✓ 0.0s Python

... '네, 카페인은 두통을 유발할 수 있습니다. 카페인은 혈관을 수축시키는 효과가 있어, 일부 사람들은 카페인을 섭취하면 두통이 발생할 수 있습니다. 또한, 카페인을 습관적으로 섭취하다가 갑자기 중단하면 카페인 금단 증상으로 인해 두통
```