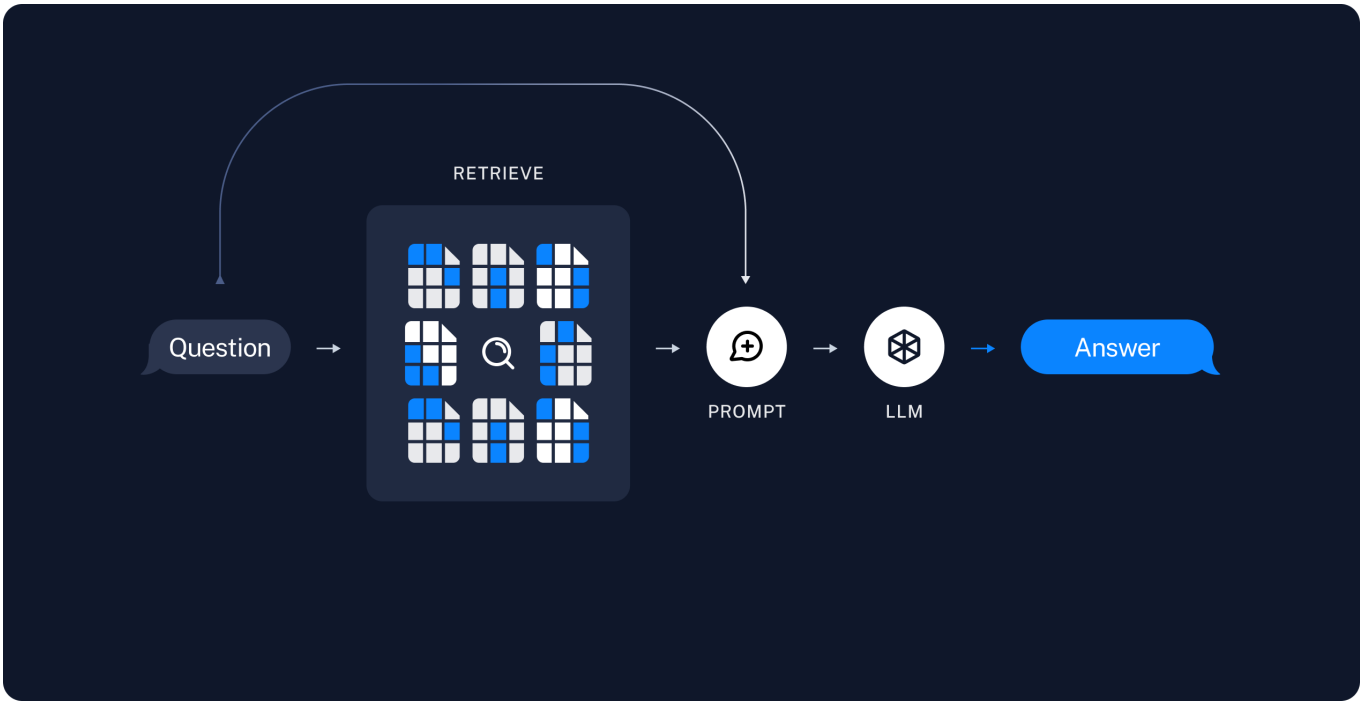
검색기 (Retriever)

Fast campus



검색기(Retriever) 단계는 Retrieval-Augmented Generation(RAG) 시스템의 다섯 번째 단계로, 저장된 벡터 데이터베이스에서 **사용자의 질문과 관련된 문서를 검색하는 과정**입니다. 이 단계는 사용자 질문에 **가장 적합한 정보를 신속하게 찾아내는 것이 목표**이며, RAG 시스템의 전반적인 성능과 직결되는 매우 중요한 과정입니다.

검색기의 필요성

1. **정확한 정보 제공**: 검색기는 사용자의 질문과 **가장 관련성 높은 정보를 검색**하여, 시스템이 **정확하고 유용한 답변을 생성**할 수 있도록 합니다. 이 과정이 효과적으로 이루어지지 않으면, 결과적으로 제공되는 **답변의 품질이 떨어질 수** 있습니다.
2. **응답 시간 단축**: 효율적인 검색 알고리즘을 사용하여 데이터베이스에서 적절한 **정보를 빠르게 검색**함으로써, 전체적인 **시스템 응답 시간을 단축**시킵니다.

사용자 경험 향상에 직접적인 영향을 미칩니다.
3. **최적화**: 효과적인 검색 과정을 통해 **필요한 정보만을 추출**함으로써 시스템 자원의 사용을 최적화하고, **불필요한 데이터 처리를 줄일 수** 있습니다.

동작 방식

1. **질문의 벡터화**: 사용자의 질문을 벡터 형태로 변환합니다. 이 과정은 임베딩 단계와 유사한 기술을 사용하여 진행됩니다. 변환된 질문 벡터는 후속 검색 작업의 기준점으로 사용됩니다.
2. **벡터 유사성 비교**: 저장된 문서 벡터들과 질문 벡터 사이의 **유사성을 계산**합니다. 이는 주로 코사인 유사성(cosine similarity), Max Marginal Relevance(MMR) 등의 수학적 방법을 사용하여 수행됩니다.
3. **상위 문서 선정**: 계산된 유사성 점수를 기준으로 **상위 N개의 가장 관련성 높은 문서를 선정**합니다. 이 문서들은 다음 단계에서 사용자의 질문에 대한 답변을 생성하는 데 사용됩니다.
4. **문서 정보 반환**: 선정된 문서들의 정보를 다음 단계(프롬프트 생성)로 전달합니다. 이 정보에는 문서의 내용, 위치, 메타데이터 등이 포함될 수 있습니다.

검색기의 중요성

검색기는 RAG 시스템에서 **정보 검색의 질을 결정하는 핵심적인 역할**을 합니다. 효율적인 검색기 없이는 대규모 데이터베이스에서 관련 정보를 신속하고 정확하게 찾아내는 것이 매우 어렵습니다.

또한, 검색기는 **사용자 질문에 대한 적절한 컨텍스트를 제공**하여, 언어 모델이 보다 **정확한 답변을 생성**할 수 있도록 돕습니다. 따라서 검색기의 성능은 **RAG 시스템의 전반적인 효율성과 사용자 만족도에 직접적인 영향**을 미칩니다.

Sparse Retriever & Dense Retriever

Sparse Retriever와 **Dense Retriever**는 정보 검색 시스템에서 사용되는 두 가지 주요 방법입니다. 이들은 자연어 처리 분야, 특히 대규모 문서 집합에서 관련 문

서를 검색할 때 사용됩니다.

Sparse Retriever

Sparse Retriever는 문서와 쿼리(질문)를 이산적인 **키워드 벡터로 변환하여 처리**합니다. 이 방법은 주로 텀 빈도-역문서 빈도(TF-IDF)나 BM25와 같은 전통적인 정보 검색 기법을 사용합니다.

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: 단어가 문서에 나타나는 빈도와 그 단어가 몇 개의 문서에서 나타나는지를 반영하여 단어의 중요도를 계산합니다. 여기서, 자주 나타나면서도 문서 집합 전체에서 드물게 나타나는 단어가 높은 가중치를 받습니다.
- **BM25**: TF-IDF를 개선한 모델로, 문서의 길이를 고려하여 검색 정확도를 향상시킵니다. 긴 문서와 짧은 문서 간의 가중치를 조정하여, 단어 빈도의 영향을 상대적으로 조절합니다.

Sparse Retriever의 특징은 각 단어의 존재 여부만을 고려하기 때문에 계산 비용이 낮고, 구현이 간단하다는 점입니다. 그러나 이 방법은 단어의 의미적 연관성을 고려하지 않으며, **검색 결과의 품질이 키워드의 선택에 크게 의존**합니다.

Dense Retriever

Dense Retriever는 최신 딥러닝 기법을 사용하여 문서와 쿼리를 연속적인 고차원 벡터로 인코딩합니다. Dense Retriever는 문서의 의미적 내용을 보다 풍부하게 표현할 수 있으며, **키워드가 완벽히 일치하지 않더라도 의미적으로 관련된 문서를 검색**할 수 있습니다.

Dense Retriever는 벡터 공간에서의 거리(예: 코사인 유사도)를 사용하여 **쿼리와 가장 관련성 높은 문서**를 찾습니다. 이 방식은 특히 언어의 뉘앙스와 문맥을 이해하는 데 유리하며, **복잡한 쿼리에 대해 더 정확한 검색 결과를 제공**할 수 있습니다.

차이점

1. **표현 방식**: Sparse Retriever는 이산적인 키워드 기반의 표현을 사용하는 반면, Dense Retriever는 연속적인 벡터 공간에서 의미적 표현을 사용합니다.
2. **의미적 처리 능력**: Dense Retriever는 문맥과 의미를 더 깊이 파악할 수 있어, 키워드가 정확히 일치하지 않아도 관련 문서를 검색할 수 있습니다. Sparse Retriever는 이러한 의미적 뉘앙스를 덜 반영합니다.
3. **적용 범위**: 복잡한 질문이나 자연어 쿼리에 대해서는 Dense Retriever가 더 적합할 수 있으며, 간단하고 명확한 키워드 검색에는 Sparse Retriever가 더 유용할 수 있습니다.

이 두 방법은 각각의 장단점을 가지고 있으며, 사용 사례와 요구 사항에 따라 적절히 선택되어야 합니다.

코드

Dense Retriever

```
from langchain_community.vectorstores import FAISS

# 단계 4: DB 생성(Create DB) 및 저장
# 벡터스토어를 생성합니다.
vectorstore = FAISS.from_documents(documents=split_documents,
embedding=embeddings)

# 단계 5: Dense Retriever 생성
# 문서에 포함되어 있는 정보를 검색하고 생성합니다.
faiss_retriever = vectorstore.as_retriever()
```

Sparse Retriever

```
from langchain_community.retrievers import BM25Retriever

# 단계 5: Sparse Retriever 생성
# 문서에 포함되어 있는 정보를 검색하고 생성합니다.
```

```
bm25_retriever =
BM25Retriever.from_documents(split_documents)
```

참고

- 벡터저장소 지원 검색기
- LangChain Retrievers