2026.02.25

김태훈

RAG Implementation

RAG 기능 및 활용

목차

01

RAG 개요

기존 LLM 모델 한계:

- 지식 컷오프 (Knowledge cut-off)으로 인한 최신 정보 부재 및 사실이 아닌 내용을 그럴듯하게 말하는 환각 현상(Hallucination) 발생
- 비즈니스 제약: 대규모 데이터를 매번 재학습(Retraining)시키는 것은 비용과 시간 면에서 비효율적이며, 보안이 중요한 기업 데이터 처리에 취약함

↘

RAG: 검색 증강 생성 (Retrieval-Augmented Generation)

- 최신 정보: LLM을 따로 학습하지 않아도 업로드 되어있는 데이터를 참조하여 정확한 정보 제공
- 인증: 소스 추적 가능 기능으로 인해 환각 현상 확인이 가능

02

Tool 상세

API

통신 및 데이터 교환 프로토콜 접점

Huggingface

오픈소스 무료 사전학습 모델 활용 데이터

LangChain

계발단계를 위한 오픈소스 프레임워크

03. 아키텍쳐 및 작동원리

A. 전체 흐름

1

인덱싱 (Indexing)

문서를 로드하고, 의미 있는 단위로 분할(split) 한 뒤, 벡터(Vector) 형태로 변환하여 벡터 데이터 베이스에 저장 단계

2

리트리벌 (Retrieval)

쿼리 단계: 사용자 질문이 들어오면 이 또한 벡터화하여 DB에서 가장 유사한 ‘가까운 이웃‘ (Closest Neighbors)’ 문서를 찾아 LLM에 전달

3

생성 (Generation)

인덱싱, 리트리벌 단계 이후 LLM에 내용 전달 이후 답변 생성하여 유저가 검사 및 확인

03. 아키텍쳐 및 작동원리

B. 인덱싱 (Indexing)

1

인덱싱 (Indexing)

문서를 로드하고, 의미 있는 단위로 분할(split) 한 뒤, 벡터(Vector) 형태로 변환하여 벡터 데이터 베이스에 저장 단계

2

리트리벌 (Retrieval)

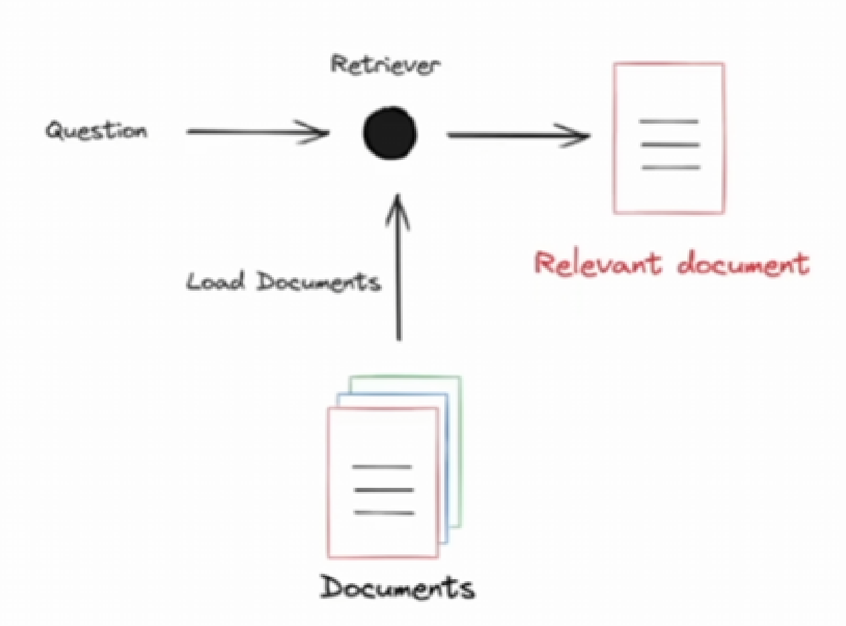
쿼리 단계: 사용자 질문이 들어오면 이 또한 벡터화하여 DB에서 가장 유사한 '가까운 이웃' (Closest Neighbors)' 문서를 찾아 LLM에 전달

3

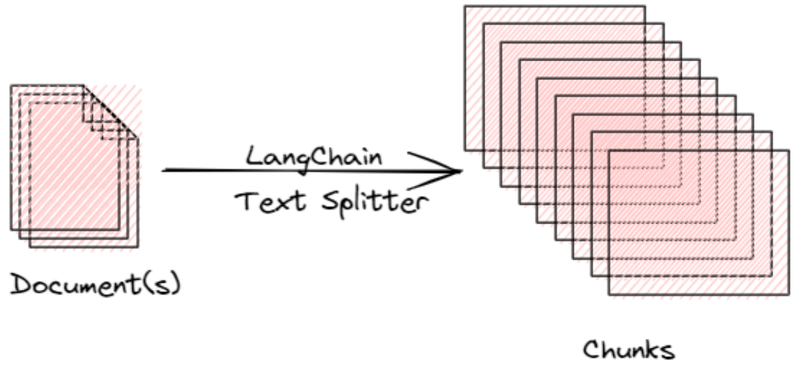
생성 (Generation)

인덱싱, 리트리벌 단계 이후 LLM에 내용 전달 이후 답변 생성하여 유저가 검사 및 확인

1. Document 업로드



2. 단위 분할



03. 아키텍쳐 및 작동원리

B. 인덱싱 (Indexing)

1
인덱싱 (Indexing)

문서를 로드하고, 의미 있는 단위로 분할(split) 한 뒤, 벡터(Vector) 형태로 변환하여 벡터 데이터 베이스에 저장 단계

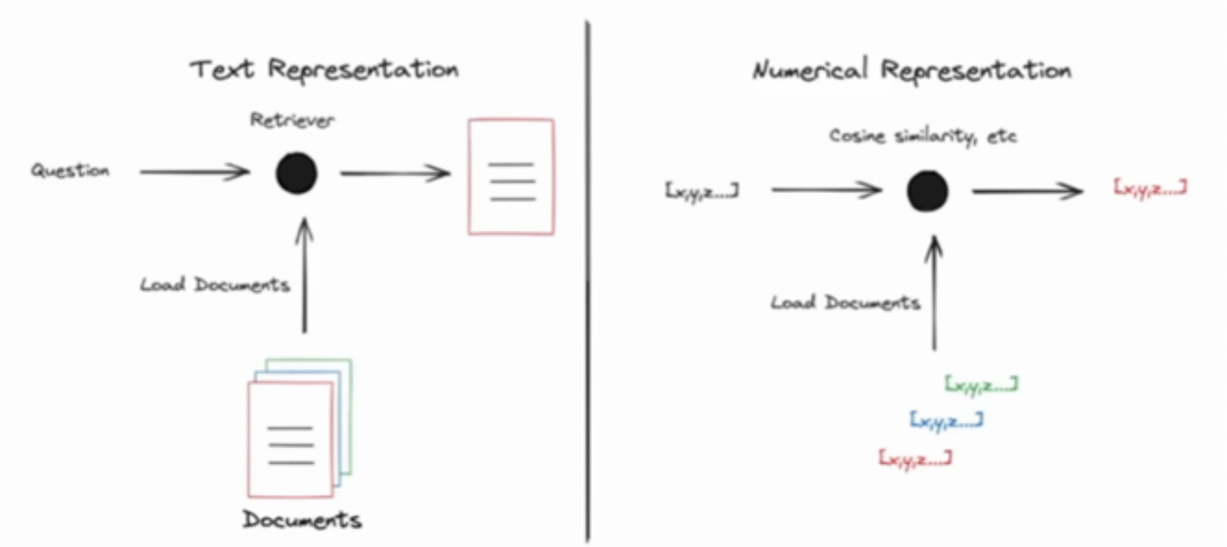
2
리트리벌 (Retrieval)

쿼리 단계: 사용자 질문이 들어오면 이 또한 벡터화하여 DB에서 가장 유사한 '가까운 이웃' (Closest Neighbors)' 문서를 찾아 LLM에 전달

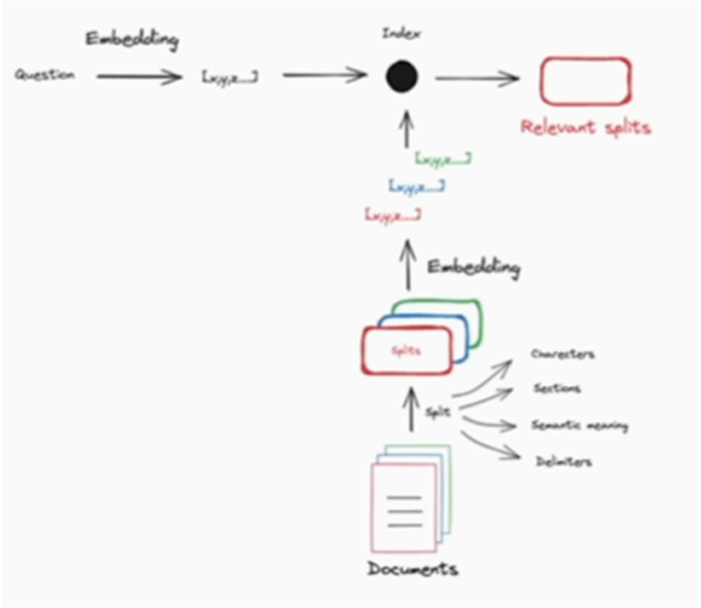
3
생성 (Generation)

인덱싱, 리트리벌 단계 이후 LLM에 내용 전달 이후 답변 생성하여 유저가 검사 및 확인

2. Embedding(Text → 숫자 전환)



3. 벡터 DB 저장

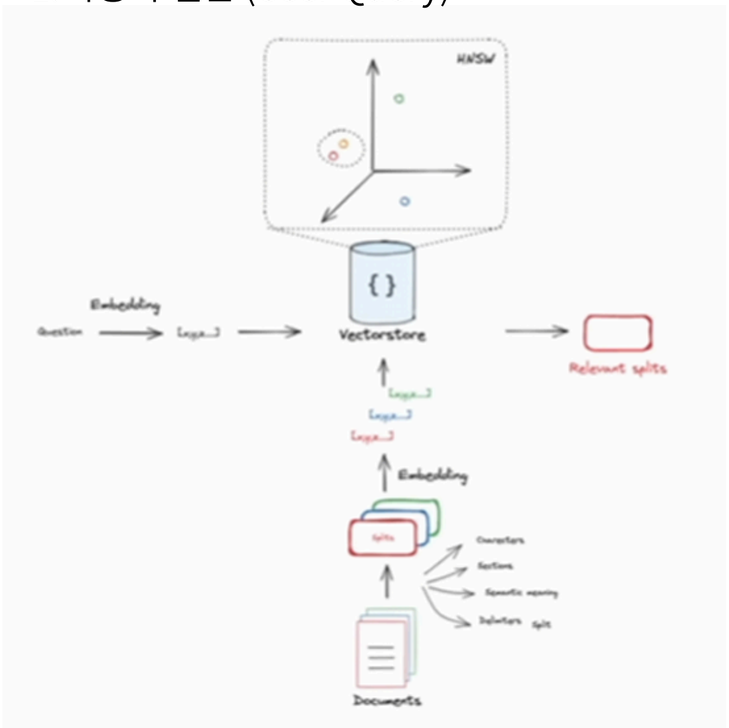


03. 아키텍쳐 및 작동원리

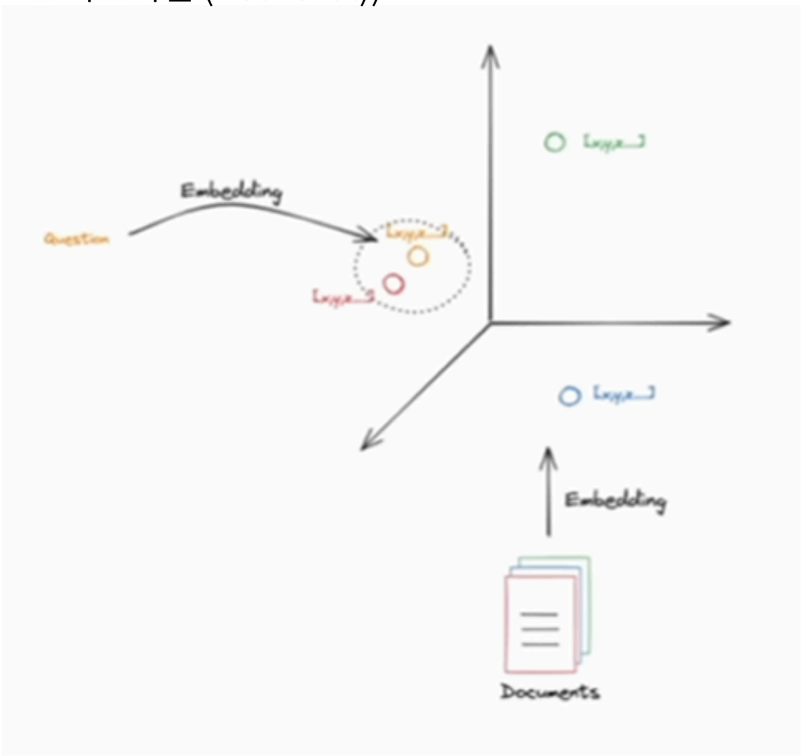
C. 리트리벌 (Retrieval)

1	2	3
인덱싱 (Indexing)	**리트리벌 (Retrieval)**	**생성 (Generation)**
문서를 로드하고, 의미 있는 단위로 분할(split) 한 뒤, 벡터(Vector) 형태로 변환하여 벡터 데이터 베이스에 저장 단계	쿼리 단계: 사용자 질문이 들어오면 이 또한 벡터화하여 DB에서 가장 유사한 '가까운 이웃' (Closest Neighbors)' 문서를 찾아 LLM에 전달	인덱싱, 리트리벌 단계 이후 LLM에 내용 전달 이후 답변 생성하여 유저가 검사 및 확인

1. 사용자 질문 (User Query)



2. 리트리벌 (Retrieval))



03. 아키텍쳐 및 작동원리

D. 생성 (Generation)

1

인덱싱 (Indexing)

문서를 로드하고, 의미 있는 단위로 분할(split) 한 뒤, 벡터(Vector) 형태로 변환하여 벡터 데이터 베이스에 저장 단계

2

리트리벌 (Retrieval)

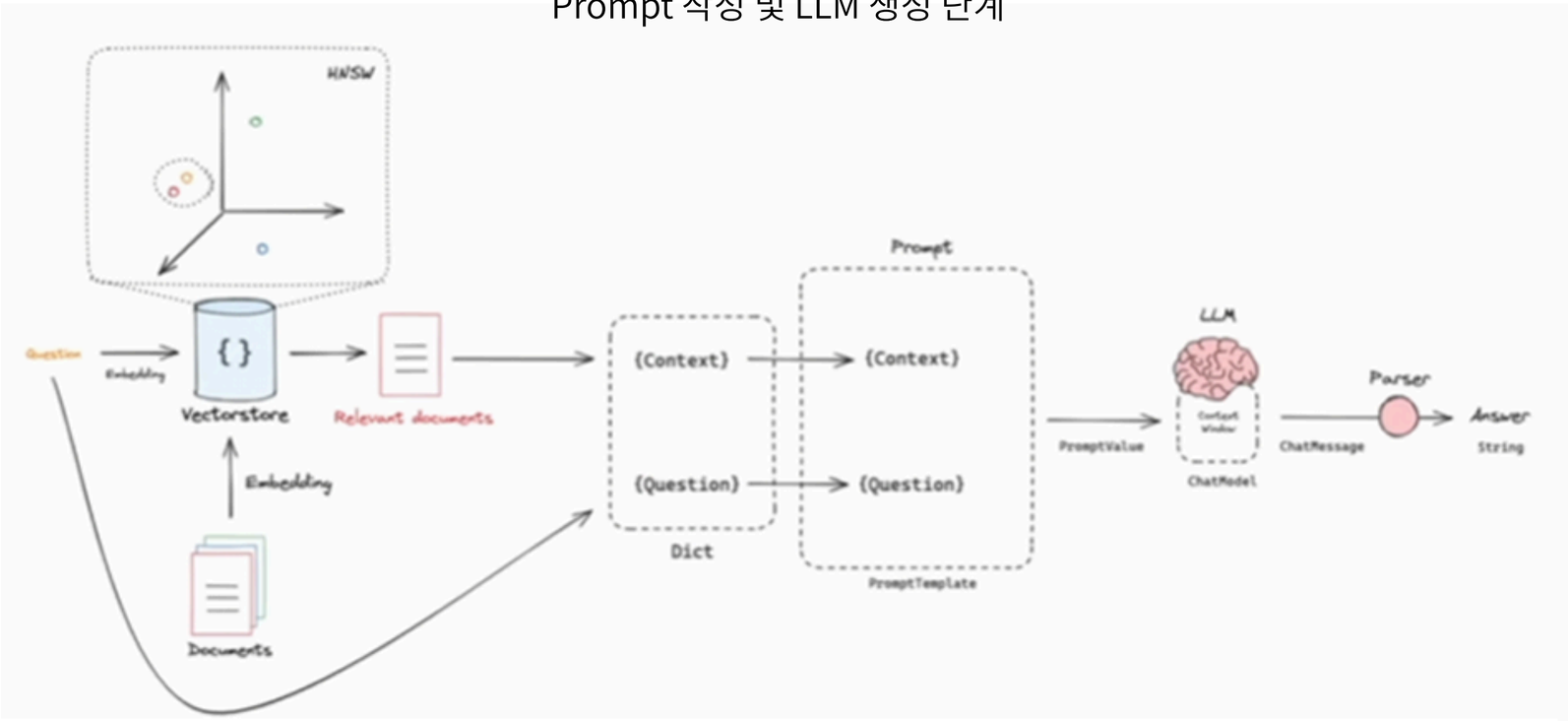
쿼리 단계: 사용자 질문이 들어오면 이 또한 벡터화하여 DB에서 가장 유사한 '가까운 이웃'(Closest Neighbors)' 문서를 찾아 LLM에 전달

3

생성 (Generation)

인덱싱, 리트리벌 단계 이후 LLM에 내용 전달 이후 답변 생성하여 유저가 검사 및 확인

Prompt 작성 및 LLM 생성 단계



04. 코드 구현

https://github.com/KimHoon98/RAG-Implementation

0.) README를 통해 페이지에 대한 설명, 프로젝트 순서를 확인할 수 있습니다.
1.) 각 파일마다 README 파일이 첨부 되어있으니 참조하시면 됩니다.
2.) 해당 파워포인트에 대한 설명은 Note explanation Section 참조하세요.

05. 개선 방향

RAG 고도화: 분석 리포트 누락 문제 해결 방안

마지막 질문 예시: 삼성전자 2025년 Q4 분기실적발표에 대해 상세하게 모든 디테일을 놓치지 않고 financial sellside analyst report 형식으로 3장으로 요약해줘

- 하지만 여기서 3장보다 훨씬 적은 양의 페이지를 output했다는 것을 확인할 수 있음 → LLM의 기존 학습 형식에 의한 문제로 파악이 가능

현재 방식의 한계점 분석:

1. LLM의 내재석 특성
    a. LLM은 확률 모델로서 정확성보다 문장의 매끄러움을 우선시하며, 긴 텍스트 생성 시 뒤로 갈수록 일관성이 떨어지는 경향이 있습니다.
2. 비효율적 대안
    a. 수동 복사/붙여넣기는 RAG의 자동화 장점을 무색하게 하며, 전체 재학습(Retraining)은 엄청난 컴퓨팅 비용을 발생시킵니다.

해결 방안:

- LLM에게 Plan-and-execute 형식으로 자동적으로 계획에 대한 RAG를 실행