Software Tool Time

VectorDB(Qdrant)를 활용한 RAG 기반 간단한 문서 Q&A 시스템 구축

- 강사
  - 이주영

- 목표
  - VectorDB 기반 검색 시스템을 직접 구성
  - RAG의 전체 파이프라인 구현
  - 실제 서비스 형태(FastAPI API + Docker + Qdrant)로 동작하는 MVP 구축

- 사전 준비
  - Python 3.10+, Docker Desktop, Git, IDE(VS code)

목차

배경

- LLM의 한계
  - LLM은 아직 학습하지 못한 최신 정보, 특정 분야, 조직 내부 데이터 등에 대해서는 정확한 답변을 주지 못함.

    → 할루시네이션 문제

    → 설명력이 부족한 정보의 생성

배경

- 해결 방안
  - RAG
  - 검색 증강 생성(Retrieval-Augmented Generation)
  - 대형 언어 모델 (LLM)이 새로운 정보를 검색하고 통합할 수 있도록 하는 기술.
  - RAG를 사용하면 LLM은 지정된 문서 집합을 참조할 때까지 사용자 쿼리에 응답하지 않음.
  - LLM은 훈련 데이터에서 사용할 수 없는 도메인 특정 및 업데이트된 정보 사용 가능
  - Ex) LLM 기반 챗봇이 내부 회사 데이터에 접근
  - Ex) 권위 있는 출처를 기반으로 응답을 생성

위키백과 - 검색증강생성성
https://ko.wikipedia.org/wiki/%EA%B2%80%EC%83%89%EC%A6%9D%EA%B0%95%EC%83%9D%EC%84%B1

배경

- RAG 주요 단계
  - 색인화
    - 참조 데이터(텍스트/반정형/정형)를 임베딩 벡터로 변환
    - 생성된 벡터를 벡터 데이터베이스(Vector DB) 에 저장
  - 검색
    - 사용자 쿼리를 임베딩하여 가장 관련성 높은 문서를 선택
    - 색인 방식에 따라 다양한 유사도 기반 검색 방법이 활용
  - 증강
    - 검색된 문서를 LLM 프롬프트에 통합하여 정보를 보강
  - 생성
    - LLM이 사용자 쿼리 + 관련 문서를 바탕으로 최종 답변을 생성

배경

- Qdrant
  - 고차원 임베딩을 빠르게 검색하기 위한 벡터 데이터베이스
  - 의미 기반 검색(Semantic Search)·추천 시스템 등에서 활용
  - 벡터 + payload(추가 정보)를 함께 저장하여 풍부한 검색 결과 제공
  - 각 데이터(문서, 이미지 등)를 벡터(embedding) 로 표현
  - 벡터 + ID + payload → Point
  - Point들을 모아 저장하는 단위 → Collection

프로젝트 개요

- 목표
  - 다양한 오픈소스 도구로 문서 기반 질의응답 시스템 구축
  - 사용자 관심 분야에 정확한 답을 제공하는 것
- 데이터
  - 2025 아주대학교 요람_소프트웨어학과
  - 학사과정_학사운영규칙
- 사용기술
  - Backend: FastAPI, Sentence-Transformers, Qdrant(VectorDB), OpenAI GPT
  - Frontend: Streamlit
  - 배포/실행: Docker Compose

시스템 구조

- 전체 파이프라인
  - 문서 정제 → Chunking → Embedding → Qdrant upsert → 검색 → 프롬프트 생성 → LLM 답변

1. 문서 정제 및 Chunking
2. Embedding 생성
3. VectorDB(Qdrant)에 저장
4. Retrieval & Generation

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

실습

- 단계
  1. 프로젝트 실행 환경 & Qdrant 인프라 세팅
  2. 문서 Ingestion & Chunking 파이프라인 구현
  3. Embedding 생성 기능 구현
  4. Qdrant Vector 저장 & 검색 기능 구현
  5. RAG Retrieval 파이프라인 구현
  6. LLM 기반 답변 생성 기능 추가
  7. API 서버 완성 및 테스트

트러블슈팅

- 임베딩 모델 변경
  - all-MiniLM-L6-v2 → multilingual-e5-small로 교체
  - 한국어 검색 품질 개선

- Chunk 전략 변경
  - 고정 길이 300자 → 문단 기준, 최대 1000자 로직으로 바꿔 문맥 보존

비교 및 대안

- Qdrant
  - 벡터 검색 품질 높음
  - 일반적인 문서 검색용 RAG
- Weaviate
  - GraphQL API라 개발 쉬움
  - multimodal vector 연동 쉬움
  - 이미지 + 텍스트 검색
- Milvus
  - 확장성이 뛰어난 오픈소스 고성능 벡터 데이터베이스
  - 초대규모 RAG 시스템에 사용

Software Tool Time (소프트웨어 툴 타임)

Visit “Software Tool Time” channel in YouTube :
https://www.youtube.com/c/SoftwareToolTime

본 영상은 2022년도 과학기술정보통신부 및 정보통신기획평가원에서 지원하는『SW중심대학사업』의 결과물입니다.