

# LangChain 및 RAG 활용 의료 LLM 개발 MediChain

2025.06.04

5조

윈도우즈



# 목차 contents

01

소개

02

수집된 데이터 및 데이터 전처리 과정 03

개발된 프로그램

04

테스트 결과 및 결과 분석 05

향후 개발 방향





# 권성호

믿음과 사랑을 가지고 프로젝트에 임했습니다. 모두들 성공하십쇼.

# 이준석

많이 부족했는데, 다들 많이 신경써주셔서 성공적으로 끝낼 수 있었습니다. 감사합니다.

# 손현성

이번 프로젝트로 협업하는 과정에서의 어려움을 경험해본 것 같아 좋은 경험이 되었고, 만족스러운 결과물이 나온것 같아 팀원들에게 감사합니다. 고생하셨습니다. 남의헌

'팀' 프로젝트를 잘 마무리 할 수 있게 협력한 팀원들에게 감사한 마음을 전합니다. 다들 고생 많으셨습니다

# 이준배

여러가지 데이터로 시작하여 모두 힘들 합쳐 하나의 프로그램을 만들었다는게 뿌듯합니다.

디드 그새 마이셔스니다



Microsoft

Microsoft

Microsoft

0000

수집된 데이터 및 전처리 과정

원본 데이터

필수의료 의학지식 데이터 전문 의학지식 데이터

초거대 AI 헬스케어 질의응답 데이터

데이터 형식

- JSON 파일 다수 (문서 단위로 저장) - content, answer 등 다양한 필드 포함

설명문, 질문/답변, 요약 등 자연어 기반 텍스트 -> 문서 별로 주제나 유형이 상이하고, 파일 수가 많으며 구조가 제각각이라 전처리 필요

데이터 전처리

텍스트 정제 - 불용어 제거 (한국어 불용어 리스트

기반)

- 특수문자, 공백 정리

- 여러 문장 병합 및 단순화

콘텐츠 추출 및 연결 - 주요 필드에서 핵심 텍스트 추출 후 하나로 연결

카테고리 부여

- 각 문서에 주제 기반 카테고리 태그 추가

벡터화된 데이터

텍스트 임베딩 - 문장을 고차원 숫자 벡터로 변환

111111

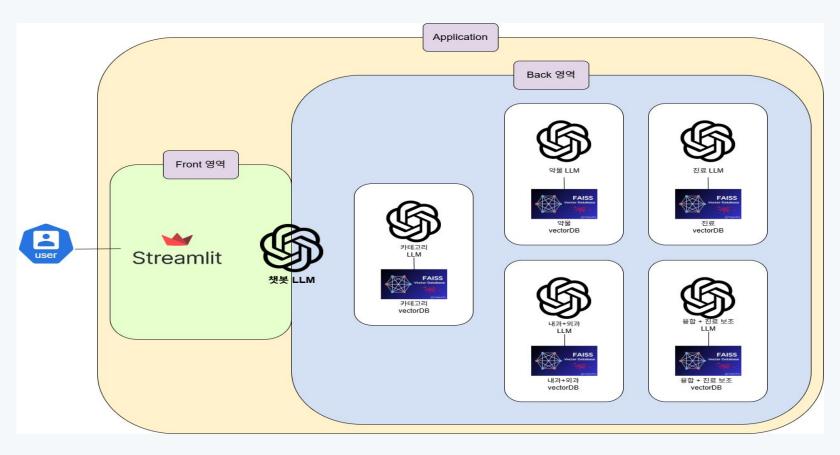
벡터 DB 구축

- 벡터 인덱싱 및 저장

- 유사도 기반 검색을 위한 구조로 변환



#### 현 시스템 아키텍처





# 의료 챗봇 현 시스템 아키텍처 요약

#### 1. Front 영역

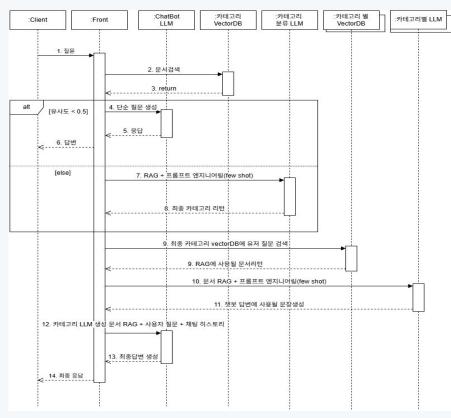
- Streamlit 기반의 프론트엔드에서 사용자가 직접 질문 입력 및 결과 확인
- 챗봇 LLM이 프론트와 연결되어, 전체 대화 흐름의 중심 역할 수행

# 2. Back 영역 (Application)

- 카테고리 분류 LLM과 전용 카테고리 VectorDB:
  - 질문의 주제(카테고리)를 자동 분류
  - 카테고리 분류 DB(FAISS) 기반으로 빠르게 분류
- 카테고리별 LLM + VectorDB:
  - 각 카테고리(약물, 진료, 내과+외과, 융합+진료보조)별로 별도 LLM과 VectorDB(FAISS) 존재
  - 해당 카테고리에 맞는 LLM이 VectorDB에서 관련 문서/context 추출 후, RAG 방식으로 전문적인 응답 생성

#### 

#### 시퀀스 다이어그램



#### 핵심 요약

- **단순 질문:** 벡터DB 유사도가 낮으면 챗봇 LLM이 바로 답변
- 복잡 질문: 벡터DB에서 문맥 확보 → 카테고리 분류 LLM이 적합 카테고리 결정 → 카테고리별 벡터DB/LLM에서 문맥 기반 응답 → 챗봇 LLM이 최종 답변
- 세션/히스토리 활용: 최종 응답 생성 시 이전 대화, 사용 문서, context 등 활용

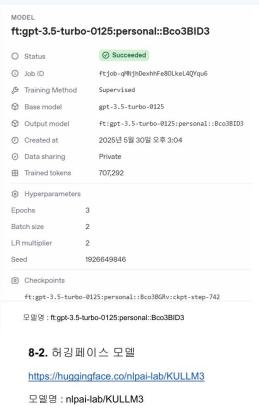
➡ 단순 질문-고도화 질문 모두 최적 경로로 분기, 카테고리별 RAG/LLM과의 연동이 핵심임



### 테스트 목적 및 대상 모델 (파인튜닝 모델 포함)

- 다양한 LLM 모델의 성능 비교
- Fine-tuning (화면 우측 모델)의 효과 확인
- KULLM3 모델의 토큰 수 및 샘플링 설정에 따른 성능 차이 분석
- 환각(Hallucination) 현상 방지 여부 확인

모델명	특징
GPT-3.5 Turbo	OpenAi 기본 모델
GPT-3.5 Turbo(FT)	사용자 데이터로 파인튜닝한 모델
GPT-4o-mini	최신 경량형 OpenAi 모델
KULLM3 (max 1024, sampling)	한국어 특화 모델, 샘플링 사용
KULLM3 (max 256, no sampling)	동일 모델, 샘플링 비활성화, 짧은 응답 유도





### 테스트 방법

- GPT 계열: '당뇨병에 대해 알려줘' 라는 질문에 대한 답변 받기
- KULLM 계열: '고려대학교에 대해서 알고 있니?' 라는 공식 문식 예시 질문에 대한 답변 받기

#### 평가 기준

- 적절성: 질문에 대한 응답이 의미 있고 구체적인지 여부
- 환각: 틀린 정보 또는 허구를 생성하는지의 여부
- 응답 길이: 너무 짧거나 불필요하게 긴 지에 대한 응답 여부
- 일관성: 유사 질문에 대해 유사한 응답을 하는지에 대한 여부



테스트 결과 (GPT-3.5 Turbo)



# 건강 질의응답 RAG 챗봇

? 궁금한 건강 관련 질문을 입력하세요:

당뇨병에 대해 알려줘

질문하기



당뇨병은 혈당 조절이 원활하지 않아 혈당 수치가 지속적으로 높은 상태를 말합니다. 이는 인슐린 분비 부 족이나 인슐린 작용 부족으로 인해 발생할 수 있습니다. 당뇨병은 혈당 조절을 위해 생활습관 개선, 약물 치료, 인슐린 주사 등 다양한 치료 방법이 있습니다. 만약 당뇨병 의심 증상이 있다면 즉시 의료진과 상담 하여 적절한 치료를 받는 것이 중요합니다.

적절성: 당뇨병에 대한 설명, 발생원인, 치료방법과 같은 필수적인 요소들이 모두 담겨있는 상태.

환각: 틀린 정보 또는 허구를 생성하지 않았음.

응답 길이: 응답 길이는 보통 길이이며, 실행 시간 또한 준수.

일관성: 유사 질문에 대해 유사한 응답.



테스트 결과 (GPT-3.5 Turbo FT)



# 건강 질의응답 RAG 챗봇

? 궁금한 건강 관련 질문을 입력하세요:

당뇨병에 대해 알려줘

질문하기



당뇨병은 인슐린 분비 부족, 인슐린 작용 부족, 또는 두 현상이 동시에 나타나는 대사 이상으로 인해 발생 하는 만성 질환입니다.

적절성: 당뇨병에 대한 설명, 발생 원인만 대답.

환각: 틀린 정보 또는 허구를 생성하지 않았음.

응답 길이: 응답 길이는 짧으며, 실행 시간은 준수.

일관성: 유사 질문에 대해 유사한 응답.



# 테스트 결과 (GPT-4o)



# 건강 질의응답 RAG 챗봇

? 궁금한 건강 관련 질문을 입력하세요:

당뇨병에 대해 알려줘

질문하기

# ★ 답변:

당뇨병은 혈당(혈액 내 포도당) 수치가 비정상적으로 높은 상태를 말합니다. 주로 인슐린의 분비 부족이나 인슐린 저항성으로 인해 발생합니다. 당뇨병은 크게 제1형과 제2형으로 나눌 수 있습니다.

- 1. 제1형 당뇨병: 주로 어린이나 청소년에서 발생하며, 췌장에서 인슐린을 생산하는 세포가 파괴되어 인 슐린이 부족해지는 상태입니다. 이 경우 인슐린 주사가 필요합니다.
- 2. 제2형 당뇨병: 성인에서 주로 발생하며, 인슐린 저항성이 주요 원인입니다. 초기에는 인슐린이 정상 또는 증가된 상태일 수 있지만, 시간이 지남에 따라 인슐린 분비가 감소하게 됩니다. 생활습관 개선, 약물 치료, 인슐린 요법 등이 필요할 수 있습니다.

당뇨병의 주요 증상으로는 잦은 소변, 갈증, 체중 감소, 피로감 등이 있으며, 장기적으로는 심혈관 질환, 신 장 질환, 신경 손상 등의 합병증을 유발할 수 있습니다. 따라서 조기 진단과 적절한 관리가 중요합니다. 식 이요법, 운동, 혈당 모니터링 등이 치료의 핵심입니다.

적절성: 당뇨병에 대한 설명, 발생 원인, 당뇨병 유형, 증상 등 여러 요소를 대답.

환각: 틀린 정보 또는 허구를 생성하지 않았음.

응답 길이: 응답 길이는 길며 자세함. 실행 시간도 준수함.

일관성: 유사 질문에 대해 유사한 응답.



테스트 결과 (KULLM3 (max 1024, sampling))

```
temperature=0.8
)

O 60m 55s

Python

C:\Users\Playdata\Desktop\3\\\_ 프로젝트\project3\Lib\site-packages\tqdm\auto.py:21: TqdmMarning: IProgress not found. Please update jupyter and ipywidget from .autonotebook import tqdm as notebook_tqdm
Loading checkpoint shards: 100%| *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% *** 100% **
```

-> 응답이 60분이 넘어가도 나오지 않음.

적절한 응답 여부와 상관없이 응답 시간이 너무 길어 의미가 없다고 판단하고 중단.



테스트 결과 (KULLM3 (max 256, no sampling))

```
● 30m 21.6s

Loading checkpoint shards: 100% 5/5 [00:00<00:00, 245.73it/s]
Some parameters are on the meta device because they were offloaded to the disk and cpu. 네, 고려대학교에
```

-> sampling 사용 시보단 훨씬 개선됐지만, 응답이 30분이 지나도 나오지 않아 실패. KULLM 사용은 힘들 거 같다고 판단.



결과 분석

GPT 계열: GPT3.5(FT) - GPT3.5 - GPT4o 순서로 답변의 질이 좋아지며 자세해짐.

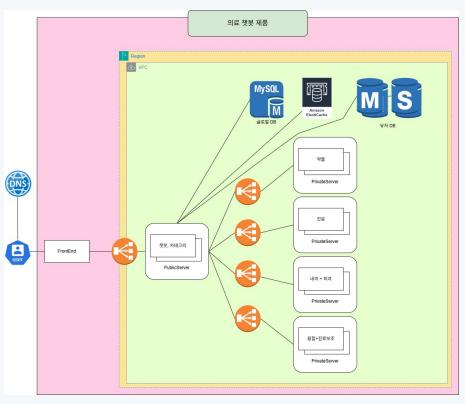
실행시간은 세 모델 모두 비슷.

-> 토큰 효율화 하되, GPT-40 쓰는것이 제일 좋을 것으로 예상.

KULLM 계열: max\_token을 줄이고 sampling을 하지 않았음에도 불구하고, 응답 시간이 너무 길어 실사용은 불가능하다고 판단.



#### 향후 시스템 아키텍처 및 참고사항



# 의료 챗봇 마이크로서비스 아키텍처 요약

#### 1. 사용자 접근

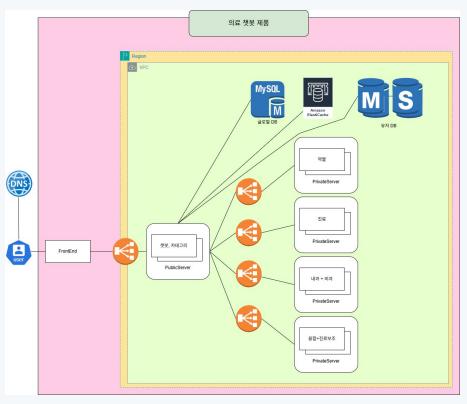
● 사용자는 인터넷을 통해 **DNS** → \*\*FrontEnd(웹/앱)\*\*로 접속

#### 2. PublicServer

- PublicServer에 첫봇 및 카테고리 분류기가 존재
- 모든 요청이 PublicServer(외부 오픈 서버)
   로 집결

#### 00000

#### 향후 시스템 아키텍처 및 참고사항



### 3. 로드밸런서 (ELB)

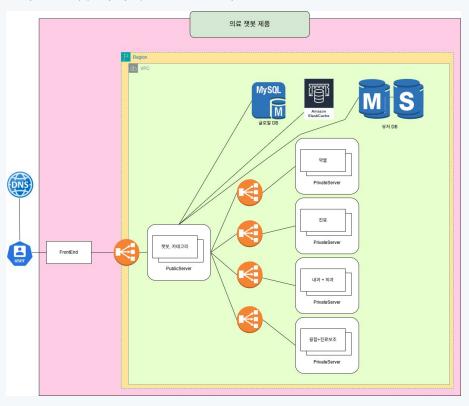
- PublicServer와 각 PrivateServer(전용 서버)
   사이에 로드밸런서가 존재
- 요청 유형(카테고리)에 따라 알맞은 PrivateServer로 분배

# 4. PrivateServer (카테고리별 마이크로서비스)

- 약물, 진료, 내과+외과, 융합+진료보조 등 각
   분야별 PrivateServer가 독립적으로 존재
- 각 PrivateServer는 개별 LLM, 벡터DB 등 독립적인 모델·DB·코드를 가질 수 있음



#### 향후 시스템 아키텍처 및 참고사항



### 5. DB 구성

- **글로벌 DB(MySQL, Redis 등)**: 전체 서비스/유저 관리
- 유저 DB: 서비스별, 분야별 필요에 따라 독립적으로 운영

# 05 향후 개발 방향



# 구조적 특징 및 장점

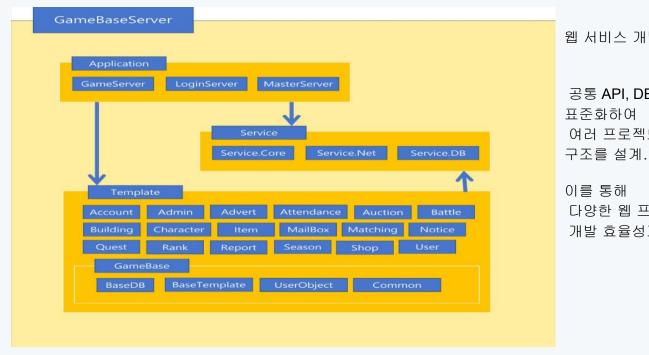
- 완전 마이크로서비스 구조
  - → 각 의료 분야를 독립 서버/컨테이너로 분리, 확장 및 교체가 매우 용이
- 로드밸런싱 기반 안정성/확장성
  - → 요청 폭주/특정 분야 트래픽 증가시 자동 확장/안정 운영 가능
- 카테고리별 RAG 및 LLM 인스턴스 분리
  - → 각 분야별 전문화/최적화 가능, 성능·비용·품질 최적화
- 서비스 간 결합도 최소화
  - → 유지보수, 배포, 장애 격리 용이
- 추후 엔터프라이즈 확장 용이
  - → 새로운 의료 분야, 데이터 소스, 외부 서비스 추가도 유연하게 대응

#### 결론

- 모든 분야/서비스를 분리하여, **확장성, 안정성, 유지보수성**을 극대화한 의료 챗봇 시스템!
- 추후 실서비스/상용화/엔터프라이즈 적용까지 고려한 설계임.



#### 향후 시스템 아키텍처 및 참고사항



웹 서비스 개발 시,

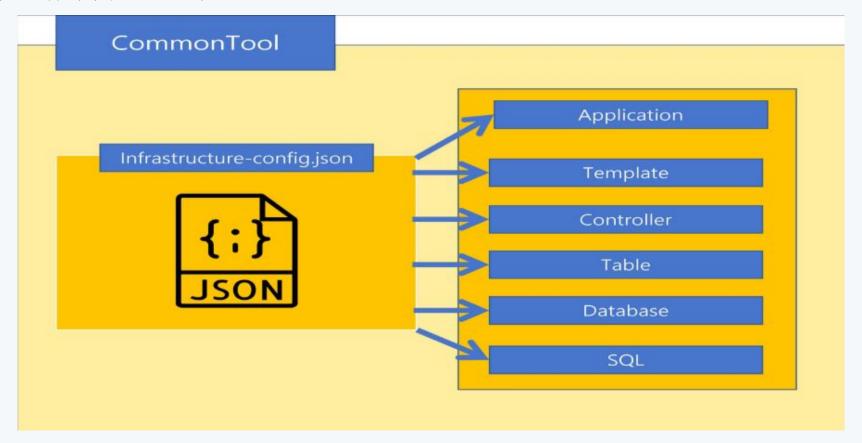
공통 API, DB 테이블, 쿼리, 데이터 구조, 콘텐츠를 표준화하여 여러 프로젝트에서 쉽게 재사용할 수 있도록 서버

#### 이를 통해

다양한 웹 프로젝트에 빠르게 적용할 수 있고, 개발 효율성과 유지보수성이 크게 향상



#### 향후 시스템 아키텍처 및 참고사항



# 05 향후 개발 방향



#### 향후 시스템 아키텍처 및 참고사항

#### 1. infrastructure-config. json

• JSON 문서를 정해진 규칙에 따라 파싱하여, 각 항목별 인프라 기능을 자동으로 분리·생성·실행

#### 2. Application 생성

• 명시된 라이브러리와 템플릿을 이용해 웹 서버 프로젝트와 기본 구조 자동 생성

#### 3. Template 생성

- 서비스/도메인별로 구분하여 템플릿 자동 생성
- 템플릿에는 공통 API, SQL, 로직 예시, 개발 가이드라인 포함

#### 4. Controller 생성

- RESTAPI 엔드포인트별 컨트롤러 코드 자동 생성
- 각 템플릿마다 별도 컨트롤러가 생성되어 서비스 기능별로 분리

#### 5. Table 생성

• 데이터 주도적 개발을 위한 데이터 테이블 자동 생성

#### 6. Database 코드/SQL 생성 및 실행

- 명세서를 기준으로 DB 테이블, 프로시저, 쿼리문 자동 생성
- 생성된 **SQL**을 실제 데이터베이스에 적용



