<프로젝트　보고서>

민방위 대피시설 + 재난 행동요령 데이터 기반 RAG & 챗봇 시스템 프로젝트

SK Networks Family 20기 — 3RD 5TEAM [문창교 김황현 권규리 김효빈 이승규]

2025.12.11

1. 프로젝트 개요

본 프로젝트는 민방위 대피시설 CSV 데이터와 재난 행동요령 JSON 데이터를 결합하여 RAG(Retrieval-Augmented Generation) 기반 재난 대응 챗봇을 구축하는 것을 목표로 수행되었다. 2025년 12월 약 2주에 걸쳐 진행된 이번 프로젝트는 단순한 LLM 대화 기능을 넘어, 신뢰 가능한 외부 문서를 기반으로 정확한 정보를 제공하는 구조적 검색–응답 시스템을 만드는 데 중점을 두었다.

특히, 대규모 언어모델의 환각을 억제하기 위해 모든 답변을 VectorDB(ChromaDB)에 저장된 실제 문서에서 Retrieval한 내용만을 기반으로 구성하도록 설계했다. 프로젝트 팀은 문창교, 권규리, 김황현, 김효빈, 이승규 총 5명으로 구성되었으며, 데이터 로딩·전처리, Document 변환, 임베딩 및 VectorDB 구축, LangGraph Agent 설계, API 서버 개발(FastAPI), 그리고 웹 UI 구축(HTML·JavaScript·TailwindCSS)까지 전체 파이프라인을 역할별로 분담해 협업하였다.

2. 프로젝트 내용

2.1 개발 목표

본 프로젝트는 **정확성**, **신뢰성**, 그리고 **실제 재난 대응 활용 가능성**이라는 세 가지 특성을 유지할 수 있는 챗봇을 만들고자 했다.

첫째, 정확성을 위해 모델이 임의로 생성한 정보가 아닌, ChromaDB에 저장된 CSV·JSON 기반 Document만을 검색하여 응답하도록 RAG 구조를 설계하였다.

둘째, 위치와 재난상황에 따른 대피소 및 재난 행동요령 정보의 제공을 목표로 했다. 사용자가 특정 위치나 상황을 입력하면 시스템은 해당 위치 주변의 대피소 정보를 빠르게 추출하고, 재난 유형에 맞는 행동요령을 함께 안내한다.

셋째, 복합 질문 처리 기능 고도화를 추구하였다.실제 재난 상황을 고려하여 단일 질문보다 "위치 + 재난 상황"이 결합된 형태의 질문이 자연스럽게 발생하므로, LangGraph Agent가 의도를 분석해 필요한 도구를 조합적으로 불러올 수 있도록 구성했다.

넷째, 사용자 경험(UX) 향상을 목표로 기능을 설계하였다. 거리 기반 정렬, 지도 기반

시각화(Naver Maps), 대화 컨텍스트 유지 기능을 통해 위급 상황에서도 빠르고 직관적으로 필요한 정보를 확인할 수 있도록 했다.

2.2 주요 구현 기능

본 시스템은 LangGraph Agent 중심의 파이프라인으로 구성되어 있으며, 다음과 같은 기능들을 통합적으로 제공한다.

첫째, LangGraph Agent 기반 의도 분류와 질문 재정의 기능을 구현하였다. 사용자의 입력 문장은 8개의 의도 카테고리로 자동 분류되며, BM25 검색 성능을 높이기 위해 핵심 키워드를 추출하는 형태로 문장이 재작성된다. 이를 통해 Sparse Retrieval과 Dense Retrieval 모두에서 검색 품질을 안정적으로 확보하였다.

둘째, Hybrid Retrieval(하이브리드 검색) 구조를 구성하였다. OpenAI 임베딩(text-embedding-3-small)을 활용한 의미 기반 Dense Retrieval과 키워드 매칭 기반의 BM25 Sparse Retrieval을 결합하여 각각의 장점을 균형 있게 활용할 수 있게 했다. 가중치를 통해 두 검색 결과를 통합함으로써 다양한 질문 유형에서 높은 정확도를 유지한다.

셋째, 위치 기반 대피소 추천 기능을 제공한다. Kakao Local API를 사용해 지명을 좌표로 변환하고, Haversine 공식을 적용해 사용자 위치와 대피소 간 거리를 계산하며, 가장 가까운 대피소를 추천한다. 이는 실제 재난 상황에서 핵심적인 기능으로, 빠른 의사결정을 돕는 데 주요한 역할을 한다.

넷째, 복합 질문 처리 기능을 포함하였다. 예를 들어 "강남역 근처인데 땅이 흔들려"와 같은 입력이 들어오면 Agent는 이를 "위치 기반 대피소 검색"과 "지진 행동요령 제공"이라는 두 기능을 조합한 복합 처리로 판단한다. 이 결과, 단일 질문에 대해 두 종류의 정보를 모두 담은 통합 응답을 제공할 수 있게 된다.

다섯째, Naver Maps 기반 시각화 기능을 추가하였다. 웹 UI에 지도 기반 결과를 표시하여 사용자가 자신의 위치와 대피소 위치를 직관적으로 확인할 수 있게 했으며, 대피소의 상세 정보는 지도 마커 및 인포윈도우 형태로 제공된다.

여섯째, 대화 메모리 기반 멀티턴 대화 기능을 구현하였다. LangGraph의 MemorySaver 기능을 활용해 사용자의 이전 질의와 맥락을 유지함으로써 연속적인 대화 흐름을 자연스럽게 처리할 수 있다.

3. 기술 스택

본 프로젝트는 데이터 처리에서부터 에이전트 로직, API 서버, 프론트엔드 시각화까지 전 과정이 유기적으로 연결되는 구조를 갖고 있다. 전반적으로 Python 기반의 백엔드와 웹 브라우저 기반 UI로 운영되며, RAG 시스템 구축에 필요한 임베딩 모델, 벡터 데이터베이

스, 검색 알고리즘 등이 체계적으로 통합되었다.

우선, 개발 언어는 Python 3.10을 핵심으로 사용했다. Python은 풍부한 데이터 처리 라이브러리와 LLM 프레임워크 생태계를 갖추고 있어 RAG 기반 시스템 구축에 적합했다. 데이터 로딩과 전처리에는 pandas를 활용하여 CSV·JSON 데이터를 일관된 형태로 다루었고, 이를 LangChain Document 형식으로 변환하는 과정에 여러 Python 유틸리티 라이브러리가 함께 활용되었다. AI·Agent 프레임워크는 LangChain과 LangGraph를 조합해 구성했다. LangChain은 임베딩 생성, Retriever 구성, Document 체계화 등 RAG 시스템의 기본 모듈을 제공하며, LangGraph는 에이전트의 상태 기반 흐름을 정의하기 위한 데 사용되었다. 특히 LangGraph는 의도 분류 → 질문 재작성 → 도구 선택 → 도구 실행으로 이어지는 단계를 독립적 노드로 구성할 수 있어 복잡한 대화 흐름을 안정적으로 통제할 수 있었다. LLM은 OpenAI GPT-4o-mini 모델을 사용하였다. 이 모델은 비교적 빠른 응답속도와 안정적인 판단 능력을 제공하며, Temperature를 0 또는 0.7로 조절해 각각 분류 작업과 일반 대화 작업에 적합하도록 구분하여 활용하였다. 문서 임베딩은 OpenAI의 text-embedding-3-small 모델을 사용하였으며, 1536차원 벡터를 생성해 문서의 의미적 특징을 효율적으로 반영했다. 벡터 데이터베이스는 ChromaDB를 채택하였다. ChromaDB는 영구 저장(persistent) 모드를 지원하여 약 17,500개의 Document를 안정적으로 저장할 수 있으며, Dense Vector 기반 검색에 최적화되어 있다. ChromaDB는 LangChain과의 통합도 용이해 Retriever 구성 단계에서 성능과 편의성을 모두 확보할 수 있었다. 검색 알고리즘은 Hybrid Retrieval 구조를 통해 구현했다. Dense Retrieval(OpenAI 임베딩)과 BM25 Search(Sparse Retrieval)를 함께 사용하고, 두 검색 결과는 Ensemble 하이브리드 알고리즘을 통해 가중 평균으로 통합했다. 이를 통해 의미 기반 검색 정확도와 키워드 기반 정밀 검색의 장점을 동시에 확보할 수 있었다.

백엔드 서버는 FastAPI + Uvicorn 조합으로 구축되었다. FastAPI는 비동기 처리 기반의 고성능 API 서버 프레임워크로, 에이전트 호출과 데이터 처리 요청을 빠르게 처리할 수 있었다. 또한 자체적으로 OpenAPI 문서(Swagger UI)를 제공하기 때문에 디버깅과 API 구조 점검이 용이했다. SSL 인증서 기반 HTTPS 통신도 함께 구성하여 외부 접속 환경에서도 안정성을 확보했다.

프론트엔드 기술은 HTML5, JavaScript, TailwindCSS로 구성되었다. 사용자는 웹 인터페이스를 통해 질문을 입력하고, 챗봇 응답과 지도 시각화를 확인한다. 지도 시각화는 Naver Maps API를 사용하며, 좌표 변환은 Kakao Local API를 통해 수행하였다. 브라우저의 geolocation 기능을 활용하면 사용자의 실시간 위치 기반 대피소 추천도 가능하다.

4. 수집된 데이터 및 데이터 전처리

본 프로젝트의 기반이 되는 데이터는 민방위 대피시설 데이터(CSV)와 재난 행동요령

데이터(JSON) 두 가지로 구성된다. 두 데이터는 출처와 구조가 상이하며, RAG 시스템이 안정적으로 작동하기 위해서는 이를 일관된 Document 구조로 통합하는 전처리 과정이 필수적이었다. 본 장에서는 데이터의 수집 과정과 이를 LangChain Document 형태로 변환하기 위한 전처리 절차를 설명한다.

우선, 민방위 대피시설 데이터는 공공데이터 포털에서 제공되는 CSV 파일을 활용하였다. 이 데이터에는 전국 약 17,000여 개에 이르는 대피소의 시설명, 주소, 위도·경도 좌표, 최대 수용인원 등 핵심 정보가 포함되어 있다. 데이터는 지역마다 관리 방식이 상이해 일부 항목이 누락되거나 값의 정규화가 필요했기 때문에, pandas를 활용해 불일치 항목을 정제하고 분석에 필요한 주요 컬럼을 추출하였다. 이후 각 행(row)은 LangChain Document로 변환되는데, 이때 page_content에는 대피소의 핵심 정보를 자연어 설명 형태로 구성하고, metadata에는 시설명, 주소, 위도, 경도, 수용인원 등 검색에 필요한 구조화된 필드를 포함시켰다.

두 번째 데이터인 재난 행동요령 JSON 데이터는 국민재난안전포털의 콘텐츠를 바탕으로 직접 구조화한 자료를 사용하였다. 이 데이터는 자연재난과 사회재난을 포함해 약 20종의 재난 유형을 다루고 있으며, 발생 전·발생 시·발생 후의 행동요령과 주의사항 등을 계층적 구조로 담고 있다. JSON 특성상 중첩 구조가 깊고 항목별 형태가 다르기 때문에, documents.py에서는 parse_node 함수를 활용해 JSON의 계층을 재귀적으로 탐색하며 개별 Document로 분해하였다. 각 Document는 "지진 > 발생 시 > 실내"와 같은 계층 경로를 metadata에 포함하며, page_content에는 실제 행동요령을 문장 형태로 구성해 저장하였다.

데이터 수집 이후 전처리 단계는 크게 네 단계로 구성되었다. 먼저 data_loaders.py에서 CSV와 JSON 파일을 불러오고 기본 구조를 표준화한다. 그 다음 documents.py에서 CSV·JSON 각각을 LangChain Document로 변환한 후 embedding_and_vectordb.py에서 OpenAI text-embedding-3-small 모델을 활용해 1536차원 임베딩 벡터를 생성한다. 마지막으로 생성된 Document와 벡터를 ChromaDB에 저장해 검색 가능한 형태로 구축한다. 이 전체 과정은 from_DataLoad_to_VectorDB.py에 통합되어 있어, 한 번의 실행으로 데이터 로딩부터 VectorDB 저장까지 자동으로 처리된다.

최종적으로 구성된 Document는 약 17,500건에 이르며, Dense Vector와 메타데이터 검색 모두에 활용할 수 있다. 다만 데이터의 특성상 정적 형태라는 제약이 있다. 즉, 공공데이터 포털이나 재난안전포털에서 정보가 갱신되더라도 시스템이 이를 자동으로 반영하지는 못하며, 최신성을 유지하기 위해서는 데이터를 재수집해 재임베딩 과정을 수행해야 한다. 또한 일부 지역의 대피소 정보가 누락되어 있을 가능성이 존재하며, JSON의 경우 수동 구성 과정에서 항목별 편차가 생길 수 있다는 점도 확인되었다.

5. 시스템 아키텍처

본 시스템의 전체 아키텍처는 데이터 계층, 에이전트 로직 계층, 백엔드 API 계층, 프론트엔드 UI 계층으로 구성하였다. 특히, 대화형 RAG 시스템이라는 특성을 고려하여 LLM 호출, 검색 로직, 외부 API 연동이 명확히 분리되어 있으며, 각 계층은 독립적인 역할을 수행하면서도 일관된 데이터 흐름을 유지한다.

우선 데이터 계층은 ChromaDB와 외부 위치 API로 구성된다. ChromaDB는 약 17,500개의 대피소 및 재난 행동요령 Document를 영구적으로 저장하고, Dense Vector 기반 검색을 처리한다. 임베딩은 OpenAI의 text-embedding-3-small 모델을 통해 벡터로 변환되어 저장되며, 이를 기반으로 Vector Search가 이루어진다. 또한 Kakao Local API는 지명 검색을 통해 좌표를 변환하는 역할을 수행하며, Naver Maps API는 지도 시각화에 사용된다. 이 상위 계층에는 LangGraph 기반의 Agent 로직이 위치한다. Agent는 시스템의 "두뇌" 역할을 수행하며, 사용자의 입력 문장을 해석하고 적절한 도구(Tool)를 선택하는 과정을 자동으로 수행한다. 전체 프로세스는 의도 분류 → 질문 재작성 → 검색 및 도구 실행 → 응답 생성의 흐름으로 구성되어 있다. 의도 분류 단계에서는 GPT-4o-mini 모델을 사용해 질문을 8개의 카테고리 중 하나로 분류하며, 이는 검색 전략 선택과 도구 선택에 직접적인 영향을 준다. 이후 질문 재작성 단계에서는 BM25 검색 성능을 높이기 위해 핵심 키워드를 강조하거나 불필요한 조사·형태소를 제거하는 방식으로 질의를 최적화한다. Agent의 핵심 요소 중 하나는 Multiplexed Tool System이다. Agent는 질문의 의도에 따라 총 7개의 도구 중 하나를 선택해 실행한다. 예를 들어 위치 기반 검색의 경우 Kakao Local API를 호출한 뒤, ChromaDB에서 대피소 목록을 가져와 Haversine 공식을 이용해 거리를 계산하고 가까운 순으로 정렬한다. 재난 행동요령 검색의 경우 Dense Retrieval과 BM25 Retrieval을 결합한 하이브리드 검색 구조를 통해 관련 Document를 추출한다. 복합 질문의 경우 위치 검색과 재난 행동요령 검색을 결합해 단일 응답으로 통합한다. 이러한 도구 기반 구조는 확장성과 유지보수성 측면에서 중요한 이점을 제공한다. 다음 계층은 FastAPI 기반 백엔드 서버이다. 이 서버는 프론트엔드에서 들어오는 요청을 받아 Agent에 전달하고, Agent가 생성한 응답을 클라이언트가 활용할 수 있는 형태로 변환한다. 서버 초기화 과정에서는 ChromaDB와 임베딩 모델을 로드하고, LangGraph Agent를 생성하여 전체 시스템이 사용할 고정된 에이전트 인스턴스를 준비한다. API 서버는 /api/location/extract, /api/shelters/nearest, /api/health, /api/status 등 기능별 엔드포인트를 제공하며, 요청과 응답은 JSON 형태로 주고받는다. 또한 CORS 설정을 적용해 브라우저 환경에서도 안정적으로 동작하도록 구성하였다. 최상위 계층은 프론트엔드 UI(shelter_1.0.html)로, 전체 시스템에서 사용자와 직접 상호작용하는 인터페이스이다. 이 UI는 JavaScript를 통해 백엔드 API에 요청을 전달하고 응답을 받아 챗봇 메시지 형태로 화면에 출력한다. 지도 시각화는 Naver Maps API를 활용하여 구현되며, 사용자 위치와 추천 대피소를 지도 위에 마커로 표시한다. GPS 기반 위치 추적 기능도 제공하여 실제

상황에서 빠르게 대피소 정보를 확인할 수 있다. 전체적으로 본 시스템의 아키텍처는 RAG 기반 대화형 검색 구조를 중심으로 설계되어 있으며, 데이터 검색과 UI 표시까지의 흐름이 명확하게 분리되어 있다.

6. 시스템 특징 및 제약사항

본 장에서는 본 프로젝의 강점과 구조적 한계점을 체계적으로 분석한다. 본 시스템은 RAG 기반 구조를 토대로 설계되었으며, 검색 정확도·구조적 확장성·사용자 경험 측면에서 여러 장점을 갖고 있음과 동시에, 데이터 특성과 외부 API 의존으로 인한 자연스러운 제약도 존재한다.

우선, 본 시스템의 가장 큰 특징은 환각(Hallucination) 최소화에 있다. 이는 대규모 언어모델이 자체적으로 생성한 정보를 기반으로 답변하지 않도록 하는 설계 전략으로, 검색 기반 구조를 통해 모든 응답이 ChromaDB에 저장된 실제 문서에서 Retrieval되도록 하고 있다. 따라서 사용자는 신뢰성 있는 정보를 안정적으로 제공받을 수 있으며, 민방위 대피시설 정보와 재난 행동요령 같은 사실 기반 데이터에서는 매우 중요한 가치로 작용한다. 또한 모델의 Temperature를 0으로 설정해 동일 질문에 대해 일관된 답변을 생성하도록 함으로써 정밀성과 재현성을 강화하였다.

두 번째 특징은 Dense Retrieval과 BM25 Retrieval을 결합한 하이브리드 검색 구조다. 의미 기반 검색(Dense)과 키워드 기반 검색(Sparse)의 장점을 절충하여, 재난명, 시설명, 주소, 행동요령 등 다양한 유형의 질문에서 균형 잡힌 검색 성능을 제공한다. 특히 재난 행동요령처럼 용어가 정형화된 문서에서는 BM25의 효과가 크고, 위치나 시설명을 포함한 문장은 Dense Retrieval의 의미 매칭이 강점으로 작용한다. 이를 가중 평균 기반 Ensemble 방식으로 통합함으로써 안정적인 검색 품질을 확보하였다.

세 번째 특징은 모듈형 아키텍처를 기반으로 한 높은 확장성과 유지보수성이다. 데이터 로딩, Document 변환, 임베딩 생성, Agent 로직, 백엔드 서버, 프론트엔드 UI가 각기 독립된 모듈로 구성되어 있어 기능 개선, 교체, 확장 작업이 용이하다. 예를 들어 대피소 데이터가 확장되거나 재난 종류가 늘어나더라도 특정 모듈만 업데이트하면 전체 시스템의 재구축 없이 반영할 수 있도록 구조화되어 있다. 이는 프로젝트 진행 중뿐 아니라 향후 운영 단계에서도 관리 효율성을 높이는 요소이다.

네 번째 특징은 위치 기반 시각화 기능이다. 특히 Naver Maps API를 사용해 지도 위에 대피소를 표시하고, Kakao Local API를 통해 지명·시설명을 좌표로 변환한다. 이를 통해 사용자는 텍스트 응답뿐 아니라 시각적으로도 정보를 확인할 수 있어 재난 상황에서의 즉각적 판단에 실질적인 도움을 준다. 또한 지도 상에서 사용자 위치와 대피소의 상대적 위치를 직관적으로 파악할 수 있어, 위치 기반 서비스로서 완성도를 강화하였다.

다섯 번째 특징은 대화 메모리 기반 멀티턴 처리 능력이다. LangGraph MemorySaver

기능을 통해 세션 내 대화 맥락을 유지할 수 있게 하여, 질문이 연속적으로 이어지거나 앞선 질문을 참고해야 하는 상황에서도 자연스러운 대화 흐름을 제공한다. 이는 단순한 검색 기능을 넘어 실질적 "대화형 에이전트"로 기능하는 데 중요한 역할을 한다.

반면, 이러한 강점에도 불구하고 몇 가지 제약사항이 존재한다. 가장 큰 제약은 정적 데이터 기반의 구조라는 점이다. 현재 사용되는 대피소 CSV와 재난 행동요령 JSON은 시스템에 별도로 저장되어 있으며, 수정되거나 새로운 정보가 추가되더라도 자동으로 반영되지 않는다. 최신 데이터를 유지하기 위해서는 전체 데이터 재수집—Document 변환—임베딩 재생성—DB 갱신 과정을 수동으로 수행해야 한다. 또한, 데이터 출처 자체의 한계도 존재한다. 공공데이터의 특성상 일부 지역의 대피소 정보가 누락되어 있을 가능성이 있고, 재난 행동요령 정보 역시 직접 구조화한 자료이기 때문에 항목 간 구성의 균형이 일정하지 않을 수 있다. 이러한 데이터 품질 문제는 검색 결과와 응답의 정밀도에 영향을 줄 수 있다. 그 외의 제약으로는 OpenAI API 의존성에 따른 비용 및 접근성 문제등이 존재한다.

종합적으로 본 시스템은 RAG 기반 재난 대응 서비스로서 기술적 완성도를 갖추고 있으며, 검색 신뢰성과 구조적 확장성 측면에서 높은 강점을 보인다. 동시에 정적 데이터·외부 API 의존 등 현실적 제약을 갖고 있어, 향후 개선을 통해 안정성과 최신성 확보가 요구된다.

7. 결론

본 프로젝트는 민방위 대피시설 데이터와 재난 행동요령 정보를 기반으로, 대규모 언어모델과 RAG 기술을 통합한 재난 대응 챗봇 시스템을 구축하는 것을 목표로 했다. 약 3주간의 개발 기간 동안 팀은 데이터 전처리, 벡터 기반 검색, LangGraph Agent 구성, API 서버 구축, UI 구현까지 전 과정을 단계적으로 완성해 나갔다. 그 결과, 의미 기반 Dense Retrieval과 키워드 기반 BM25 Retrieval을 결합한 하이브리드 검색 구조를 구현하고, 위치 기반 대피소 추천과 지도 시각화 기능을 포함한 통합형 재난 대응 서비스를 제공할 수 있게 되었다. 시스템은 ChromaDB를 활용해 약 17,500개의 문서를 구조화하고, LLM의 환각을 최소화하도록 설계하여 신뢰도 높은 응답을 제공한다. 또한 LangGraph Agent를 통해 질문 의도 파악, 도구 선택, 재난 행동요령 검색, 위치 기반 처리 등 복잡한 기능을 안정적으로 처리할 수 있는 구조적 기반을 마련하였다. Frontend와 Backend의 명확한 계층 분리, 모듈형 코드 구조는 향후 유지보수와 기능 확장을 용이하게 하였다. 다만, 정적 데이터 기반으로 인해 최신성 확보가 어렵고, OpenAI API 및 외부 지도·위치 API에 대한 의존성이 존재한다는 점은 향후 개선해야 할 과제로 남아 있다. 또한 다국어 지원 및 고가용성 인프라 구성은 실제 서비스화 과정에서 추가적으로 고려해야 할 사항이다. 그럼에도 불구하고 본 프로젝트는 단순한 챗봇 구현을 넘어, 데이터—검색—에이전트—UI까지 이어지는 전체 파이프라인을 완성했다는 점에서 교육적·기술적 성취를 이루

었으며, 재난 대응 분야에서 LLM 기반 정보 제공 시스템이 어떤 방식으로 구현될 수 있는지 실질적인 방향성을 제시하였다. 향후 데이터 자동 업데이트, 실시간 재난 알림 연동, 다국어 확장, 클라우드 배포 등을 통해 본 시스템은 보다 안정적이고 고도화된 형태로 발전할 수 있을 것이다.