

LLM을 활용한 검색 증강 생성(RAG) 모델 구현: 래피드마이너 기반으로

(Implementation of Retrieval Augmented Generation (RAG) Model Using LLM: A
RapidMiner-Based Approach)

양치복*, Kim Yang Sok**

(Chi Bok Yang*, Yang Sok Kim**)***

요약

LLMs을 기반으로 하는 생성형 AI기술은 기존의 한계를 극복하기 위하여 많이 되고 있다. 특히, RAG는 최신 정보나 학습되지 않은 특정 도메인 지식을 활용하여 LLMs의 환각 현상을 줄일 수 있어 생성형 AI 서비스에서 주목받고 있는 방법이다. 하지만, 대부분 연구에서는 프로그래밍 기반으로 구축하는 방법을 제안하고 있다. 본 연구에서는 래피드마이너를 이용하여 프로그래밍 지식이 없어도 RAG 아키텍처를 구축하는 방법을 제시한다. Qdrant 벡터 데이터베이스를 활용하여 임베딩을 저장한 후, 검색하는 과정을 설명하고, OpenAI API를 통하여 검색된 문서를 바탕으로 질의응답을 생성하는 구체적인 방법을 소개한다. 또한, 실제 질문을 하고, RAG 답변 결과를 제공하여 실제 적용 가능성을 확인하였다. GUI 기반의 래피드마이너를 통한 RAG 구현 방법을 제공함으로써, LLMs을 활용한 생성형 AI 서비스를 보다 직관적이고 신속하게 개발할 수 있음을 제시한다.

■ 중심어 : 검색증강생성 ; 대형 언어 모델 ; 생성형 인공지능 ; 래피드마이너

Abstract

Generative AI technology, driven by Large Language Models (LLMs), is being increasingly utilized to overcome existing limitations. Retrieval-Augmented Generation (RAG) has emerged as an effective approach to reduce hallucination in LLMs by leveraging up-to-date and domain-specific knowledge beyond training data. However, most studies propose programming-based implementations. This research introduces a GUI-based RAG framework using RapidMiner, to construct RAG systems without programming proficiency. The methodology includes storing and retrieving embeddings with the Qdrant vector database and generating question-and-answer pairs via the OpenAI API. Practical demonstrations confirm the system's effectiveness in real-world scenarios, offering a simpler and more efficient method for developing generative AI services with LLMs.

■ keywords : Retrieval Augmented Generation ; Large Language Model ; Generative Artificial Intelligence ; RapidMiner

I. 서 론

자연어 생성 및 추론 분야에서 ChatGPT와 같은 대규모 언어모델(Large Language Model, LLM)은 중요한 역할을 하고 있다. 대량의 텍스

트 데이터 학습을 통하여 주제 분류, 질의응답, 번역 등 자연어 처리에 탁월한 성능을 보이고, 경영, 금융, 소매, 의료, 법률, 건축, 교통 등 다양한 분야에 적용되고 있다[1]. 하지만, LLM에도 한계는 존재한다. 학습에 필요한 데이터양이 방

* 정회원, 계명대학교 경영정보학과

** 정회원, 계명대학교 경영빅데이터학과

대하여 소요 시간과 비용이 크고, 기존 학습 데이터와 다른 전문 분야나 지식, 최신 정보에 관한 질문에 정확한 답변을 하지 못한다. 특히, 자신이 모르는 사실을 생성하는 현상인 환각 현상을 일으킨다[2]. 생성된 내용은 사실처럼 보이지만, 실제로는 잘못된 정보인 경우가 많아 환각 현상은 생성형 인공지능(Artificial Intelligence, AI)의 대표적인 한계점이다. 이러한 현상이 발생하는 원인 중 하나는 학습 데이터에 최신 내용이나 전문 지식과 같은 특정 분야에 대한 정보의 부재이다[3]. 환각 현상을 줄이고, 실제 정보와 일치하는 답변을 얻을 수 있도록 프롬프트 컨텍스트(Prompt Context) 추가, 자기 일관성 증진[4], CoT(Chain Of Thought)[5] 기법 등 다양한 방법이 제시되고 있다. 하지만, 비즈니스 차원에서 기업 내부 정보에 대한 질문이나 학습되지 않은 최신 정보에 대한 답변은 어렵다. 이를 개선하고자 제시된 첫 번째 방법은 파인튜닝(Fine-tuning)이다. 사전 학습된 LLM에 최신 데이터나 특정 도메인 정보를 추가 학습하는 것을 말한다. 파인튜닝 방법은 전체 모델 파인튜닝(Full Fine-tuning), 부분 파인튜닝(Partial-Fine tuning), 프롬프트 투닝(Prompt Tuning), LoRA(Low-Rank Adaptation), RLHF(Reinforcement Learning from Human Feedback) 등이 있다. LLM의 경우, 대부분 모델이 매우 크기 때문에 시간과 비용적인 측면에서 전체 파라미터를 모두 업데이트하기보다는 일부 파라미터를 업데이트하는 방식을 사용한다. DALL·E, InstructGPT 등이 대표적으로 파인튜닝된 사례라고 할 수 있다. 두 번째 방법은 사용자가 필요한 정보가 담긴 문서를 직접 프롬프트 컨텍스트에 넣어주고 답변을 받는 것이다. GPT-4의 경우, 대화 시, 50페이지 정도를 기억할 수 있지만, 모든 정보를 컨텍스트에 넣는 것은 불가능하다. 이를 효율적으로 해결하는 방법이 검색 증강 생성(Retrieval Augmented Generation, RAG) 아키텍처이다. RAG는 사용

자가 질의를 할 때, 사용자 혹은 기업 정보를 저장한 데이터베이스에서 검색한 정보를 LLM에 컨텍스트로 전달하여 답변하도록 하는 방법이다. 질문과 관련된 참고자료를 미리 제공함으로써 환각을 줄이고 정확한 답변을 생성할 수 있도록 한다. LLM이 학습하지 못한 분야에서도 새로운 데이터를 학습 없이 고품질 답변을 제공할 수 있어 의료, 법률, 금융 등 비즈니스 도메인에서 활용될 수 있는 기술이다. 최근 RAG를 적용하여 LLM 한계를 개선하거나 내부 정보를 활용할 수 있도록 개발하는 등 연구가 진행되고 있다[6]. 하지만, RAG 아키텍처를 적용한 LLM 애플리케이션은 대부분 Python 기반 LangChain 프레임워크로 구축한다. 프로그래밍에 대한 기본적인 이해가 없다면, 구축하는 과정을 이해하기가 쉽지 않다.

본 연구는 RAG 아키텍처를 Python 기반의 코드가 아닌 GUI(Graphical User Interface) 기반의 래피드마이너(RapidMiner) Generative models로 쉽게 구축할 수 있음을 제시한다. 래피드마이너는 셀프서비스 분석을 지원하는 예측적 데이터 분석(Predictive Data Analytics) 플랫폼이다. 완전 GUI 방식으로 데이터 로딩부터 모델 적용까지 시각적으로 작업 흐름을 설계할 수 있어 비전문가도 쉽게 사용할 수 있다. 즉, 프로그래밍을 알지 못하더라도 RAG 구축 과정을 직관적으로 볼 수 있어 이해가 쉬울뿐만 아니라 구출도 가능하다. 본 연구는 세부적인 RAG 구축 과정을 설명하고, 구현 사례를 제시하여 생성형 AI를 구축하면서 래피드마이너의 활용성을 보여주는 것을 목표로 한다.

연구 내용은 다음과 같다. 2장은 선행 연구와 주요 개념에 관해 기술하고, 3장은 RAG 구축 방법을 상세하게 설명한다. 4장은 래피드마이너로 RAG 구축 사례를 통하여 결과를 보여주고, 5장은 연구 결론 및 한계점, 향후 연구 방향에 관해 기술한다.

II. 이론적 배경

1. RAG

RAG는 전통적인 자연어 처리(Natural Language Process, NLP) 작업에서 잘못되거나 오래된 정보를 제한하기 위한 접근법이다[7, 8]. 최근 생성형 AI 분야에서는 필요한 정보를 데이터베이스에 저장하고, 학습된 언어 모델을 통한 검색 결과를 바탕으로, 실시간으로 응답을 생성하는 기술로 설명한다[9]. RAG의 주요 요소는 검색기와 생성기 두 가지이다. 검색기는 필요한 정보를 검색하여 LLM에 전달하고, 생성기는 검색된 문서를 바탕으로 LLM이 답변을 생성한다. 특히 검색(Retrieval)의 출력은 LLM 입력에 일부가 되기 때문에 RAG의 핵심 요소이다. 고전적인 검색 방법으로 TF-IDF나 BM25[10]가 있지만, 밀도 검색(Dense Retrieval)이 쿼리와 문서를 표현할 수 있는 다차원 공간에 의미를 매칭할 수 있어 효과적인 방법이다. 이러한 검색기는 질의응답 시스템에 자주 사용된다. RAG 검색기를 통하여 LLM에 관련된 문서를 제공하고, 생성 단계에서 내용을 통합할 수 있어 답변의 품질을 높일 수 있다[11].

ChatGPT와 같은 대화형 AI의 가장 큰 문제는 기존 학습 정보의 제한으로 정확한 답변을 하지 못하는 환각 현상이 일어난다는 것이다. 이 문제를 해결하기 위해 새로운 데이터로 파인튜닝을 하거나 프롬프트에 직접 정보를 작성할 것을 제시한다. 그러나 파인튜닝은 새로운 데이터를 학습하기 위한 비용과 자원이 상당히 크고, 프롬프트에 필요한 정보를 모두 작성하는 것도 여전히 한계가 있다. 최근 연구에서 RAG는 새로운 데이터를 학습하지 않고, 최신성과 정확성이 높은 답변을 제공할 수 있어, 환각을 줄일 수 있는 대안으로 제시되고 있다. Béchard 등 [12]은 엔터프라이즈 플랫폼에 RAG와 sLLM을 결합하여 환각을 줄이고 자원이 제한된 환경에서도 시스템을 구축할 수 있음을 제시하였다. Yi 등 [6]은 기

업 내부 정보가 유출되지 않도록 직접 지식 데이터베이스를 구축하고, 질의응답 할 수 있는 RAG 기반 시스템을 구현하였다. Kumar 등 [13]은 치료 데이터의 벡터 표현을 저장하고 검색하는 벡터 데이터베이스를 활용한 RAG를 적용하여 YOLOv8에서 식별한 특정 질병을 기반으로 맞춤형 권장 사항을 제공할 수 있는 시스템을 제안하였다. RAG는 다양한 분야의 시스템뿐 아니라 LLM 아키텍처를 활용한 생성형 AI 서비스 구현에도 중요한 기술이다. 하지만, RAG를 구축하는 대부분의 연구는 Python 기반 LangChain 프레임워크를 활용하였다. 하지만, 본 연구는 코드 기반이 아닌 GUI기반의 데이터 분석 플랫폼 래피드마이너를 활용하여 RAG 시스템의 구축 방법을 상세하게 설명하고자 한다.

2. 래피드마이너

오픈 소스 데이터 도구인 래피드마이너는 랄프 클링켄버그, 잉고 미에르스와, 사이먼 피셔가 개발한 데이터 과학 소프트웨어 플랫폼이다[14]. 데이터 준비, 결과 시각화, 모델 검증 및 최적화를 포함한 모든 머신러닝 프로세스를 지원하고, 비즈니스, 상용 애플리케이션, 연구, 교육, 훈련, 신속한 프로토타이핑 개발에 사용한다. 래피드마이너는 사용자의 작성을 요구하는 코딩 작업을 제거함으로써 전달 속도를 높이고 오류를 줄이는 템플릿 기반 프레임워크를 통하여 고급 분석 솔루션을 제공한다. 자바 프로그래밍 기반으로 분석 방법을 설계할 수 있도록 GUI를 제공하는 방식으로 개발되었다. 분석과정을 설계하는 것을 "프로세스"라고 하고, 이는 여러 연산자로 구성된다. GUI 방식의 래피드마이너는 Python으로 분석하는 것과는 다르게 데이터 분석과정을 직관적으로 볼 수 있고, 드래그 앤 드롭(Drag&Drop)으로 머신러닝과 딥러닝과 같은 알고리즘을 쉽게 수행할 수 있다.

최근 생성형 AI의 급속한 발전에 따라 래피드마이너에서도 RAG 아키텍처와 생성형 AI 모델

을 활용할 수 있도록 확장형 프로그램인 Generative Models를 개발하였다. 이 프로그램을 활용하면 프로그래밍 지식이 없이 Python 기반의 Langchain 프레임워크를 통하여 구현한 RAG와 유사한 결과물을 빠르게 구축할 수 있다는 장점이 있다. 또한, Huggingface나 OpenAI에 있는 수십만 개의 LLMs을 활용할 수 있고, 프롬프트 생성, 임베딩, 벡터 데이터베이스 등 RAG도 구현할 수 있다.

III. 연구 방법

RAG 모델은 풍부한 정보를 가지고 있는 대규모 문서 데이터베이스에서 관련 정보를 검색하고, 언어 모델을 통하여 정확하고 상세한 답변을 생성한다. RAG 구현은 크게 두 단계로 나눠서 수행되는데, 첫 번째 단계는 사전 준비 단계이고, 두 번째는 검색 및 답변 생성 단계이다. 사전 준비 단계는 총 4개의 단위 작업이 필요하다. 데이터를 수집하여 로드하고, 청크(Chunk) 단위로 텍스트를 분할하여 숫자인 벡터로 전환하는 임베딩 과정을 거쳐 벡터 저장소에 저장한다. 데이터 로드는 검색하거나 생성할 내용에 기반한 데이터를 수집하고 준비하는 과정으로 문서, 웹페이지, 뉴스, 코드 등이 데이터 소스가 될 수 있다. CSV, PDF, WORD 그리고 JSON 파일 등 다양한 형식의 문서를 로드할 수 있다. 텍스트 분할은 청크라는 단위로 텍스트를 나눠 정보 검색을 용이하게 하는 작업이다. 청크는 문장 또는 문단과 같이 데이터를 처리할 수 있는 가장 작은 조각을 의미한다. 청크의 크기나 나누는 방식에 따라 검색의 정확도가 차이가 날 수 있기 때문에 Small2big, Sliding Windows 등 검색 품질을 향상할 수 있는 분할 방법을 사용한다. 임베딩 단계는 생성된 청크를 벡터 형태로 변환하는 과정이다. 사전 학습된 언어 모델을 사용하여 텍스트의 의미를 벡터로 수치화한다. 사용자의 질문과 청크의 의미를 효과적으로 매칭 할 수 있는 임베딩 모델을 사용하여야 한다. 사전 준비 단계의 마지-

막은 벡터 스토어에 저장하는 것이다. 벡터 데이터베이스는 임베딩 벡터를 메타데이터와 함께 저장하여 문서를 효율적으로 검색할 수 있도록 한다[5]. 인덱스 유형, 수십억 개 규모의 벡터 지원, 하이브리드 검색, 클라우드 네이티브 기능 등에 따라 다양한 벡터 데이터베이스가 오픈 소스로 제공되고 있고, 대표적으로 FAISS(Facebook AI Similarity Search), Chroma, Qdrant, Milvus 등이 있다.

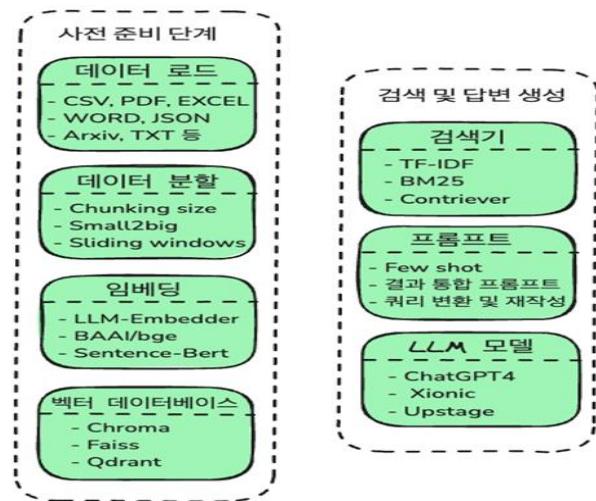


그림 1. RAG 구현 단계

두 번째 단계는 검색 및 답변 생성 단계이다. 질문을 하면, 벡터 데이터베이스에서 관련된 청크를 검색하고, 이를 바탕으로 언어 모델을 위한 질문을 구성하는 프롬프트를 만든다. 구성된 프롬프트를 LLM에 전달하여 답변을 생성한다. 검색기는 사용자 질문과 벡터 데이터베이스에 저장된 청크 간의 유사성을 기반으로 상위 K개 문서를 선별하는 과정이다. 검색된 문서를 사용하여 질문에 대한 답변을 생성하게 되는데, 질문에서 정보가 부족하거나, 의미가 불분명한 경우에 검색 성능이 저하되기도 한다. 검색 품질을 높이기 위하여 질문 재작성, 질문 분해, 가상 문서 생성 등 다양한 방법을 제시하고 있다. 대표적 검색 방법으로 BM25, HyDE[9] 등이 있고, 최근에는 어휘 기반 검색과 벡터 검색을 결합한 방법도 제안되었다[15]. 프롬프트 단계는 사용자 질문과

검색된 문서 간의 정보를 통합하여 LLM에 제공하는 컨텍스트를 만드는 작업이다. 사용되는 정보나 내용이 중복적이고, 문장이 과도하게 길거나 짧을 경우, LLMs가 생성하는 답변이 부정확할 수 있다. 이를 방지하기 위하여 LLMs가 답변 품질을 높일 수 있는 프롬프트를 작성해야 한다. 정보 검색 결과를 효과적으로 통합하는 프롬프트 템플릿을 설계하거나 답변 형식에 대한 예시를 포함하는 Few-Shot 프롬프트 방법 등이 있다. 마지막으로 LLMs를 통하여 답변 생성하는 과정이다. 생성할 텍스트의 종류나 언어, 길이 등을 지정할 수 있다. 대표적으로 LLMs은 OpenAI의 ChatGPT, Anthropic의 Claude 등이 있고, 국내 기업의 경우에는 업스테이지(Upstage)의 Solar 모델, 사이오닉에이아이(XionicAI)의 Xionic 모델 등이 있다.

그림 2는 구축하고자 하는 RAG 아키텍처를 나타낸다. 사용자가 질문을 하면 임베딩을 통하여 벡터값으로 변환하고, 벡터 데이터베이스에서 벡터값을 비교하여 유사한 문서를 찾는다. 검색된 문서와 사용자 질문에 내용을 통합하는 프롬프트를 작성하여 LLM에 입력으로 전달하고, LLM은 답변을 생성하여 사용자에게 제공한다.

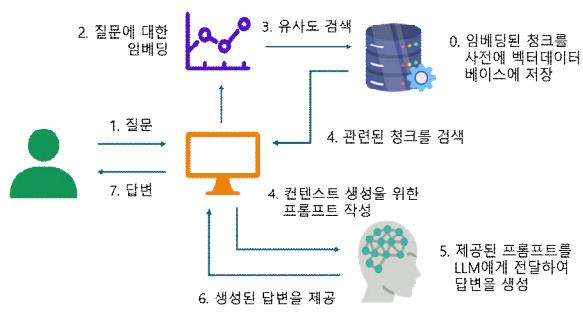


그림 2. RAG 모델 작동 방식

IV. 시스템 구현

본 장은 3장에서 소개한 RAG 아키텍처에 따라 데이터 예측 플랫폼인 래피드마이너를 활용하여 RAG를 구현한다. 또한, 사례를 통하여 단계별로 구현하는 방법과 고려사항을 제시하고자 한다.

1. 구현 환경

본 사례에서 제안하고자 하는 RAG 구축은 그림 2와 같은 형태로 논문을 검색할 수 있는 프레임워크이다. 문서를 청크 단위로 나누고, 임베딩으로 변환 후, Qdrant에 저장하는 단계를 설명하며, 컨텍스트 기반 답변을 효과적으로 제공할 수 있도록 프롬프트 설계하여 답변 결과를 살펴본다. 개발 프로그램은 래피드마이너를 활용하고, LLM은 OpenAI의 API를 사용한다. 각 구현 요소별 개발환경은 다음과 같다.

표 1. 개발환경에 사용된 구성요소

구성요소	사용
Embedding	BAAI/bge-base-en
Vector DB	Qdrant
LLM 모델	ChatGPT-3.5-turbo

2. 단계별 구현 결과

가. 벡터 데이터베이스 설치

임베딩 기반으로 텍스트 문서를 저장하고 검색하는 가장 좋은 방법은 벡터 데이터베이스를 활용하는 것이다. RAG 구축을 위하여 많은 벡터 데이터베이스가 있지만, 래피드마이너는 Qdrant와 Milbus를 지원하고 있다. 본 연구는 Qdrant를 사용하여 RAG를 구축한다. 먼저, RAG 환경 설정을 위하여 Qderant를 설치한다. 개발 및 테스트를 위하여 도커(Docker)에서 Qdrant를 설정한다. Qdramt 컨테이너 이미지를 실행하는 명령어를 입력하여 설치한다. \$(pwd)/path/to/dataDocker 구성을 수정하는 명령어를 입력하면 Qdrant 설치와 실행이 완료되고, 래피드마이너에서 연결 작업을 수행할 수 있다. 그림 4와 같이 Create Connection 메뉴 실행하여 연결 타입은 Dictionary를 선택하고, 연결 이름을 작성한다. 생성된 Dictionary Connection에 키 이름, 호스트, 포트 번호를 입력하여 연결을 완료한다.

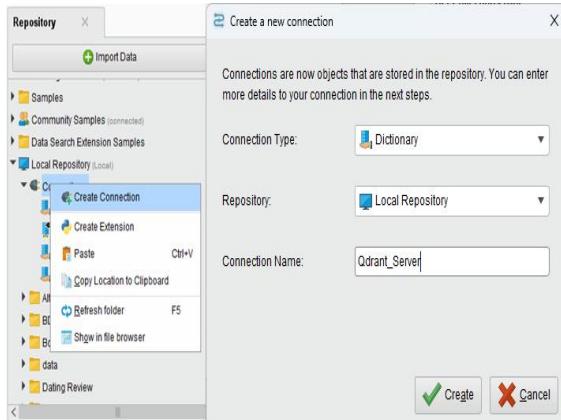


그림 4. 래피드마이너 Qdrant 연결

나. 임베딩 생성 및 저장

벡터 데이터베이스 설정 완료 후, 데이터셋을 불러오고, 임베딩으로 변환하여 저장하는 사전 단계를 수행한다. XLSX, CSV, TXT 그리고 PDF와 같이 다양한 형태의 데이터를 로드할 수 있고, 웹페이지나 유튜브 같은 동영상 형태 데이터도 사용할 수 있다. 구조화되지 않은 데이터를 읽어오는 작업이 선행되어야 한다. 본 연구는 엑셀 파일로 된 문서를 로드하여 텍스트를 추출하고 임베딩으로 변환한다. 임베딩에 사용하는 모델은 BAAI/bge-base-en으로, 768차원의 벡터를 출력한다. 데이터를 로드하여 임베딩 변환한 다음 그 결과를 저장한다.

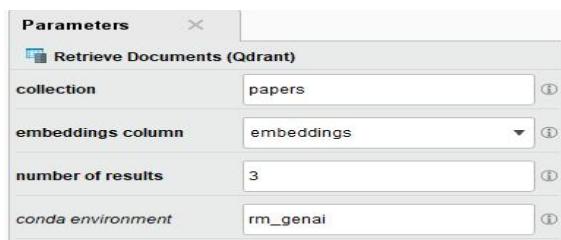


그림 5. 임베딩 생성

생성된 임베딩 결과를 가져와 Qdrant에 저장하기 위하여 Retrieval Qdrant와 Insert Documents 오퍼레이터를 연결하여 Qdrant에 저장한다. 그림 6은 임베딩을 Qdrant에 저장하는 프로세스이다.

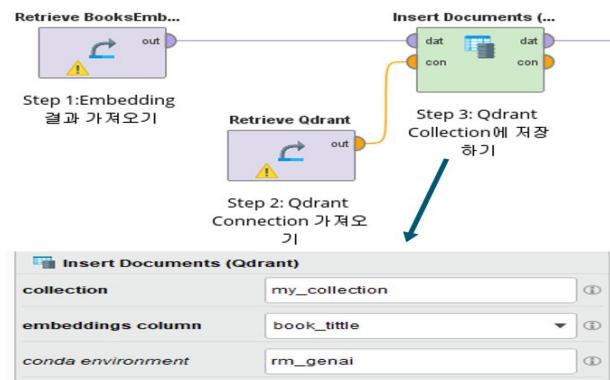


그림 6. 임베딩을 Qdrant에 저장

다. OpenAI를 활용한 RAG 구축

Qdrant에 임베딩 저장을 완료하였다면, 본격적으로 RAG 구축을 할 수 있다. 그림 7은 본 연구에서 구축한 RAG의 프로세스이다.

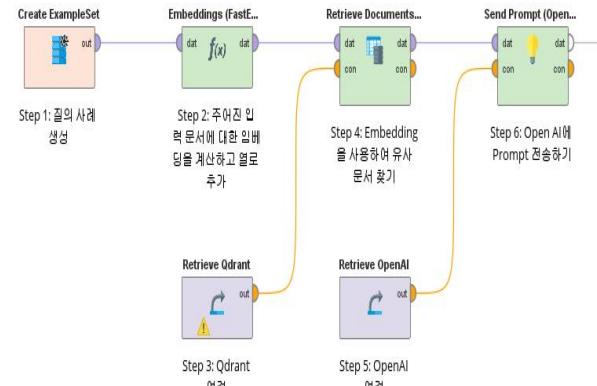


그림 7. RAG 구축 프로세스

먼저, 사용자가 질문하는 단계로, 질의 사례 생성 오퍼레이터를 만든다. Create ExampleSet이라는 오퍼레이터에서 파라미터 값으로 사용자의 질문을 작성하여 넣는다. 그림 8은 연구의 예제로, 인간의 성별 편견이 언어와 텍스트 생산에 반영되는지에 대한 질의이다.

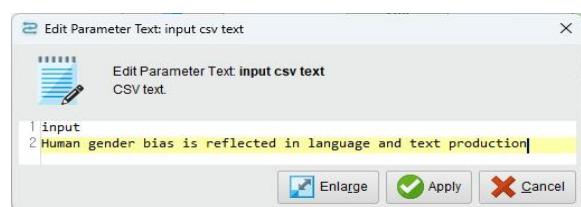


그림 8. 질의 사례

질의 사례에 FastEmbed 오퍼레이터를 사용하여 임베딩 생성 작업을 수행한다. 임베딩 모델 차원의 크기가 벡터 데이터베이스에 저장된 차원의 크기가 동일할 수 있도록 설정한다. 임베딩 변환 후, Retrieve Qdrant 오퍼레이터를 Retrieve Documents(Qdrant) 오퍼레이터에 연결하여 Qdrant에서 질문 사례와 유사한 문서를 찾을 수 있도록 사용한다. 데이터가 저장된 Collection, 질의 결과 개수 등을 지정한다.

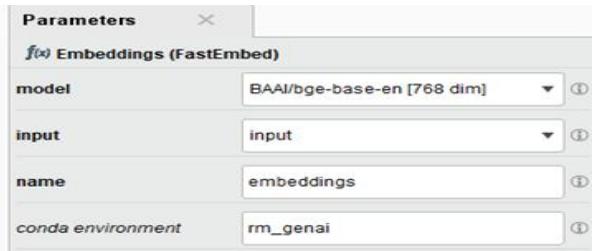


그림 9. 질의 사례 임베딩 변환

본 연구는 사전 작업에서 임베딩을 저장한 papers라는 collection을 지정하고, 검색 결과 개수는 3개로 설정한다. 마지막으로 OpenAI의 ChatGPT-3.5-turbo를 LLM 모델로 사용하기 위하여 Retrieve OpenAI와 Send Prompt(OpenAI) 오퍼레이터를 연결한다. Send Prompt(OpenAI)를 사용하여 서비스를 요청하고 타입, 모델명, 프롬프트 등을 작성한다.

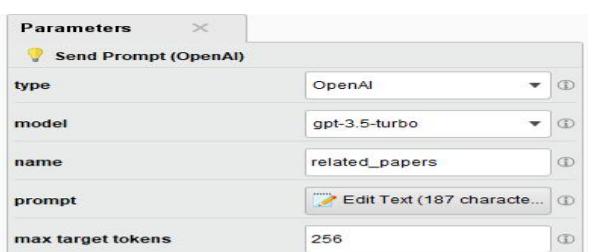


그림 10. LLM 답변 생성을 위한 파라미터 설정

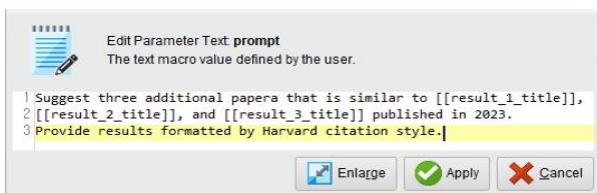


그림 11. LLM 답변 생성을 위한 프롬프트 작성

input	embeddings	result_1_sc...	result_2_sc...	result_3_sc...
Human gend...	-0.00587760...	0.887	0.876	0.872

그림 12. 구축된 RAG 실행 결과

RAG을 구축한 프로세스를 실행하면 그림 11과 같이 결과를 확인할 수 있다. 질의 사례에 대한 임베딩 값은 -0.0058776으로 나타나고, 이 값과 가장 유사한 3개 문서와의 유사도에 대한 수치도 0.887, 0.876, 0.872로 확인할 수 있다. 검색된 문서와 생성한 답변의 상세한 결맞값은 표 2, 3에서 확인할 수 있다. Retrieve Documents(Qdrant)에서 설정한 검색 결과 수와 동일하게 Qdrant에서 질의 사례의 검색 결과는 3개였고, 생성된 답변도 프롬프트에 작성한 대로 3개의 답변을 받았다.

표 2. Qdrant에서 검색된 문서 결과

문서 검색 결과
result_1_title: Second Order WinoBias (SoWinoBias) Test Set for Latent Gender Bias Detection in Coreference Resolution
result_2_title: Extending Challenge Sets to Uncover Gender Bias in Machine Translation: Impact of Stereotypical Verbs and Adjectives
result_3_title: Towards Measuring Fairness in Speech Recognition: Casual Conversations Dataset Transcriptions

표 3. 생성된 답변 결과

생성된 답변
Zhao, J., Wang, T., & Chang, K. (2023). Detecting Bias in Visual Question Answering: A New Challenge Dataset. In Proceedings of the 12th International Conference on Natural Language Processing (ICONLP 2023) (pp. 100–110).
Patel, S., Jackson, L., & Lee, H. (2023). Exploring Gender Bias in Hate Speech Detection Models: A Comparative Analysis. In Proceedings of the 20th International Conference on Artificial Intelligence and Data Science (ICAIDS 2023) (pp. 75–85).
Kim, Y., Liu, M., & Smith, J. (2023). Examining Bias in Sentiment Analysis: The Role of Social Media Texts. In Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2023) (pp. 220–230).

III. 결 론

1. 구현 사례 고찰

본 연구는 데이터 예측 플랫폼인 래피드마이너와 벡터 데이터베이스 Qdrant를 활용하여 임베딩 데이터를 저장하고, 문서 검색과 답변을 생성할 수 있는 RAG를 구현하였다. 구축한 RAG에 성별 편견이 언어와 텍스트 생산에 어떻게 반영되는지를 탐구하는 질의 사례를 중심으로 시스템을 테스트하여 실제 검색 결과를 확인하였다. 본 연구는 RAG 구축 과정을 래피드마이너 Generative Models로 코드 없이 구현할 수 있는 구체적인 사례를 보였다. 이는 프로그래밍에 대한 이해도가 없는 비전문가도 RAG 아키텍처를 구현할 수 있다는 데 큰 의의가 있다. 대부분 RAG 구현 사례는 Python과 Langchain 프레임워크를 사용하는 코드 기반으로 구축한다. 프로그래밍에 대한 이해가 없다면, RAG를 구축하는 것은 쉽지 않다. 래피드마이너는 분석을 설계할 수 있는 GUI를 제공하기 때문에 코드 없이 드래그 앤 드롭만으로 RAG를 구축하는 프로세스를 만들 수 있다. 설계된 분석 과정을 직접 눈으로 확인할 수 있어 직관적으로 이해하기도 쉽다.

본 연구에서는 다음과 같은 시사점이 있다. 첫 번째로 RAG 아키텍처에 대하여 직관적으로 이해할 수 있는 구현 방법을 제시하였다. 두 번째로 래피드마이너라는 소프트웨어를 통하여 코드 없이 RAG 구조를 구현할 수 있는 구체적인 사례를 소개하였다. 최신 LLM과 벡터 데이터베이스 기술을 코드 없이 어떻게 구현할 수 있는지 RAG 적용 사례를 제시하여 생성형 AI 기술을 더 쉽게 활용할 수 있다는 가능성을 확인하였다. 대부분 연구에서 RAG 모델을 프로그래밍 기반으로 구현하고 있는 상황에서, 본 연구의 결과는 이를 활용하고자 하는 많은 이해 관계자들이 생성형 AI 기술을 코드 없이 구현할 수 있는 참고 자료로 활용될 수 있다.

2. 한계점 및 향후 연구

본 연구는 데이터 분석 플랫폼인 래피드마이너를 통하여 LLM을 활용한 RAG 모델을 구축하는 방법을 제시하였다. 특히, Python 프로그래밍에 대한 지식 없이 RAG를 구현할 수 있는 방법을 제안하였다. 하지만, 여러 가지 한계점은 존재한다.

첫 번째는 본 연구는 프로그래밍 없이 RAG를 구축하는 방법을 제시하였지만, 프레임워크별 RAG 성능 비교에 대하여 정량적으로 제시하지 않았다. 향후 연구에서, 코드 기반 RAG와 래피드마이너로 구축한 RAG에 대한 성능을 비교할 필요가 있다.

두 번째는 벡터 데이터베이스는 Qdrant, LLM은 ChatGPT-3.5-turbo 외에도 RAG 모델을 최적화할 수 있는 여러 가지 요소가 있지만, 고려하지 않았다. RAG에서 있어서 벡터 데이터베이스의 선택은 매우 중요하다. 오픈 소스 DB인 Weaviate, Vespa, 안정적인 클라우드 호스팅 기반 Pinecone 등 다양한 벡터 데이터베이스가 존재하지만, 본 연구는 Qdrant DB만을 선택하여 구축하였다. LLM 경우, 강력한 성능을 가진 ChatGPT-4o, Claude, Llama3 등을 사용하지 않았다. 이를 활용하여 RAG 모델을 구축한다면 높은 성능을 보일 수 있을 것으로 기대한다.

마지막으로, LLM 활용으로 인한 비용 문제이다. ChatGPT-4와 LLM은 좋은 성능을 보이지만, 토큰 사용 비용이 커진다. 비용 부담을 줄이려면, Llama3과 같이 오픈된 모델을 활용하거나, 작은 크기의 언어 모델을 활용해야 한다. 기존 LLM보다는 성능이 떨어질 수 있지만, 효과적인 결과를 얻기 위하여 파인튜닝과 같은 방법을 고려해 볼 수 있다. 래피드마이너 Generative Models는 LLM을 파인튜닝할 수 있는 오퍼레이터도 제공한다. 래피드마이너를 통하여 코드 없이 LLM을 파인튜닝하여 RAG 모델 성능을 개선하는 것을 향후 과제로 제안할 수 있다.

REFERENCES

- [1] Chen, Z.Z., Ma, J., Zhang, X., Hao, N., Yan, A., Nourbakhsh, A., Yang, X., McAuley, J., Petzold, L. and Wang, W.Y., "A Survey on Large Language Models for Critical Societal Domains: Finance, Healthcare, and Law," arXiv preprint arXiv:2405.01769, May 2024.
- [2] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y. and Wang, L., "Siren's song in the AI ocean: a survey on hallucination in large language models," arXiv preprint arXiv:2309.01219, Sep. 2023.
- [3] 정천수. "LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: 검색증강생성 모델과 LangChain 프레임워크 기반," *지능정보연구*, 제29권, 4호, 129~164쪽, 2024년 12월
- [4] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A. and Zhou, D., "Self-consistency improves chain of thought reasoning in language models," arXiv preprint arXiv:2203.11171, Mar. 2023.
- [5] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V. and Zhou, D., "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems*, 35, pp. 24824–24837, 2022.
- [6] Yi, G.W. and Kim, S.K., "Design of a Question-Answering System based on RAG Model for Domestic Companies," *Journal of The Korea Society of Computer and Information*, Vol.29, No.7, pp. 81~88 , Jul. 2024.
- [7] Izacard, G. and Grave, E., "Leveraging passage retrieval with generative models for open domain question answering," arXiv preprint arXiv:2007.01282, Feb. 2021.
- [8] Shuster, K., Poff, S., Chen, M., Kiela, D. and Weston, J., "Retrieval augmentation reduces hallucination in conversation," arXiv preprint arXiv:2104.07567, Apr. 2021.
- [9] Gao, L., Ma, X., Lin, J. and Callan, J., "Precise zero-shot dense retrieval without relevance labels," arXiv preprint arXiv:2212.10496, Dec. 2022.
- [10] Sparck Jones, K., "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, Vol.28, No.1, pp. 11~21, Jan. 1972.
- [11] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W.T., Rocktaschel, T. and Riedel, S., "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, 33, pp. 9459~9474, 2020.
- [12] Bechar, P. and Ayala, O.M., "Reducing hallucination in structured outputs via Retrieval-Augmented Generation," arXiv preprint arXiv:2404.08189, Apr. 2024.
- [13] Kumar, S.S., Khan, A.K.M.A., Banday, I.A., Gada, M. and Shanbhag, V.V., "Overcoming LLM Challenges using RAG-Driven Precision in Coffee Leaf Disease Remediation," *2024 ICETCS*, pp. 1~6, Bengaluru, India, Apr. 2024.
- [14] Pynam, V., Spanadna, R.R. and Srikanth, K., "An extensive study of data analysis tools (rapid miner, weka, r tool, knime, orange)," *Int. J. Comput. Sci.* Vol. 5, No. 9, pp. 4~11, 2018.
- [15] Sawarkar, K., Mangal, A. and Solanki, S.R., "Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers," arXiv preprint arXiv:2404.07220, Aug. 2024.
- [16] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J. and Wang, H., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, Dec. 2023.

저자 소개



양치복(정회원)

2016년 계명대학교 통계 학과 학사 졸업.
2019년 계명대학교 경영정보 학과 석사 졸업.
2024년 계명대학교 경영정보 학과 박사 졸업.

<주관심분야 : 자연어 처리, 머신러닝, 검색증강생성>



김양석(정회원)

1995년 서울시립대학교 경제학과 학사 졸업.
2004년 University of Tasmania 컴퓨터 공학 석사 졸업.
2009년 University of Tasmania 컴퓨터 공학 박사 졸업.

<주관심분야 : Machine Learning and Data Analytics, Recommender Systems, Knowledge Engineering>