ISSN 2288-4866 (Print)
ISSN 2288-4882 (Online)
https://www.jiisonline.org
J Intell Inform Syst 2023 December: 29(4): 129~164 http://dx.doi.org/10.13088/jiis.2023.29.4.129

LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: RAG모델과 LangChain 프레임워크 기반

정천수
삼성SDS AI Automation Team
(csu.jeong@samsung.com)

최근 생성형 AI 기술의 발전으로 인해 대형 언어 모델(Large Language Model, LLM)의 활용 및 도입이 확대되고 있는 상황에서 기존 연구들은 기업내부 데이터의 활용에 대한 실제 적용사례나 구현방법을 찾아보기 힘들다. 이에 따라 본 연구에서는 가장 많이 이용되고 있는 LangChain 프레임워크를 이용한 LLM 애플리케이션 아키텍처를 활용하여 생성형 AI 서비스를 구현하는 방법을 제시한다. 이를 위해 LLM의 활용을 중심으로, 정보 부족 문제를 극복하는 다양한 방법을 검토하고 구체적인 해결책을 제시하였다. 이를 위해 파인튜닝이나 직접 문서 정보를 활용하는 방법을 분석하며, 이러한 문제를 해결하기 위한 RAG 모델을 활용한 정보 저장 및 검색 방법에 대해 주요단계에 대해 자세하게 살펴본다. 특히, RAG 모델을 활용하여 정보를 벡터저장소에 저장하고 검색하기 위한 방법으로 유사문맥 추천 및 QA시스템을 활용하였다. 또한 구체적인 작동 방식과 주요한 구현 단계 및 사례를 구현소스 및 사용자 인터페이스까지 제시하여 생성형 AI 기술에 대한 이해를 높였다. 이를 통해 LLM을 활용한 기업내 서비스 구현에 적극적으로 활용할 수 있도록 하는데 의미와 가치가 있다.

주제어 : 대규모언어모델, 생성형 AI, 검색 증강 생성, LangChain, 벡터저장소, QA 시스템

논문접수일 : 2023년 10월 16일 논문수정일 : 2023년 11월 28일 게재확정일 : 2023년 11월 29일
원고유형 : Regular Track 교신저자 : 정천수

1. 개요

최근 시장 조사 기관인 가트너는 2023년 8월에 발표한 하이프 사이클 보고서를 통해 생성형 AI에 대한 글로벌 기대치가 정점에 달하였다고 밝혔다. 해당 보고서는 차기 2~5년 내에 생성형 AI가 혁신적 성과를 이루어 인간의 생산성과 기계의 창의성에 새로운 시대가 열릴 것이라고 전망했다(Gartner, 2023). 이는 OpenAI의 GPT 시리즈와 같은 대형 언어 모델(Large Language Model, LLM)들이 자연어의 생성 및 이해 분야에서 주목할만한 발전을 이뤄낸 결과이며, ChatGPT를 시작으로 이미지 생성 AI인 Stable Diffusion과 Midjourney에 이르기까지 생성형 AI 기술이 대중화되고 있음을 시사한다(정천수, 2023c). 대형 언어 모델은 방대한 텍스트 데이터 학습을 통해 복잡한 자연어 작업을 수행할 수 있으며, 이를 통해 고객 서비스, 창의적 콘텐츠 생성, 질의응답 등 다양한 분야에 적용되고 있다.

하지만 생성형 AI는 아직 다양한 한계를 가지고 있다. LLM은 방대한 양의 데이터를 학습해야 하기 때문에 많은 비용과 시간이 소요되고, LLM은 새로운 데이터에 대한 적응력이 떨어지기 때문에 기존에 학습한 데이터와 관련성이 없는 질문에는 정확한 답변을 제공하기 어렵다. OpenAI에서 서비스하고 있는 ChatGPT는 환각(Hallucination)을 일으키는 경향이 있다. 즉, 자신이 모르는 사실은

적당히 지어내서 얘기하는 것이다. 따라서 언뜻 그럴듯해 보이지만 실제로는 잘못된 정보일 경우가 많다. 최신 정보와 상충되는 답변을 생성할 수도 있으며 이러한 환각은 대표적인 생성형 AI의 한계이며 이전에 학습한 내용을 사용하므로 새로운 문제나 도메인에 대한 이해도가 낮을 수 있다(정천수, 2023c). 학습된 데이터의 부재로 인한 정보 부족은 환각 현상을 발생시키는 원인 중 하나 임으로 환각을 줄이기 위한 접근 방법으로는 프롬프트에 컨텍스트 추가, 자기 일관성(Self-Consistency) 증진, CoT(Chain Of Thought) 기법, 모델에게 간결한 답변을 요청하는 등의 방법들을 통해 환각을 최소화하고 실제 데이터와 일치하는 응답을 얻을 수 있는 노력이 진행되고 있다. 또한 LLM의 정보 부족으로 인한 제한된 답변 능력을 보인다. 예를 들어, GPT 3.5모델은 2021년 9월까지 정보만 학습되어 있고 2023년 11월에 발표한 GPT-4 Turbo모델은 2021년 이후부터 2023년 4월까지 일어난 일에 대한 정보만을 반영하여 이후의 데이터가 없으므로 최신 뉴스에 대한 답변을 제공할 수 없다.

또한 오픈 되어 있는 외부정보를 이용하여 답변을 하기 때문에 기업차원에서 비즈니스 내부 정보에 대한 질문 시 이에 대한 답변을 얻기 위한 방법의 필요성도 대두되고 있다.

이런 문제를 해결하기 위한 첫 번째 방안으로 사전 학습된 LLM에 신규 데이터를 추가하여 파인튜닝(Fine-tuning)하는 것으로, 특정 도메인의 새로운 데이터나 변경된 정보를 추가 학습 한다. 초거대 AI일 경우 전체 파라미터(Parameter)를 모두 업데이트하기는 어렵다. 그래서 P-tuning, Prefix tuning, Prompt tuning, LoRA같이 일부 파라미터만 학습하는 방식을 사용한다. OpenAI에서도 이러한 한계를 극복하고자 GPT-3.5 Turbo 모델에 대한 파인튜닝이 가능하게 2023년 8월에 기능을 업데이트 하였고 GPT-4모델에 대한 파인튜닝 기능은 2023년 말에 제공될 예정이다. 현재 기준으로 GPT-3.5-Turbo 모델의 파인튜닝 비용은 초기 학습비용과 사용 비용이라는 두 가지 버킷으로 분류되며, 예를 들어, 3개의 에포크(Epoch) 동안 훈련된 100,000개의 토큰 훈련 파일이 있는 GPT-3.5-Turbo 파인튜닝 작업의 예상 비용은 $2.40이나 된다(RevFactory, 2023). 또 다른 대안으로는 사용자가 원하는 정보가 담긴 문서를 직접 프롬프트 컨텍스트(Context)에 넣어주고 원하는 답변을 얻는 것이다. 그러나 모든 정보를 컨텍스트에 일일이 넣어주는 것은 현실적으로 불가능하며 GPT-3.5가 대화 시 4~5페이지를 기억한다면 GPT-4는 50페이지를 기억한다(박창호, 2023). 이러한 상황에서는 사용자나 기업이 가진 정보를 데이터베이스에 저장해두고, 사용자의 질의가 들어올 때(예를 들어, 챗봇을 통해 회사 '근태복무규정'에 대한 질문을 했을 때) 관련된 정보를 검색하여 해당 정보가 담긴 문서들을 프롬프트를 통해 LLM에 전달하는 방식이 더 효율적이다. 예를 들어 PDF 문서를 업로드하고 질문을 하면, PDF에서 해당하는 정보를 찾아서 답변을 주는 방식이다. 이것이 두 번째 방법인 '검색 증강 생성'(Retrieval Augmented Generation, RAG) 서비스 아키텍처이다. 이렇듯 RAG는 LLM에게 미리 질문과 관련된 참고자료를 알려줌으로써 환각을 줄이고 보다 정확하게 대답을 생성할 수 있도록 한다. 이렇게 RAG 아키텍처는 LLM의 정보 부족 문제를 해결하고, 새로운 데이터 학습 없이도 고품질의 응답을 제공할 수 있는 잠재력을 가지고 있다. 이렇게 RAG 모델은 사용자의 질의에 더 정확하고 적절한 답변을 제공함으로써 실제 비즈니스 환경에서도 유용하게 활용될 수 있는 기술로서 다양한 비즈니스 도메인에 적용할 수 있어 LLM의

성능을 향상시키는 방법이 될 수 있다. 이전부터 언어 모델 성능을 향상시키기 위해 검색을 활용하는 시도가 많았으며, 대표적으로 2021년에 딥마인드에서 발표한 자체 DB에서 정보를 찾는 RETRO와 OpenAI가 같은 해에 발표한 Bing에서 검색을 하는 WebGPT가 있다. 그러나 현재는 RAG 아키텍처가 산업계에서 주목받고 있는데 이는 비약적으로 향상된 문맥 내 학습(In-context learning) 능력과 모델 학습이 따로 필요하지 않은 편리함 때문이다.

이와 같이 ChatGPT의 한계 및 내부정보를 이용하기 위한 방법으로 파인튜닝을 통해 특정 도메인의 새로운 데이터로 추가 학습하거나, RAG를 이용하여 LLM에게 미리 질문과 관련된 참고자료를 알려주어 ChatGPT의 정확성과 신뢰도를 향상시킬 수 있다. 하지만 최근 대두되고 있는 문제 해결방법으로 실제 적용사례나 구현방법을 찾아보기 힘들다.

이런 배경에서 본 연구에서는 생성형 AI의 발전을 촉진하고 한계를 극복하기 위한 방법을 모색하며 그 방안으로 RAG 아키텍처를 활용한 LLM 애플리케이션을 쉽게 구현할 수 있도록 절차와 방법을 제안하고 구현사례를 제시한다. 이를 위해 본 연구에서는 먼저 LLM의 정보 부족 문제를 직접 문서 정보 활용을 통해 극복하는 방법을 검토한다. 그리고 RAG 모델의 구체적인 작동 방식과 주요 단계에 대해 자세히 알아보고, 벡터 DB를 활용한 정보 저장 및 검색 방법에 대해 논의한다. 또한, LLM에게 적절한 프롬프트를 제공하는 방법과 이를 위한 오케스트레이션 프레임워크에 대한 설명을 포함한다. 이에 따라 본 연구에서는 세부적인 구현 방법과 사용 가능한 도구들을 소개하며, 실제로 RAG 모델을 구현하는 과정을 상세히 설명한다. 최신 LLM 모델과 벡터 DB 기술의 활용 가능성을 고려하여, 이러한 기술들을 어떻게 조합하고 최적화할 수 있는지를 탐구한다. 또한 여러 비즈니스 도메인에서 RAG 모델을 적용한 구현 코드 및 UI를 제시하고, 이를 통해 RAG 모델의 실제 적용 가능성과 생성형 AI 서비스의 실제 비즈니스 환경에서의 적용 가능성을 알아보고, 생성형 AI 기술의 발전과 산업적 활용을 촉진하고자 하는데 기여하고자 한다.

2장에서는 이론적 배경으로 본 연구를 위한 선행연구 및 주요 관련 개념들에 대하여 기술하고 있으며, 생성형 AI 및 LLM의 국내외 동향에 대하여 알아보고, RAG 모델을 적용하기 위해 LLM의 정보 부족 문제를 극복하는 방안에 대해 기술하였다. 3장에서는 구체적인 구현 방법으로 다양한 오픈소스 및 상업용 도구들을 활용하여 RAG 모델을 구현하는 방법을 설명하고, 기존의 데이터 파인튜닝과 직접 문서 정보 활용을 비교하여 각각의 장단점을 분석하고, 비용과 효율성을 고려하여 최적의 방법을 선택하여 제안하고 있으며, RAG 모델의 핵심인 정보 검색 및 활용 방법을 자세히 탐구하여 벡터 DB를 활용하여 문서 정보를 임베딩(Embedding)하여 저장하고 검색하는 과정을 설명하고, 검색 결과를 프롬프트로 변환하여 LLM에게 전달하는 방법을 제안한다. 또한 오케스트레이션 프레임워크를 활용하여 사용자의 질의와 검색 결과를 효과적으로 조합하여 최적의 답변을 생성하는 방법을 설명한다. 4장에서는 최신 LLM 모델과 벡터 DB 기술을 활용하여 어떻게 구현할 수 있는지 비즈니스 도메인에서 실제로 RAG 모델을 적용한 다양한 구현 코드를 제시하여 생성형 AI 기술의 현실적인 활용 가능성을 확인하고자 한다. 마지막으로 결론인 5장에서는 연구결과 및 한계점, 향후 연구방향에 대하여 기술하고 있다.

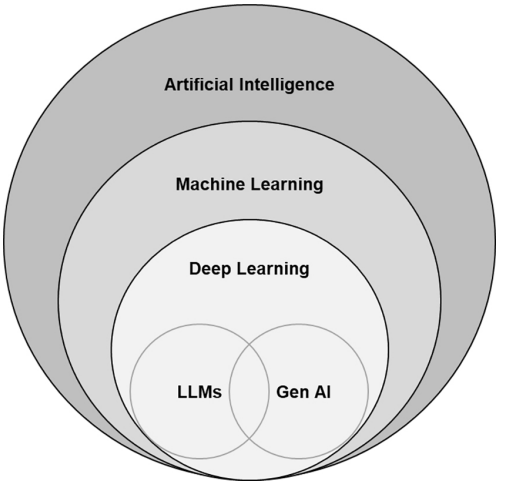
2. 이론적 배경

본 연구를 위해 생성형 AI 및 LLM과 관련된 자료에 대하여 최근에 나온 주요 연구논문 및 저널, 기사, 도서를 조사하였으며 본 장에서는 LLM 및 생성형 AI 전반에 대해 살펴보도록 한다. 먼저 생성형 AI 개념 및 응용분야에 대하여 상세하게 알아보고, 본 논문에서 다루는 LLM에 적용되는 요소기술 및 프레임워크, RAG 부문, 선행연구 고찰 영역으로 나누어 설명한다.

2.1. 생성형 AI 개념

생성형 AI(Generative AI)는 방대한 양의 데이터를 학습하여 텍스트, 이미지, 오디오, 비디오 등 새로운 콘텐츠를 만들어내는 인공지능 기술이다. 2022년 11월에 출시된 ChatGPT는 대규모의 데이터를 학습한 언어 모델 기반의 서비스인 반면, DALL-E와 Midjourney는 이미지 생성에 특화된 모델을 활용한다. 이러한 생성형 AI는 생성되는 콘텐츠의 종류에 따라 언어 모델, 이미지 모델, 비디오 모델 등으로 분류되며, 최근에는 텍스트와 이미지를 동시에 학습하는 멀티모달(Multi-modal) 모델들이 발전하고 기초모델(Foundation Model)로서 주목받고 있다. 이들 멀티모달 모델은 기능과 성능이 지속적으로 향상되는 추세에 있다.

LLM과 생성형 AI는 <그림 1>과 같이 AI의 딥러닝 안에 포지셔닝하고 있으며(Mayank, S., 2023), 컴퓨터가 짧은 프롬프트에 응답하여 텍스트, 오디오, 비디오, 이미지 및 코드와 같이 이전에 생성된 콘텐츠를 사용하여 새로운 콘텐츠를 생성할 수 있도록 하는 비지도(Unsupervised) 및 반지도(Semi-supervised) 알고리즘을 포함하는 컴퓨터 과학의 한 분야로 정의하고 있다(IDC, 2023). 최근 몇 년 동안 생성형 AI 기술은 급속한 발전을 이루었고, 다양한 산업에서 새로운 기회를 창출하고 있다. 생성형 AI는 기존 데이터를 단순히 가공하거나 분석하는 것이 아니라, 새롭고 독창적인 콘텐츠를 생성하는 AI 접근 방식이다. 생성형 AI 모델은 패턴을 학습하고 훈련 데이터와 유사한 새로운 출력을 생성하기 위해 대규모 데이터 세트에서 훈련된다.



〈그림 1〉 LLM & Generative AI Relation Diagram

2.1.1. 생성형 AI 동향

초거대AI 기술은 2023년에 들어서 급속한 발전을 거듭하고 있다. <표 1>은 2023년에 출시된 LLM과 생성형 AI 서비스 현황을 보여주고 있으며 3월에 오픈AI는 GPT-4를 발표했으며, 구글은 5월에 선보인 PaLM2 모델에서 파라미터 수는 줄였음에도 불구하고, 이전 모델 대비 약 5배 더 많은 텍스트 데이터인 토큰을 학습하여 실제 성능을 더 높였다. 또한 삼성전자는 11월에 삼성 가우스(Samsung Gauss)를 첫 공개하였으며 향후 출시될 갤럭시 S24등 제품들에 삼성 가우스를 단계적으로 탑재할 계획이라고 밝혔다. 또한 카

카오 등 유수의 기업들도 자체 생성형 AI를 구축 중이거나 검토하고 있으며, 모델 크기보다는 학습량에 초점을 둔 메타(Meta)의 라마(LLaMA) 와 팰컨(Falcon) 등 오픈소스 LLM에 대한 관심이 높아지고 있다. 엔비디아(NVIDIA)는 기업 데이터 센터 등에 AI 모델을 탑재하는 AI팩토리 개념을 추구하고 있으며, 소형화된 LLM(sLLM)을 통해 비용 부담이 적고 기업 내에서 사용하기 쉬운 모델을 개발하고 있다. 특히, 네이버 클라우드는 커머스·금융·법률·교육 등 각 전문분야에 특화된 한국어 중심의 초거대AI인 하이퍼클로바 X(HyperClova X)를 8월에 공개하였으며 하이퍼클로바 X는 도입 기업의 데이터와 해당 도메인에 특화된 형태로 서비스되며, API 방식이나 뉴로 클라우드 방식으로 구축할 수 있다. 이를 통해 LLM 시장은 기존의 성능 경쟁에서 특화 경쟁으로 넘어가는 2라운드에 접어들었다고 볼 수 있다.

특히, 오픈소스 LLM들은 LLaMA를 기반으로 파인튜닝한 형태로 Alpaca가 발표된 이후부터 GPT4All 등 여러 LLM들이 지속적으로 출시되고 있다. GPT4All은 LLaMA 7B 모델을 기반으로 Alpaca에서 영감을 받아 GPT-3.5-Turbo모델의 800,000개의 프롬프트-응답 쌍을 수집하여 코드, 대화 및 내러티브(Narrative)를 포함하여 430,000개의 어시스턴트 스타일 프롬프트 학습 쌍을 만들었으며, 이 쌍은 Alpaca보다 대략 16배 더 크다.

〈표 1〉 2023년 생성형 AI 모델 출시 현황

Country	Company	Foundation Model	Parameters	Source	Release Date	Service
USA	OpenAI / MS	GPT-4 Turbo	미공개	Closed	2023.11	ChatGPT, GPTs / MS Bing AI, MS Copilot, MS 365 Copilot
	Google	Gemini	10 억~1.6 조개	Closed	2023.12	Bard(텍스트, 이미지, 오디오, 비디오)
	META	LLaMA2	70~650 억개	Open	2023.07	Code Llama
	Stanford Univ.	Alpaca	70 억개	Open	2023.03	LLaMA 7B Fine-tuning Model
	Nomic	GPT4All v2	30~130 억개	Open	2023.04	LLaMA 7B Fine-tuning Model
	Hugging Face	BLOOM	1,760 억개	Open	2022.07	-
South Korea	Naver	HyperClova X	미공개	Closed	2023.08	폴라리스오피스 AI, 루이스 등
	LG	EXAONE2.0	3,000 억개	자체활용	2023.07	AI 아티스트 틸다 등
	NC Soft	VARCO	미공개	Closed	2023.08	VARCO Art/Text/Human/Studio
	SKT	A. Enterprise	미공개	자체활용	2023.08	문서요약, 문서생성, Q&A 기능
	KT	MI:DEUM2	2,000 억개	자체활용	2023.10	지니 TV, AICC, AI 통화비서
	SAMSUNG	Samsung Gauss	미공개	자체활용	2023.11	Gauss Language/Code/Image
	Kakao	Ko GPT2.0	60~650 억개	자체활용	2023.4Q	물류, 의료, 금융 등 특화 플랫폼 제공
China	Huawei	PanGu 3.0	1,000 억개	Open	2023.07	Pangu-Weather, etc.
	Baidu	Ernie 3.5	1,300 억개(추정)	자체활용	2023.06	Ernie Bot 3.5
	Alibaba	Tongyi Qianwen	10 조억개(추정)	자체활용 Open	2023.04	DingTalk, Tmall Genie 오픈소스 서비스형: ModelScope

이 모델은 GPU가 필요하지 않고 CPU에서 실행할 수 있는 장점이 있다(최성철, 2023).

이러한 여러 기업의 생성형 AI모델의 출시로 응용 분야도 확대되고 있다. 생성형 AI는 이미 예술, 게임, 엔터테인먼트 등 다양한 분야에서 활용되고 있다. 최근에는 의료 분야에서는 생성형 AI를 사용하여 환자의 병변을 식별하거나, 새로운 약물이나 치료법을 개발하는 데 사용할 수 있으며 제조 분야에서는 생성형 AI를 사용하여 새로운 제품 디자인을 개발하거나, 생산 공정을 개선하는 데 활용할 수 있다. 금융 분야에서는 생성형 AI를 사용하여 새로운 금융 상품을 개발하거나, 금융 거래의 위험을 관리하는 데 활용할 수 있다. 이렇게 생성형 AI는 일상 대화는 물론 금융, 의료, 교육, 엔터테인먼트까지 다방면에서 적용되고 있다(안정희, 박혜옥, 2023).

2.1.2. 생성형 AI 종류 및 응용분야

생성형 AI 기술은 <표 1>에서 제시된 바와 같이 텍스트, 코드, 이미지, 비디오, 3D 모델링, 오디오 등 다양한 형태의 데이터를 생성하는 능력을 지니고 있다. 또한 <표 2>에는 이러한 다양한 데이터 형태별로 대표되는 생성형 AI 모델(Gen AI Models)들과 해당 모델들이 어떠한 응용 분야에서 활용되고 있는지에 대한 정보가 나열되어 있다(정천수, 2023c; Kim, 2023).

텍스트 생성부분에서는 소설, 시, 코드, 이메일, 편지 등 다양한 종류의 텍스트를 생성할 수 있고, 코드 생성영역에서는 웹사이트, 앱, 게임 등 다양한 종류의 코드를 생성할 수 있다. 이미지 생성부분에서는 사진, 그림, 일러스트 등 다양한 종류의 이미지를 생성을 하며, 비디오 생성 영역에서는 영화, TV 프로그램, 광고 등 다양한 종류의 비디오를 생성할 수 있다. 또한 3D 모델링에서는 의류, 가구, 건물 등 다양한 종류의 3D 모델을 생성을 하며, 오디오 영역에서는 노래, 악곡, 멜로디 등 다양한 종류의 음악을 생성할 수 있다. 이렇게 생성형 AI는 다양한 분야에서 활용되고 있다.

〈표 2〉 생성형 AI 대표적 모델 및 응용분야

Type	Gen AI Models	Application Field
Text	OpenAI GPT-4, Google PaLM2, DeepMind Gopher, Meta LLaMA, Hugging Face Bloom, Samsung Gauss	Marketing, Sales, Customer Support, General Writing, Translation, Summarization
Code	OpenAI Codex, Google PaLM2, Ghostwriter, Amazon CodeWhisperer, Tabnine, Stenography, AI2sql, Pygma, Samsung Gauss	Code Generation, Document Code, Generate SQL Queries, Generate Web Apps, Testing, Code formatting
Image	OpenAI Dall-E2, Stable Diffusion, Midjourney, Google Imagen, Meta Make-A-Scene, Samsung Gauss	Generate Image, SNS, Advertising, Design, Data visualization
Video	MS X-CLIP, Meta Make-A-Video, RunwayML, Synthesia, Rephrase AI, Hour One	Video Generation, Video Editing, Video Summarization
3D	DreamFusion, NVIDIA GET3D, MDM	Generate 3D Models, Generate 3D Scenes
Audio	Resemble AI, WellSaid, Play.ht, Coqui, Harmonai, Google MusicLM	Speech Synthesis, Voice Cloning, Generate Music, Sound effect design

2.2. 생성형 AI 주요 요소 이해

2.2.1. 파운데이션 모델(Foundation Model)

생성형 AI 모델은 어떤 출력을 생성하는가에 따라서 언어모델, 이미지 모델, 비디오 모델 등을 사용한다. 하지만 현재는 이미지와 텍스트를 동시에 학습하는 멀티모달(Multi-modal) 모델들이 하루가 다르게 성능과 기능이 업그레이드고 있고 있으며 기초모델(Foundation Model)로 자리잡아 가고 있다. 파운데이션 모델(Foundation Model)은 Bommasani 등 스탠포드대 여러 연구자들이 “파운데이션 모델들의 기회와 리스크”라는 논문을 발표하면서 처음 제안하였다. 파운데이션 모델의 데이터는 텍스트, 이미지, 음성, 정형데이터, 3D 시그널 등 구분하지 않고 학습에 이용되며, 인간의 창의력과 추론력을 포함한 일을 수행하며 이러한 기초모델이 AI 패러다임의 변화와 파운데이션 모델이라고 논하고 있다(Bommasani et al., 2021). 여기에서 방대한 양의 데이터는 비지도 학습(Unsupervised learning)을 통해 모델을 학습시킨 후 배포되어 사용자가 원하는 목적에 맞게 다운스트림(Downstream) 작업에 대해 파인튜닝이나 문맥 내 학습(In-context learning)등과 같은 과정을 거쳐 완성되는 것이 파운데이션 모델이라고 볼 수 있다(Bommasani et al., 2021). 현재 파운데이션 모델은 하나의 출력에만 국한하지 않고 데이터가 어떤 형식이든 처리할 수 있다.

파운데이션 모델은 창발성(Emergence) 특징이 있다. 모델이 스스로 어떠한 문제를 해결하기 위한 지식을 도출하는 능력으로 사전에 프로그래밍 되는 것이 아니라 AI신경망은 데이터만 있으면 알아서 다음 행동을 결정하거나 유추할 수 있다는 점이 창발성의 기본이 된다. 앞으로도 데이터가 점점 많아질수록 창발성의 특징은 더 강하게 작용할 것이다(장민, 안재관, 2023). 또한 균일성(Homogenization) 특징을 갖고 있어 모델이 점차 일반화된 지식을 도출해 낼 수 있게 됨에 따라, 하나의 일관성 있는 데이터만 있다면 거대한 파운데이션 모델이 여러 가지 문제를 풀 수 가 있다(Bommasani et al., 2021).

2.2.2. LLM (Large Language Model)

ChatGPT와 같은 챗봇 서비스에 가장 널리 쓰이고 있는 생성형 AI 모델이 대규모 언어모델(LLM, Large Language Model)로 텍스트와 같은 언어 데이터를 학습하여 결과를 제공하는 생성형 AI 모델로 LLM은 최근 인공지능 분야에서 큰 주목을 받고 있으며, OpenAI의 GPT(Generative Pretrained Transformer) 시리즈 및 Google의 BERT(bidirectional encoder representations from transformers) 등이 대표적인 LLM 모델이며, LLM은 기계 학습과 딥러닝 기술을 기반으로 하며, 훈련에 쓰일 대규모 데이터셋과 높은 계산 능력을 필요로 한다(조정임, 2023). LLM은 NLP(자연어 처리) 및 NLG(자연어 생성) 작업에서 딥 러닝을 활용하는 기반 모델로서, 언어의 복잡성과 연결성을 학습할 수 있도록 돕기 위해 대규모 언어 모델은 방대한 양의 데이터에 대해 사전학습(Pre-training)되며 파인튜닝, In-context learning, Zero/one/few-shot learning 같은 기술을 사용한다(Dilmegani, C., 2023).

다음은 데이터 소스 수집, 전처리, 파인튜닝 방법 등 LLM의 학습 단계를 기술하였다.

- LLM의 학습 단계

1) 데이터 수집 및 전처리: 첫 번째 단계는 LLM이 학습될 리소스인 학습 데이터 세트를 수집하는 것이다. 데이터는 책, 웹사이트, 기사, 공개 데이터세트 등 다양한

소스에서 가져올 수 있다. 유능한 LLM 을 개발하기 위해 사전 학습된 자료로 텍스트 데이터 세트를 사용한다. 사전 학습된 코퍼스의 소스는 크게 일반 데이터와 전문 데이터의 두 가지 유형으로 분류할 수 있으며 웹 페이지, 서적 및 대화 텍스트와 같은 일반 데이터는 크고 다양하며 접근 가능한 특성으로 인해 대부분의 LLM 에서 활용되며 LLM 의 언어 모델링 및 일반화 능력을 향상시킬 수 있다. 또한 다국어 데이터, 과학 데이터 및 코드와 같은 보다 전문화된 데이터 세트로 확장하여 LLM 에 특정 작업 해결 기능을 부여하는 연구도 있다(Taylor et al., 2022; Chowdhery et al., 2022; Nijkamp et al., 2022).

〈그림 2〉 LLM 사전 학습 절차

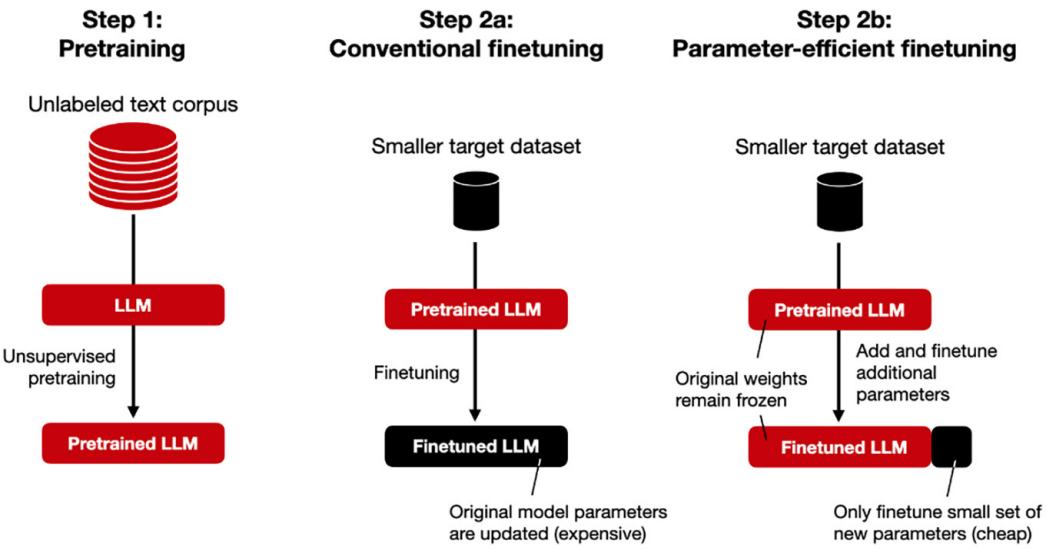
<그림 2>는 일반적인 LLM 의 데이터 수집 및 사전학습 절차를 보여주고 있다(Zhao et al., 2023).

데이터 세트를 찾는 인기 있는 공개 소스는 캐글(Kaggle), Google 데이터 세트 검색, 위키피디아 데이터베이스 등이 있다. 다음 단계로 데이터를 정리하고 학습을 위해 준비해야 한다. 여기에는 데이터 세트를 소문자로 변환하고, 불용어를 제거하고, 텍스트를 구성하는 토큰 시퀀스로 토큰화하는 작업이 포함될 수 있다.

2) 모델 선택 및 구성: Google의 BERT 및 OpenAI의 GPT-3.5와 같은 대형 모델은 모두 최근 몇 년 동안 정교한 NLP 애플리케이션에 일반적으로 선택되는 트랜스포머(Transformer) 딥 러닝 아키텍처를 사용한다. 그리고 트랜스포머 블록의 레이어 수, 어텐션 헤드수(Number of attention heads), 손실 함수(Loss function), 하이퍼파라미터(Hyperparameters) 같은 모델의 일부 핵심 요소를 트랜스포머 신경망 구성할 때 지정해야 한다. 모델구성은 원하는 사용 사례와 교육 데이터에 따라 달라질 수 있으며 모델 학습 시간에 직접적인 영향을 미친다.

3) 모델 학습: 모델은 지도 학습(Supervised learning)을 사용하여 전 처리된 텍스트 데이터에 대해 학습된다. 학습 중에 모델에는 일련의 단어가 제시되고 해당 시퀀스의 다음 단어를 예측하도록 학습된다. 모델은 예측과 실제 다음 단어 간의 차이를 기반으로 가중치를 조정한다. 이 프로세스는 모델이 만족스러운 성능 수준에 도달할 때까지 수백만 번 반복된다. 모델과 데이터의 크기가 크기 때문에 모델을 학습하려면 엄청난 계산 능력이 필요하며 학습 시간을 줄이기 위해 모델 병렬화라는 기술이 사용된다. 이렇게 대규모 언어 모델을 처음부터 학습하려면 상당한 투자가 필요하다. 따라서 보다 경제적인 대안은 기존 언어 모델을 특정 사용 사례에 맞게 파인튜닝 하는 것이다.

4) 평가 및 파인튜닝: 학습 후에는 모델 성능을 측정하기 위한 학습 데이터 세트로 사용되지 않은 테스트 데이터 세트에서 모델을 평가한다. 평가 결과에 따라 모델의 성능을 향상시키기 위해 하이퍼파라미터를 조정하거나, 아키텍처를 변경하거나, 추가 데이터에 대한 교육을 통해 일부 파인튜닝이 필요할 수 있다.



〈그림 3〉 LLM의 Fine-tuning 단계

LLM은 대규모 데이터 모음에 대해 훈련되고 일반적인 지식을 가지고 있으나 파인튜닝 없이는 특정 작업에서 제대로 수행되지 않을 수 있기 때문에 <그림 3>과 같이 파인튜닝 단계를 거쳐 모델의 성능을 향상시키는데, 이때 모든 매개변수를 튜닝하는 것이 아닌 사전 훈련된 LLM(Pre-trained LLM)에 소수의 새로운 매개변수를 추가하고 추가된 매개변수만 파인튜닝하여 적은 비용으로 더 나은 성능을 발휘하도록 하는 PEFT(Parameter-Efficient Fine-Tuning)방법을 주로 사용한다(Raschka, S., 2023). 특히, 사전 학습 데이터가 LLM의 성능에 크게 영향을 미치기 때문에 소규모 언어 모델과 비교하여 LLM은 모델 사전 학습을 위한 고품질 데이터에 대한 수요가 더욱 필요하며, 모델 용량은 사전 학습을 위한 말뭉치(Corpus) 데이터 수집과 사전학습 처리 방법에 크게 의존하게 된다.

2.2.3. 프롬프트 엔지니어링 (Prompt Engineering)

ChatGPT에서 대화 시에는 질문이나 요청인 프롬프트(Prompt)를 얼마나 자세하게 전달하느냐에 따라 완성도 높은 답변을 얻을 수 있기 때문에 LLM으로부터 프롬프트 입력 값들의 조합을 찾는 작업을 탐구하는 프롬프트 엔지니어링도 중요한 요소로 작용한다(정천수, 2023c). 대량의 데이터로 모델의 파라미터를 업데이트하는 파인튜닝 과정 없이도 답변의 성능을 높이는 방법으로 프롬프트 질문 시 답변 예시를 제공함으로써 그와 유사한 답변을 생성할 수 있도록 유도하는 방법이다(정천수, 2023c). 프롬프트의 지시문에 따라, 포함시키는 예시가 없는 경우인 Zero-Shot, 예시가 1개인 One-shot, 2개 이상인 Few-shot Learning으로 구분되며, 다양한 예시를 제공할수록 일정 수준이상의 답변을 얻을 수 있다(정천수, 2023c). <표 3>은 프롬프트 방법별 예시를 보여주고 있다(삼성SDS, 2023a).

〈표 3〉 Zero/One-shot/Few-shot Learning

방 법	내 용
	Prompt : Write a short alliterative sentence about a curious cat exploring a garden
Zero-shot learning	[Prompt에 포함시킨 예시 정보] - [ChatGPT 답변] A cat looks at flowers in the garden
One-shot learning	[Prompt에 포함시킨 예시 정보] Peter Piper picked a peck of pickled peppers. [ChatGPT 답변] Curious cat cautiously checking colorful cabbages.
Few-shot learning	[Prompt에 포함시킨 예시 정보] Example 1: Peter Piper picked a peck of pickled peppers. Example 2: She sells seashells by the seashore. Example 3: How can a clam cram in a clean cream can? [ChatGPT 답변] Curious cat crept cautiously, contemplating captivating, colorful carnations

또한 구조, 기능, 복잡성에 따라 7가지 유형으로 분류될 수 있다(Sivarajkumar et al., 2023).

1) Prefix Prompt는 제어 및 관련성을 위해 응답 유형/형식/어조를 나타내는 단어나 문구를 앞에 추가하는 가장 간단한 유형의 프롬프트이다.

2) Cloze Prompt는 입력 텍스트에 마스크된 토큰을 생성하고 언어 모델에 누락된 단어나 구문을 예측하도록 요청하는 빈칸 채우기 연습 아이디어를 기반으로 한다.

3) Anticipatory Prompt는 경험이나 지식을 바탕으로 다음 질문/명령을 예상하고 대화를 안내하는 프롬프트이다.

4) CoT Prompt는 LLM에 지시하는 과정 속에서 모델이 연쇄적으로 사고하며 정보를 연결하고 정밀한 답변을 내놓도록 유도하는 프롬프트 엔지니어링 기법으로, 복잡한 문제에 대한 명확한 해답을 추론하기 위해 적용된다.

5) Ensemble Prompt는 집계된 출력에 대해 다수 투표를 사용하여 여러 프롬프트를 결합하는 프롬프트로 다양한 유형의 프롬프트를 사용하여 동일한 입력에 대해 여러 출력을 생성한 다음 가장 일반적인 출력을 최종 답변으로 선택한다.

6) Heuristic Prompt는 포괄적인 답변을 위해 복잡한 쿼리를 더 작은 부분으로 나누는 규칙 기반 프롬프트이다.

7) Few-shot Prompt는 2개 이상의 예시를 제공하여 모델이 더 나은 성능을 발휘하도록 유도하는 문맥 내 학습을 가능하게 하는 기술로 사용할 수 있다.

이와 같이 프롬프트 엔지니어링은 LLM을 사용하여 창의적인 콘텐츠를 생성하는 방법이다. LLM은 방대한 양의 텍스트와 코드 데이터 세트에서 훈련된 AI 로서 LLM은 텍스트를 생성하고, 언어를 번역하고, 다양한 종류의 창의적인 콘텐츠를 작성하고, 질문에 답변할 수 있다(오영환, 2023). 또한 프롬프트 엔지니어링은 LLM의 창의적인 텍스트 생성 능력을 향상시키기 위해 LLM에 제공되는 프롬프트의 품질을 개선하는 데 중점을

둔다. 프롬프트는 LLM에 제공되는 지시 사항과 지침으로 프롬프트는 간단하거나 복잡할 수 있으며, LLM이 생성해야 하는 텍스트의 형식, 내용, 스타일을 지정할 수 있다. 이렇게 함으로써 LLM의 창의적인 텍스트 생성 능력을 크게 높일 수 있어, 인공 지능 모델이 새로운 정보나 작업을 습득하고 예측하는 방법을 다양하게 나타내며 프롬프트 엔지니어링을 통해 모델의 학습 및 추론 능력을 향상시킬 수 있다.

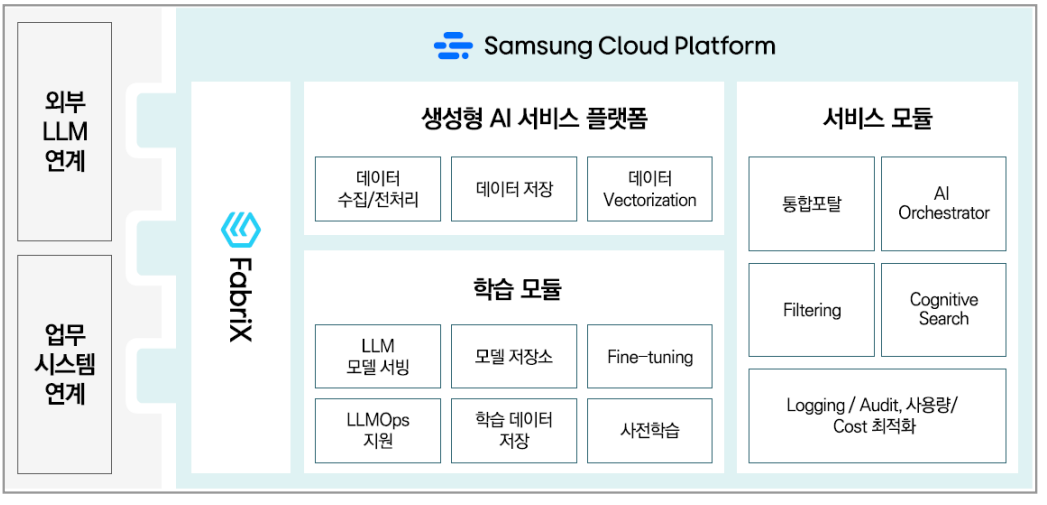
2.2.4. 생성형 AI 기술 스택

수많은 기업들이 생성형 AI 도입을 검토하거나 사용을 고민하고 있으나, LLM을 자체 보유하거나 외부 LLM을 사용한다고 하더라도 지원하는 인프라가 뒷받침되지 못하면 LLM은 무용지물이 될 수 있다.

따라서, 기업들은 LLM을 쉽게 적용하고 운영 관리할 수 있는 <그림 4>와 같은 생성형 AI 서비스 플랫폼을 도입하게 된다(삼성SDS, 2023b).

일반적인 생성형 AI 서비스 기술 구성 계층은 어플리케이션, LLM 허브, 인프라, 오케스트레이션 계층으로 분류할 수 있으며 각 계층은 다양한 공급업체의 솔루션을 포함 할 수 있다(Andreessen Horowitz, 2023; Jeong C.S., 2023). 또한 새롭게 떠오르는 생성형 AI 스택에서 기업은 자체 모델을 개발하거나, API를 통해 타사 생성 AI를 호출하거나, 필요에 맞게 미세 조정된 오픈 소스 모델을 활용하여 애플리케이션을 구축하고 있다.

- 애플리케이션 계층: Apps은 사용자가 자체 모델 파이프라인(“E2E Apps”)을 실행하거나 타사 API를 사용하여 생성형 AI 모델을 사용한다(예: Jasper 및 Copilot).
- 모델 계층: 파운데이션 모델에는 비공개 소스 독점 모델(예: GPT-4), 오픈 소스 모델(예: Stable Diffusion)이 있고 파운데이션 모델을 공유하고 호스팅하는 모델 허브가 있다.
- 오케스트레이션 및 LLMops 계층: 생성형 AI 스택에는 LLM 모델의 선택 및 배포,



〈그림 4〉 생성형 AI 서비스 개념도 예시(SAMSUNG Gen AI Platform)

모니터링을 하기 위한 기능을 포함한다(예; LangChain, Prompts 관리 등).

- 인프라 계층: 생성형 AI 모델에 대한 학습 및 추론 워크로드를 실행하는 클라우드 플랫폼 및 H/W 계층이다(예: AWS, Azure 등 클라우드 플랫폼 및 GPU/TPU등 하드웨어).

2.3. RAG 모델 이해

LLM을 사용하는 ChatGPT와 같은 같은 대화형 AI 모델의 발전은 다양한 분야에서 큰 관심을 받고 있으며 LLM을 활용한 애플리케이션 개발이 주목받고 있다. 예를 들어 금융 분야에서는 고객 응대 챗봇을 통해 최신 정보 제공이 중요한데, LLM의 정보 부족으로 인해 제한된 답변 능력을 보인다는 문제점이 있고 환각 문제로 모델이 자신이 실제로는 모르는 정보를 적당히 지어내어 이야기하는 현상을 나타낸다. 이러한 문제를 극복하기 위한 전략으로는 LLM을 새로운 데이터로 파인튜닝하는 방법과 정보를 직접 프롬프트 콘텍스트에 넣어주는 방법이 제시되었다. 그러나 전자의 경우 상당한 비용이 발생하며, 후자의 경우 모든 정보를 프롬프트에 넣어주는 것은 현실적으로 어렵다. 이에 대안으로 RAG모델이 제안되었으며, 이 모델은 정보를 데이터베이스에 저장하고, 필요한 정보를 검색하여 LLM에 전달하는 방식으로 구현된다. LLM에게 미리 질문과 관련된 참고자료를 제공하여 사용하는 방식이기 때문에 모델은 참고자료를 활용하여 보다 정확하고 신뢰성 있는 답변을 생성하게 된다. 본 절에서는 본 논문에서 구현 방법으로 사용하고 있는 RAG모델에 대해 자세히 알아보고자 한다.

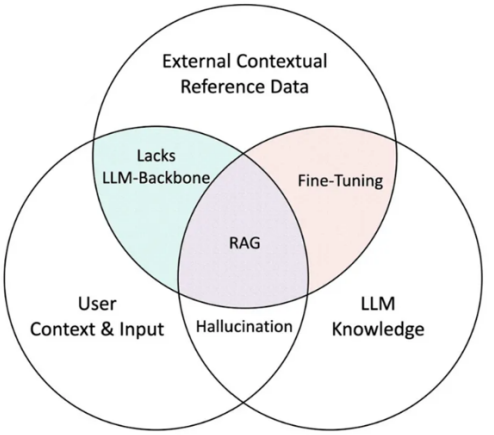
대화형 AI, 특히 LLM을 활용한 ChatGPT와 같은 모델들은 다양한 산업 분야에서 활용되며 크게 주목받고 있다. 특히 금융 분야에서는 고객 응대 챗봇을 통한 최신 정보 제공의 중요성에 대두되며, LLM의 정보 제한성과 환각 문제는 이러한 모델들의 도전 과제로 지적되고 있다. 이를 해결하기 위한 접근 방식으로는 새로운 데이터로의 파인튜닝과 프롬프트 콘텍스트에 직접 정보를 삽입하는 방안이 있으나, 파인튜닝의 경우에 상당한 비용발생 하며 모든 정보를 프롬프트에 넣어주는 것도 현실적으로 어렵다. 이에 대안으로 RAG모델이 제안되었으며, 이 모델은 정보를 데이터베이스에 저장하고, 필요한 정보를 검색하여 LLM에 전달하는 방식으로 구현된다. LLM에게 미리 질문과 관련된 참고자료를 제공하여 사용하는 방식이기 때문에 모델은 참고자료를 활용하여 보다 정확하고 신뢰성 있는 답변을 생성하게 된다. 본 절에서는 RAG 모델의 구현과 그 세부 사항을 더 심층적으로 다루고자 한다.

2.3.1. RAG 모델 아키텍처

RAG 모델은 텍스트 생성 작업을 수행하는 모델로 주어진 소스 데이터로부터 정보를 검색하고, 해당 정보를 활용하여 원하는 텍스트를 생성하는 과정을 수행한다. RAG 사용을 위한 데이터 처리 과정은 원본 데이터를 청크(Chunk)단위의 작은 조각으로 나누고 텍스트 데이터를 숫자인 벡터로 전환하는 임베딩(Embedding)을 거쳐 벡터 저장소에 저장된다(Microsoft, 2023).

- 소스 데이터(Source Data) 수집 및 준비: 모델의 학습 및 활용을 위해 관련된 소스 데이터가 필요하며 문서, 웹 페이지, 뉴스 기사 등이 소스 데이터가 될 수 있으며, 이 데이터는 모델이 정보를 검색하고 생성할 내용에 대한 기반이 된다.

- 검색 가능한 청크 생성(Chunking): 소스 데이터를 처리하여 작은 단위인 청크로 분할한다. 청크는 보통 문장 또는 문단과 같은 작은 텍스트 조각을 의미하며, 이 작은 청크 단위로 정보를 검색하고 활용하기 용이 해진다.
- 임베딩(Embedding): 생성된 청크는 텍스트를 벡터 형태로 변환하는 과정을 거친다. 주로 사전 학습된 언어 모델을 사용하여 텍스트를 의미 있는 밀집 벡터로 변환하고, 이렇게 함으로써 텍스트의 의미와 관련 정보가 벡터로 포착된다.
- 벡터 데이터베이스 (Vector Database) 구축: 임베딩된 청크들을 기반으로 벡터 데이터베이스를 구축한다. 이 데이터베이스는 벡터 공간에서 각 청크의 위치를 나타내며, 검색 및 유사성 계산을 효율적으로 수행할 수 있게 해준다.
- 검색 및 정보 통합: 생성할 텍스트의 컨텍스트에 맞는 정보를 검색하기 위해 벡터 데이터베이스에서 적절한 청크를 검색한다. 이 때, 검색된 청크는 원래 텍스트 데이터로 디코딩하여 정보를 추출하고 이 정보는 생성 과정에서 활용된다.
- 텍스트 생성 (Generation): 검색된 정보를 기반으로 텍스트를 생성한다. 이 때, 생성할 텍스트의 종류나 길이, 언어적인 스타일 등을 지정할 수 있으며, RAG 모델은 정보 검색과 생성 과정을 통합하여 더 정확하고 의미 있는 텍스트를 생성할 수 있도록 설계되었다.



〈그림 5〉 RAG & Fine-Tuning

RAG 모델 사용은 검색과 생성을 결합하여 보다 정확하고 의미 있는 텍스트 생성을 가능케 하는 혁신적인 방법 중 하나이다. 이 아키텍처는 정보 검색의 장점과 텍스트 생성의 창의성을 결합하여 다양한 응용 분야에서 활용될 수 있다. 또한 최근 RAG 모델은 파인튜닝과 경쟁해 왔으며 <그림 5>는 파인튜닝과 RAG 간의 중복을 보여주고 있으며, 기업에서는 RAG와 파인튜닝의 조합을 하는 것이 필요하다(Greyling, 2023).

2.3.2. LLM에서 임베딩 활용

임베딩 또는 임베딩 벡터(Embedding Vector)는 텍스트를 고정 크기의 floating point 숫자들로 구성되어 있는 고정된 크기의 배열 같은 실수(floating) 벡터 형태로 표현하는 것을 의미한다. 특정 단어, 문장 또는 문서를 임베딩 생성 모델에 입력하면, 실수로 이루어진 벡터가 출력된다. 이러한 벡터는 사람이 직접 이해하기 어렵지만, 서로 다른 단어나 문서의 임베딩 간 거리를 계산하면 의미적 관계를 파악할 수 있다. 요즘에는 대규모 문서 집합을 활용하여 신경망 구조를 갖는 LLM

같은 언어 모델을 학습하여 “Learned Embedding” 생성 모델을 만드는 것이 일반적이다. 이 학습 과정에서 모델은 다양한 단어들을 입력으로 받아들이게 되며, 문맥 속에서 의미적으로 유사한 단어들의 임베딩 벡터 간 거리를 가깝게, 그렇지 않은 단어들의 임베딩 벡터 간 거리를 멀게 만드는 방식으로 의미적인 관계를 학습한다(Jeong C.S., 2023).

LLM에 활용하기 위한 임베딩 은 서로 다른 단어 또는 문서들 간의 의미적 관계를 파악할 수 있는 아래 2가지 방법을 활용할 수가 있다.

- 시멘틱 검색(Semantic Search): 키워드 매칭이 아닌 문장의 의미에 초점을 두어 검색 사용자의 인지적 노력을 최소화하면서 사용자 질의의 문맥을 이해하여 의미에 맞는 문서를 정확히 찾아주는 기술이다(홍기주 등, 2016). 이것은 여러 문서 중에서 의미를 기반으로 하나의 문서를 검색 또는 서로 비교할 때 사용할 수 있으며 임베딩을 활용한 시멘틱 검색은 다음과 같은 과정을 거친다.

  1) 문서 임베딩 계산: 검색 대상인 문서 모음집에 있는 각 문서에 대한 임베딩을 계산한다. 이러한 임베딩은 문서의 의미적 특성을 수치적으로 표현하며, 별도의 저장소에 보관되며 이 저장소는 로컬 드라이브, 벡터 데이터베이스 등의 형태로 구성될 수 있다.

  2) 검색어 임베딩 계산: 사용자가 입력한 검색어에 대한 임베딩을 계산한다. 이 이것은 사용자의 검색 의도를 수학적으로 표현한 것이며, 이를 통해 문서와의 의미적 유사성을 측정한다.

  3) 코사인 유사도(Cosine similarity) 계산 및 정렬: 검색어 임베딩 과 각 문서 임베딩 간의 코사인 유사도를 계산한다. 코사인 유사도는 두 벡터 사이의 각도를 기반으로 하며, 높은 유사도는 의미적으로 관련된 문서를 나타내며 이 유사도를 기준으로 모든 문서를 내림차순으로 정렬한다.

  4) 상위 k 개 문서 선택 및 반환: 정렬 결과 중에서 가장 높은 유사도를 갖는 상위 k 개의 문서를 선택하고, 문서들의 텍스트 내용을 불러와 의미적으로 연관성이 높은 정보를 검색 결과로 제시한다.

- QA(Question Answering): LLM이 결과를 생성하기 위한 부가적인 정보를 추가하는 방법으로 QA는 LLM의 중요한 응용 프로그램으로 의료, 교육 및 고객 서비스 전반에 걸쳐 챗봇 기능을 형성한다. 그러나 광범위한 LLM 통합은 LLM API 사용에 드는 높은 비용으로 인해 중소기업에 어려움을 안겨주며 도메인별 LLM 응답을 위한 쿼리와 함께 도메인별 데이터(컨텍스트)를 사용하면 비용이 급격히 증가하기 때문에 도메인별 QA 시스템에서는 질문이 포함된 컨텍스트가 프롬프트로 LLM에 제공되어 답변을 제공받는다. QA 시스템은 다음과 같은 두 단계로 나눌 수 있다(Arefeen et al., 2023).

  1) 도메인 데이터 수집: 이 단계에서는 문서가 텍스트 분할기(Text splitter)에 의해 여러 개의 고정 청크(Fixed chunks)로 분할된다. 임베딩 함수는 임베딩 생성기를 사용하여 각 청크의 임베딩을 계산한다. 각 청크의 임베딩 벡터(Embedding vector)와 함께 청크는 벡터 데이터베이스(Vector database)에 저장된다.

2) 질문 답변(QA) 단계: 사용자 쿼리가 주어지면 시멘틱 검색(의미기반 검색)을 사용하는 쿼리 임베딩과 유사한 임베딩을 통해 유사한 N개의 청크가 검색된다. 이러한 청크는 컨텍스트를 형성하고, 마지막으로 도메인 데이터의 하위 집합인 컨텍스트가 LLM 에 입력되어 답변을 얻는다.

2.3.3. **벡터 데이터베이스**(Vector Database)

벡터 데이터베이스는 LLM의 장기 기억 부족 문제를 해결하기 위해 개발된 새로운 유형의 데이터베이스이다. 이 데이터베이스는 고차원 실수 벡터 인덱스를 효율적으로 저장하고 관리하는데 특화되어 있으며 벡터 데이터베이스는 전통적인 데이터베이스와 다르게 쿼리를 실수 벡터(Embedding) 형태로 표현하며, 이와 유사한 벡터를 가진 데이터를 추출하는 방식을 지원한다. 따라서 AI 모델의 결과로 얻은 벡터를 저장하고 활용하는데 최적화되어 있다. 벡터 데이터베이스에 대한 일반적인 벡터 파이프라인은 아래 3단계를 거친다(Devtorium, 2023).

1) Indexing: 벡터 데이터베이스는 PQ, LSH 또는 HNSW 와 같은 알고리즘을 사용하여 벡터를 인덱싱한다. 이 단계는 더 빠른 검색을 가능하게 하는 데이터 구조에 벡터를 매핑한다.

2) Querying: 벡터 데이터베이스는 가장 가까운 근접 데이터을 찾기 위해 인덱싱된 쿼리 벡터를 데이터 세트의 인덱싱된 벡터와 비교한다. 이때 해당 인덱스에서 사용하는 유사성 메트릭을 적용한다.

3) Post Processing: 경우에 따라 벡터 데이터베이스는 데이터 세트에서 가장 가까운 최종 근접 데이터를 검색하고 사후 처리하여 최종 결과를 반환한다. 이 단계에는 다른 유사성 측정을 사용하여 가장 가까운 근접 데이터의 순위를 다시 매기는 작업이 포함될 수 있다.

이렇듯 LLM의 데이터베이스는 RDB가 아닌 벡터 DB를 사용해야 한다. Claypot AI 창립자인 칩 후옌(Chip Huyen)이 ‘그래프 DB의 해가 2021년이라면, 벡터 DB의 해는 2023년’이라고 말한 것처럼 최근 이 분야에 대한 관심이 높다(Huyen, C., 2023). Chroma, FAISS(Facebook AI Similarity Search) 같은 독립형 벡터 인덱스도 벡터 임베딩 검색을 개선하지만, DB의 관리 기능이 부족하다. 반면에 벡터 데이터베이스는 데이터 관리, 메타 데이터 저장 등 벡터 데이터베이스는 확장성 문제, 번거로운 통합 프로세스, 실시간 업데이트 및 내장된 보안 조치의 부재와 같은 독립형 벡터 인덱스의 한계를 해결함으로써 벡터 임베딩을 처리하기 위한 우수한 솔루션을 제공한다. Pinecone은 안정적인 클라우드 호스팅 기반으로 서비스되며, Weaviate, Vespa, Qdrant는 단일 노드 기반의 오픈소스 DB이다. 그리고 Chroma, FAISS는 로컬 벡터 관리 라이브러리로, 복제(Replication)와 샤딩(Sharding)을 통한 성능 보장, 내결함성 향상, 모니터링, 접근 제어, 백업과 컬렉션 등 DB의 주된 기능을 제공하지 않기 때문에 엄밀히 DB는 아니지만 LangChain 및 OpenAI 임베딩 모델과 같은 다른 AI 관련 도구와 쉽게 연계할 수 있어 임베딩 검색 용도로 가볍게 사용할 수 있다.

2.3.4. LLM **서비스 구현을 위한 오케스트레이션 프레임워크**(Orchestration Framework)

LLM 서비스를 구현하기 위한 오케스트레이션

프레임워크로는 대표적으로 Semantic Kernel과 LangChain이 있으며 두 프레임워크 모두 LLM의 기능을 효율적으로 활용하고 사용자에게 편리한 사용 경험을 제공하는 것을 목표로 한다(Jeong C.S., 2023).

- Semantic Kernel 프레임워크: Semantic Kernel은 Microsoft에서 개발한 오픈 소스 SDK로, 개발자가 기존 애플리케이션에 LLM을 쉽게 통합할 수 있도록 지원한다. 또한 C#, Python, Java 등 다양한 언어를 지원하며, Azure OpenAI, OpenAI, Hugging Face 등 다양한 LLM 서비스와 연계된다. Semantic Kernel은 자연어 처리, 기계 학습, 계획 등의 AI 기술을 사용하여 애플리케이션에 다음과 같은 기능을 추가할 수 있다.

  - 자연어 인터페이스: 사용자가 자연어로 애플리케이션과 상호 작용할 수 있음
  - 텍스트 생성: 텍스트, 코드, 스크립트, 음악 작품, 이메일, 편지 등 다양한 종류의 창의적인 텍스트 콘텐츠를 생성
  - 질문에 대한 답변: 사용자의 질문에 유익하고 포괄적인 답변을 제공
  - 작업 자동화: 애플리케이션이 사용자의 지시에 따라 작업을 자동으로 완료.

- LangChain 프레임워크: LangChain은 생성형 AI의 언어 모델을 활용하여 애플리케이션을 개발할 수 있도록 지원하는 오픈 소스 프레임워크이다. LangChain은 Model I/O, Retrieval, Chains, Agents, Memory, Callbacks 모듈로 구성되어 있으며, LLM을 사용하여 다양한 언어 처리 작업을 수행할 수 있다(LangChain, 2023). OpenAI, Hugging Face 등 다양한 LLM 모델을 지원하며, 사용자는 자신의 요구에 맞는 모델을 선택하여 사용할 수 있다. LangChain의 핵심적인 개념은 LLM 프롬프트의 실행과 외부 소스의 실행(계산기, 구글 검색, 슬랙 메시지 전송이나 소스코드 실행 등)을 엮어 Chaining하는 것으로 LLM을 사용하여 번역, 요약, 질문 답변, 텍스트 생성, 자연어 추론 등 다음과 같은 다양한 작업을 수행할 수 있다.

  - 번역 서비스: 번역 모델을 배포하고 관리할 수 있고 다양한 언어를 지원하는 번역 모델을 제공
  - 요약 서비스: 요약 모델을 배포하고 관리할 수 있어 다양한 주제에 대한 요약 모델을 제공
  - 질문 답변 서비스: 질문 답변 모델을 배포하고 관리할 수 있고 다양한 분야에 대한 질문 답변 모델을 제공
  - 텍스트 생성 서비스: 텍스트 생성 모델을 배포하고 관리할 수 있고 다양한 형식의 텍스트를 생성할 수 있는 모델을 제공
  - 자연어 추론 서비스: 자연어 추론 모델을 배포하고 관리할 수 있으며 다양한 자연어 추론 작업을 수행할 수 있는 모델을 제공

Semantic Kernel과 LangChain은 모두 LLM 서비스를 구현하기 위한 강력한 프레임워크로 사용자는 구현하고자 하는 요구 사항에 따라 적절한 프레임워크를 선택하여 사용할 수 있다.

2.3.5. AI Assistant

AI Assistant는 ChatGPT같은 AI Chatbot을 통해 생성형 AI 서비스를 활용할 수 있는데 챗봇을

통해 원하는 질문에 대한 프롬프트를 입력할 수 있으며 답변을 받을 수가 있다. 오늘날 인공지능 기반의 많은 챗봇 시스템들은 단순문의 응답형인 FAQ, 복잡하고 다양한 규칙기반 시나리오뿐만 아니라 사람의 감정과 의도를 분석해 답변을 제시하는 수준에까지 이르고 있다(정천수, 2023a). 최근들어 챗봇은 NLU(Natural Language Understanding) 기술의 발전으로 Context 모델과 Transformer 언어모델 활용으로 복잡한 대화 처리가 가능하다. 또한 STT(Speech to Text)와 TTS(Text to Speech) 기술 성숙으로 음성 서비스도 대중화되어가고 있다(정천수, 2023a).

챗봇은 사람과 서비스 봇 간에 문자나 음성을 통해 질문에 알맞은 답이나 각종 연관 정보를 제공하는 인공지능 기반의 대화형 소프트웨어이다(한국정보화진흥원, 2016). Sánchez-Díaz, Ayala-Bastidas, Fonseca-Ortiz & Garrido 연구에서는 챗봇을 사용자가 일반적으로 텍스트 또는 음성을 통해 대화를 할 수 있는 지능형 에이전트라고 정의하였다(Sánchez-Díaz et al., 2018; 정천수, 정지환, 2020). 챗봇은 사람의 질문을 인식하고 사람이 말하는 것처럼 자연스럽게 답변을 제공하기 위해 NLP 기술을 이용한다(정천수, 2023b).

NLP는 질문인 자연어를 읽고 이해하는 NLU (Natural Language Understanding)와 답변을 위해 자연어를 생성하는 NLG(Natural Language Generation) 컴포넌트로 구성되어 있다(정천수, 2023b). NLG는 ML을 사용하여 대규모 말뭉치(Corpus)를 학습하여 답변을 생성할 수 있는데 2020년에 OpenAI가 출시한 NLP모델인 GPT-3(Generative Pre-trained Transformer-3)는 3,000억개의 데이터셋을 학습에 사용했고 매개변수(Parameter)는 1,750억개였으며 2021년에 Microsoft와 NVIDIA가 공개한 MT-NLG (Megatron-Turing Natural Language Generation) 모델의 매개변수는 5,300억개나 되는 대형언어모델(LLM, Large Language Model)을 개발하여 자동 대화 생성, 번역 등에 활용할 수 있게 됐다(정천수, 2023b) 또한 2022년 11월에는 OpenAI에서 GPT-3에 인간 전문가 집단이 피드백(RLHF, Reinforcement Learning from Human Feedback)시키는 학습과정을 거치면서 대화형으로 발전시킨 GPT-3.5모델을 적용하여 사람과 유사한 자연어를 생성하도록 학습된 AI 챗봇인 ChatGPT를 공개하여 출시 2개월만에 월간 이용자가 1억명을 넘기며 많은 관심을 받고 있다(정천수, 2023b). 챗봇 도입 초기에는 챗봇에 대한 신뢰성 저하로 업무에 직접 적용하지 않고, 주로 단순한 질문에 답을 해주는 상담 기능 등에 주로 활용되었지만 최근에는 챗봇에 RPA 및 OCR 등 타 솔루션과 연계하여 챗봇을 업무에 직접적으로 활용하여 효율성을 높이고 있다(정천수, 정지환, 2020). 또한 챗봇은 자동화 플랫폼과 연계하여 트리거 또는 자동화 업무 처리의 채널 역할로 진행상태 및 처리결과안내 등에 활용할 수 있다(정천수, 2023b). 특히, ChatGPT같은 오픈된 도메인에 대한 챗봇 대화에서는 ChatGPT와 LLM모델과의 Prompt로 인터페이스를 하게 되는데, 이때 인터페이스를 하기 위한 어플리케이션 사이에 사용하는 매개변수를 적용할 때 주의해야 할 정보는 개인정보에 관한 것은 회피해야 한다. 개발자가 동의 없이 사용자로부터 데이터를 수집하고 사용하는 것을 방지하는 법이 이미 있지만, 실제 생활에서 사용자는 개발자가 데이터를 얼마나 많이 가져오고, 해당 데이터가 어디에 있는지 알기 어렵기 때문이다(Jeong, J. H. and Jeong, C. S., 2022).

지금까지 2장에서는 생성형 AI와 관련된 여러 개념과 기술들을 살펴보았다. 그러나 이러한 다양한 기술들을 어떻게 조합하고 관리하여 실제

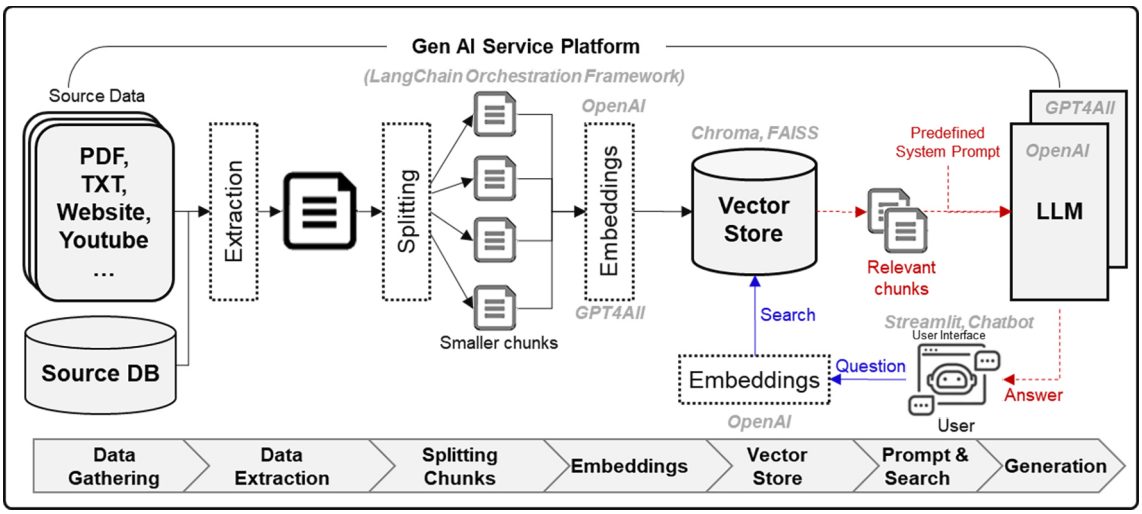
업무에 LLM 서비스로써 적용할 수 있는지, 이에 대한 구현 방법과 절차를 찾아보기 힘들다. 또한 다양한 생성형 AI 기술 스택의 여러 기술을 효과적이고 종합적으로 조율할 수 있는 프레임워크 관점에서 보안 문제를 해결하기 위한 내부 업무 정보 활용에 대한 체계적인 연구 자료는 더욱 찾아보기가 힘들다. 이러한 한계를 극복하기 위해 본 연구에서는 생성형 AI 관련 문헌 연구의 한계와 실제 업무에 적용하기에 부족한 부분을 고려하여, LLM 서비스를 구현하기 위한 프레임워크를 활용하여 생성형 AI 기술을 통합하여 적용할 수 있는 구현 방법 및 구현 화면까지 체계적으로 제시하고자 한다.

3. 생성형 AI 활용 서비스 구현 방법

본 장에서는 앞서 서술했던 생성형 AI 기술 스택의 여러 기술을 조합하여 효과적이고 종합적으로 조율할 수 있는 생성형 AI서비스 플랫폼(Gen AI Service Platform)의 전체 구현 프레임워크를 구성하고 기술 요소별로 구현 방법에 대하여 제시한다.

3.1. RAG 모델 활용 생성형 AI 서비스 구현 플랫폼

본 연구에서 제안하는 생성형 AI 서비스 플랫폼은 선행 연구를 기반으로 하여 전체적인 구현 단계에서 필요한 핵심 요소 기능을 포함하고 있다. 이는 <그림 6>에서 상세히 개념화하고 있으며, 플랫폼을 오케스트레이션하는 프레임워크를 통해 소스 데이터로부터의 정보 추출 및 LLM의 활용 절차까지 나타내고 있다. 과정은 문서에서 정보를 추출하여 소규모 청크로 분할하고, 이를 임베딩하여 벡터 데이터베이스에 저장하는 것으로 시작한다. 사용자의 질문이 임베딩된 후, 질문과 유사한 벡터를 저장소에서 검색하여 LLM에 전달되고, LLM은 이러한 청크들을 조합하여 사용자 정의 데이터로부터 응답을 생성해 제공한다.



〈그림 6〉 RAG기반 생성형 AI 서비스 구현 플랫폼

3.2. RAG 모델 및 LangChain 통합 구현 절차

생성형 AI 서비스 구현 플랫폼의 각 단계별 구축 기술요소에는 많은 솔루션들이 나와 있다. 이러한 솔루션 중 본 연구에서 구성을 제안하는 플랫폼의 프레임워크 구성 기술 요소는 2장의 선행연구를 바탕으로 각 영역 별로 인지도와 비용 측면을 고려하여 기본적으로 오픈소스 기반의 제품을 선정을 하였고, LLM은 생성형 AI의 촉발을 가져온 OpenAI 모델과 오픈소스 모델을 같이 사용하여 <그림 6>과 같이 구성하였다. 전체 오케스트레이션 프레임워크로는 LangChain을 사용하였으며 청크 작업 및 임베딩은 OpenAI와 GPT4All모델을 사용하였다. 벡터 저장소는 쉽게 적용할 수 있는 Chroma DB를 사용하였으며, LLM은 OpenAI의 GPT-3.5-turbo모델과GPT4All을 사용하여 구현하고자 하는 사용자가 다양한 선택을 통한 최상의 개발을 할 수 있도록 하였다. 또한 사용자 인터페이스는 간단하게 구현하여 배포할 수 있는 Streamlit과 Brity Assistant를 사용하여 구현했다.

3.2.1. RAG기반 구현 절차

RAG 모델은 2장 RAG 모델 아키텍처에서 살펴본 바와 같이 주어진 질문이나 주제에 관련된 정보를 검색하여 가져오고, 이를 기반으로 응답을 생성하는 데 활용되는 검색 증강 생성 모델이다. 각 단계는 <그림 6>과 같은 절차로 진행된다.

1) 소스 데이터 수집(Source Data Gathering) 및 추출(Data Extraction) 단계

데이터 수집 단계에서는 정형 데이터와 비정형 데이터를 모두 수집하는 단계이다. 데이터는 파일형태로 존재하거나 DB에 저장한 형태로 존재할 수 있다. 정형 데이터는 CSV, JSON, XML 등과 같은 표준 포맷으로 저장된 데이터로, 비교적 수집이 쉬우나, 비정형 데이터는 PDF, TXT, HTML, 이미지, 비디오 등과 같은 형식으로 저장된 데이터로, 정형 데이터에 비해 수집이 어렵다. 업무에 관련 자료 규정집, 사용자 매뉴얼, 약관 등 사전에 자료를 준비하고 LLM에 활용할 자료를 LangChain 모듈을 이용하여 로드(Loading)한다.

2) 청크 생성 (Splitting Chunks) 단계

소스 데이터를 처리하여 작은 단위인 청크로 분할하는 단계이다. 청크는 보통 문장 또는 문단과 같은 작은 텍스트 조각으로 분할되며 LLM에서 검색 가능한 단위로 정보를 검색하고 활용하기 용이 하게 LangChain 모듈을 이용하여 분할한다.

3) 임베딩 (Embeddings) 단계

생성된 청크 단위 텍스트 데이터를 수치화된 벡터 형태로 변환하는 과정을 거친다. 이 단계에서는 단어나 문장을 벡터로 매핑하는 작업이 이루어지며 OpenAI 또는 GPT4All에서 제공하는 라이브러리를 이용할 수 있다.

4) 벡터 데이터베이스 (Vector Database) 구축 단계

임베딩된 청크들을 기반으로 벡터 데이터베이스를 구축하는 단계이다. 이 데이터베이스는 벡터 공간에서 각 청크의 위치를 나타내며, 검색 및 유사성 계산을 효율적으로 수행할 수 있도록 한다. 일반적으로 각 문서의 콘텐츠를 포함시킨 다음 임베딩과 문서를 벡터 저장소에 저장하고 임베딩을 사용하여 문서를 인덱싱하며, Chroma나 FAISS같은 벡터 인덱스를 활용할 수 있다.

5) 질문(User Prompt)과 검색(Search)결과 통합 단계

프롬프트로 질문한 내용으로 검색하고 관련 정보를 통합하는 단계이다. 생성할 텍스트의 컨텍스트에 맞는 정보를 검색하기 위해 벡터 데이터베이스에서 적절한 청크를 검색된 관련 청크들을 LLM에 보내어 답변 생성 과정에서 활용될 수 있도록 하며, LangChain에 있는 벡터스토어 유사성 검색하는 많은 검색기를 활용한다.

6) 답변 생성 (Generation) 단계

검색된 정보를 기반으로 답변 텍스트를 생성하는 단계이다. 이 때, 생성할 텍스트의 종류나 길이, 언어적인 스타일 등을 지정할 수 있으며, OpenAI 또는 GPT4All 등 LLM에서 LangChain의 유사내용 검색 모듈을 통해서 검색된 문서에서 LLM이 답변을 생성하여 제공한다.

7) 사용자에게 채널을 통해 답변제공

LLM에서 생성된 답변은 Streamlit 같은 UI 생성 라이브러리로 구현된 화면이나 Brity Assistant 같은 챗봇 메신저를 통해 최종 사용자에게 제공한다.

3.3. RAG 기반 Vector Store선정 및 답변 제공유형 정의

3.3.1. RAG기반 Vector Store 유형 선정 및 처리 절차

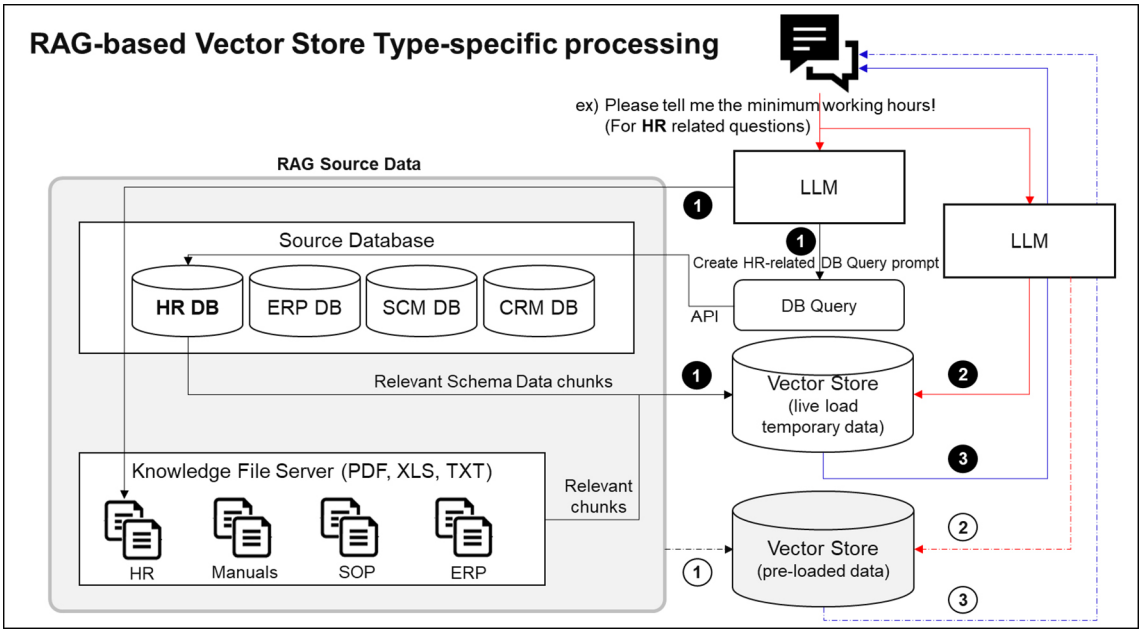
RAG기반 Vector Store 저장 유형은 <그림 7>과 같이 모든 소스 데이터를 사전에 Vector Store에 저장하는 경우와 질문 시 실시간으로 넣는 경우로 나눌 수 있다. 처리절차는 아래와 같다.

① 실시간 소스데이터 임시 저장 형태는 질문이 들어왔을 때, 데이터 베이스에서 Query하는 경우에는 기업에 있는 여러 개의 소스 데이터베이스에서 질문과 관련된 데이터베이스를 조회하는 DB Query prompt를 생성한 후 API를 통해 관련된 데이터 베이스에서 모든 스키마를 가져오거나, DB형태가 아닌 File형태의 지식은 파일을 Upload하여 최적의 많은 내용을 가져와서 실시간으로 임베딩한 후 Vector Store에 저장한다 (예; 직원의 최소근무시간 알려줘! → 소스 DB중에서 인사(HR) DB 선택하여 DB Query 저장 또는 '인사정보.pdf' File Upload 저장). 또 다른 형태인 사전 전체 소스데이터 저장 형태는 사전에 정의한 대상 범위내의 모든 소스 데이터베이스의 내용이나 Vector화 할 지식 File을 모두 임베딩하여 Vector Store에 저장한다.

② 실시간으로 저장된 Vector Store에서 질문과 관련된 내용을 찾아와서 질문과 함께 프롬프트를 이용하여 LLM으로 전달한다.

③ 질문에 해당하는 유사한 답변을 여러 개 찾아서 질문과 함께 LLM에 전달하여 최적의 최종 답변을 생성한다.

기업에서는 내부지식을 Open LLM을 통해 서비스하게 되었을 때 보안이슈 때문에 Local LLM을 사용하는 것이 중요한 이슈이다. 이때 각 구간별로 가장 잘 처리할 수 있는 Local LLM을 여러 개 구성하여 사용하는 것이 효과적인데, <그림 7>에서 여러 개의 소스 데이터베이스에서 질문에 맞는 데이터베이스의 내용을 가져오는 DB Query를 생성하는데 특화된 LLM이나, 특정지식에서 답변을 생성해줄 수 있는 도메인 특화된 LLM을 별도 구축할 수도 있다. 임베딩 하는 방법에서는 사전 전체 소스데이터 저장 형태를 선택할 때는 향후,



〈그림 7〉 RAG기반 Vector Store 구성유형 및 처리절차

더 나은 임베딩 모듈이 나와 교체할 경우에 많은 노력이 필요하기 때문에 임베딩 할 데이터의 양이나 범위를 신중히 따져서 구축하는 게 필요하다.

3.3.2. RAG기반 답변 제공 유형

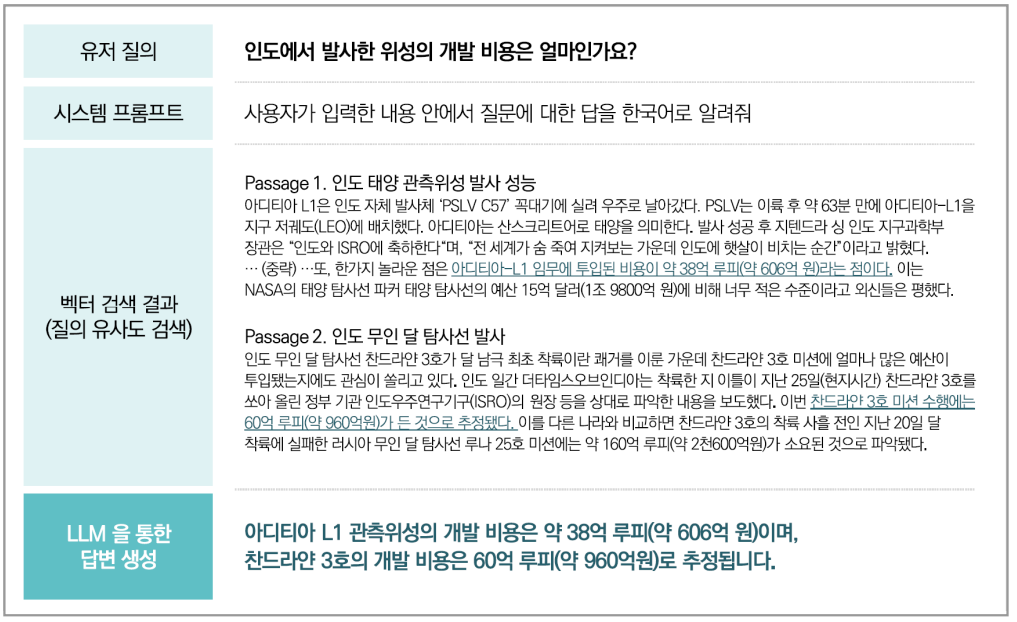
RAG 모델을 활용하여 답변을 제공하는 방법으로는 QA시스템을 활용하여 유사도 기반의 벡터 검색과 같은 알고리즘을 활용하여 답변을 내는 방법과 QA시스템이 없을 경우에는 직접 전체 벡터화된 지식에서 답변을 제공하는 방법을 활용할 수가 있다. 전체 지식을 벡터화 하는데 드는 비용 및 효과를 따져 구축하는 것이 필요하다. <그림 8>은 QA시스템에서 검색한 두가지 유사 Passage를 질문과 같이 프롬프트를 넣어 LLM에 전달하여 최종 한 개 답변을 전달받을 수 있는 것을 보여주고 있다(삼성SDS, 2023b).

4. 구현 결과

본 장에서는 3장에서 소개한 생성형 AI 활용 서비스 구현 플랫폼을 기반으로 기업 내부에서 활용할 만한 데이터를 활용하여 RAG 모델 및 LangChain 프레임워크의 통합 구현 절차에 따라 각 구현 단계별로 다양한 사례를 구현하였다. 이를 통해 구현하는 방법 및 구축 시 고려해야 할 사항에 대하여 구현 사례를 통해 제시한다.

4.1. 개발 환경

본 사례에 적용하는 솔루션 및 개발 플랫폼의 프레임워크는 <그림 6>에서 제시하는 것을 기본으로 적용하여 LangChain 및 OpenAI API를 활용하여 문서를 변환하는 구현 방법을 제시하고자 한다. 문서를 덩어리로 나누고, 임베딩으로 변환



〈그림 8〉 QA시스템을 활용한 LLM 지식답변 예시

하고, ChromaDB에 저장하는 단계를 다루며 이 접근 방식을 사용하면 컨텍스트 기반(Context-based) 답변을 효과적으로 제공하는데 필요한 정보에 액세스할 수 있도록 하였으며, 개발 언어는 Python 활용하여 서비스를 구현하였다. Python은 AI 개발에 필요한 다양한 라이브러리와 프레임워크가 있어 AI 개발에 가장 많이 사용되는 언어이기 때문에 AI 개발에 적합한 개발 언어이다. 또한 QA 시스템을 활용하기 위하여 별도 설치 없이 바로 구현이 가능한 PaaS로 제공되는 Brity Assistant로 구현하였다.

각 구현 요소별 개발환경은 아래와 같다.

- 오케스트레이션 프레임워크: LangChain
- 데이터 추출 및 Chunking: LangChain 모듈
- 임베딩: OpenAI, GPT4All
- 벡터 데이터베이스: Chroma, FAISS
- LLM: OpenAI GPT-3.5-tubo모델, GPT4ALL
- Python 개발 환경: Google Colab
- QA시스템: Brity Assistant MRQA 모듈
- UI: Streamlit (Python apps library)

4.2. 단계별 구현 결과

4.2.1. 기본 라이브러리 설치 및 OpenAI API Key 설정

기본적으로 데이터 분할 등 개발 전반적인 오케스트레이션 프레임워크 역할을 하는 LangChain과 OpenAI 라이브러리를 설치한다. 또한 Colab에서 Google드라이브를 연동해서 지식파일을 쉽게 관리할 수 있게 하였다. 개발환경에서 유일하게 오픈소스가 아닌 OpenAI는 별도로 Key를 발급받아 구현 시 사용할 수 있도록 설정해야 한다.

```
1 !pip install -q openai  # openai 라이브러리를 설치.
2 !pip install -q langchain # 랭체인 라이브러리를 설치
3 !pip install -q python-dotenv  # .env file load 라이브러리를 설치
```

```
1 # Google 드라이브 연동
2 from google.colab import drive
3 drive.mount('/content/drive')
```

```
1 cd /content/drive/MyDrive/GenAI_paper/data
```

```
/content/drive/MyDrive/GenAI_paper/data
```

```
1 import os
2 os.listdir()
```

```
['지급보험금계산.txt',
 '회사생활가이드(QA형).csv',
 '근무 복장 기준.docx',
 '.env',
 '휴직 및 복직 관리기준.pdf']
```

```
1 from dotenv import load_dotenv # Add
2 load_dotenv() # load .env
```

```
True
```

〈그림 9〉 기본 라이브러리 설치 및 OpenAI API Key 설정

특히, Key값은 보안상 오픈 되지 않게 하기 위해 특정 위치에 .env 환경파일 형태로 입력해서 관리하는 것을 권장하고 개발소스에서는 Key 값이 들어 있는 파일 내용을 읽어서 인증을 받도록 구현한다. <그림 9>는 환경변수 관리용인 dotenv 라이브러리를 설치하고 Key값이 들어 있는 .env 화일을 정상적으로 읽어 들인 것을 확인할 수 있다.

4.2.2. 소스 데이터 수집(Source Data Gathering) 및 추출(Data Extraction) 단계 구현

소스 데이터에는 비정형 데이터로 MS-Office 계열의 docx, xlsx, cvs, pptx 등 다양한 문서에 들어 있거나 또다른 형태의 txt, pdf 같은 파일로 존재한다. 또한 웹페이지나 유튜브 같은 동영상으로도 존재할 수 가 있다. 다양한 구조화되지 않은 원시 데이터에서 정보를 추출하기 위해서는 먼저 데이터를 로드 해야 한다. <표 4>는 기업 내부 정보로 관리되는 문서 중 MS-Office 계열문서중 대표적으로 doc문서로 '근무 복장 기준.docx' 문서와 '지급보험금계산.txt', '휴직 및 복직관리기준.pdf', '회사생활가이드(QA형).csv' 같은 비정형 문서화일 정보를 <그림 9>와 같이 특정 위치에 저장해 놓고 로드 하는 문서 타입별 구현 코드이다.

〈표 4〉 소스 데이터 타입

Doc Type	Python Code
DOC	from LangChain.document_loaders.word_document import UnstructuredWordDocumentLoader loader = UnstructuredWordDocumentLoader("근무 복장 기준.docx")
TXT	from LangChain.document_loaders import TextLoader loader = TextLoader("지급보험금계산.txt")
PDF	from LangChain.document_loaders import PyPDFLoader loader = PyPDFLoader("휴직 및 복직 관리기준.pdf")
CSV	from LangChain.document_loaders.csv_loader import CSVLoader loader = CSVLoader("회사생활가이드(QA형).csv")

또한 <그림 10>은 비정형 문서 중에서 대표적으로 PDF문서를 로드 하는 부분의 구현된 코드를 보여주고 있다. 나머지 유형의 문서들도 <표 4>와 같이 로드 할 수 있도록 구현하였다.

4.2.3. 청크 생성 (Splitting Chunks) 및 임베딩 단계 구현

다음 부분은 다시 LangChain 통합 기능을 사용하여 큰 원시 소스 문서를 임베딩 및 벡터 저

```
!pip install -q pypdf  # Library to extract text from pdf document
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("휴직 및 복직 관리기준.pdf")
pages = loader.load_and_split()
data = loader.load()
print(f"{len(data)} documents, contain {len(data[0].page_content)} words")
```

```
1 documents, contain 1021 words
```

〈그림 10〉 소스 데이터 수집 및 추출

```
### Splitting Chunks
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size = 300,
                                               chunk_overlap = 0)
all_splits = text_splitter.split_documents(data)

### Embeddings & Vector stores
!pip install chromadb # 벡터스토어
!pip install tiktoken # 토큰 계산용
import tiktoken   #tiktoken is a fast BPE tokeniser for use with OpenAI's models.

from langchain.llms import OpenAI
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Chroma

vectorstore = Chroma.from_documents(documents=all_splits, embedding=OpenAIEmbeddings())
```

〈그림 11〉 청크 분할 및 임베딩

장을 위해 여러 개의 작은 청크 단위로 분할하는 작업을 한다. <그림 11>은 문서가 최대 300자('chunk_size = 300')의 작은 덩어리(Chunk)로 분할되도록 구현한 코드이다. 다음으로 분할된 청크를 임베딩하여 벡터저장소에 저장한다. 먼저 저장하기 위한 벡터 데이터베이스인 Chroma를 설치한다. 청크는 OpenAI 임베딩 모듈을 사용하여 숫자 벡터로 변환되고 이러한 방식으로 생성된 각 벡터는 나중에 검색할 수 있도록 ChromaDB에 저장되도록 <그림 11>과 같이 구현한다. 기본적으로 오픈소스인 ChromaDB 저장소는 메모리에 있으며 세션이 다시 시작되면 내용이 지속되지 않는다.

4.2.4. 질문(User Prompt)과 검색(Search)결과 통합 단계 구현

유사성 검색을 사용하여 모든 질문에 대한 관련된 분할정보를 검색하는 단계이다. 사용자가 쿼리를 보내면 먼저 Chroma DB에서 가장 관련성이 높은 청크를 찾기 위해 이를 임베딩으로 변환해야 한다. LangChain을 사용하여 이 작업을 수행하는 방법을 구현한 코드는 <그림 12>와 같다. 사용자가 "휴직 기간에 대하여 알려줘"라고 질문을 하면 '휴직 및 복직관리기준.pdf' 문서가 저장된 벡터데이터 베이스에서 질문과 유사한 내용을 찾게 된다. RetrievalQAWithSourcesChain 개체는 질문에 답하고 답이 검색된 소스 문서를 반환할

```
question = "휴직 기간에 대하여 알려줘"

docs = vectorstore.similarity_search(question)
len(docs)
print(f"{len(data)}개의 문서에 {len(data[0].page_content)}개의 단어를 가지고 있습니다")
```

```
1개의 문서에 1021개의 단어를 가지고 있습니다
```

```
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQAWithSourcesChain

retriever = vectorstore.as_retriever(search_kwargs={"k": 1})
llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)
chain = RetrievalQAWithSourcesChain.from_chain_type(
    llm=llm,chain_type="stuff",retriever = retriever,return_source_documents=True)

result = chain(question)
result
```

```
{'question': '휴직 기간에 대하여 알려줘',
 'answer': '휴직기간은 휴직 사유에 따라 다양한 기간 내에서 필요에 따라 부여될 수 있습니다. 육아휴직의 경우 최대 2년이며, 진학의 경우 국내진학은 최대 2년, 해외진학은 학위 기간 + 출입국 기간을 포함하여 최대 2.5년입니다. 어학의 경우 최대 1년입니다.₩n',
 'sources': '휴직 및 복직 관리기준.pdf',
 'source_documents': [Document(page_content='가족돌봄이 필요하여 1개월 이상 정상적인 직무를 ₩n 수행할 수 없다고 인정될 경우 ₩n사사육아 - 육아로 인해1개월 이상 정상적인 직무를 수행할 수 ₩n 없다고 인정되는 경우 ₩n기타 - 가족돌봄을 제외한 기타 개인사정 (건강관리 , 임신기 , ₩n 육아等 )으로1개월 이상 정상적인 직무를 수행할 수 ₩n 없다고 인정될 경우 ₩n3. 휴직기간 ₩n휴직기간은 휴직 사유별로 다음과 같은 기간 내에서 필요에 따라 부여할 수 있음 ₩n(1) 휴직 기간 ₩n₩uf06c 육아휴직 : 최대2年, 1회 분할 사용가능 (상세내용은 육아휴직 기준 참조) ₩n₩uf06c 진학: 국내진학은 학위 기간 내 최대2년, 해외진학은 학위 기간 + 출입국 ₩n기간(0.5년)내 출입국 기간을 포함하여 최대2.5년 ₩n₩uf06c 어학: 최대1년(국내 최소3개월 이상, 해외 최소6개월 이상)',
metadata={'page': 0, 'source': '휴직 및 복직 관리기준.pdf'})]}
```

〈그림 12〉 질문(User Prompt)과 검색(Search)결과

수 있는 체인으로 RetrievalQAWithSourcesChain 개체의 chain() 메서드를 호출하여 쿼리에 대한 답을 반환하게 된다. 이 검색 체인을 통해 1개의 문맥을 추천하도록 설정(search_kwargs={"k": 1})하여 질문에 대한 답변이 될만한 유사한 1개 문맥이 나타난 것을 볼 수 있다.

4.2.5. 프롬프트 활용 및 답변 생성 (Generation) 단계 구현

Vector Store인 Chroma DB에서 의미상 사용자 질문에 가장 가까운 관련된 청크를 추출하고, 이것은 올바른 맥락으로 사용자의 질문에 답변할 수 있도록 LLM에 제공된다. 관련 청크가 식별되면 이를 컨텍스트로 사용하여 LLM에서 응답을 생성한다. 이때 시스템 프롬프트를 사용하여 정확한 답변을 하도록 구성하여 하나의 완성된 자연스러운 답변을 할 수 있도록 <그림 13>과 같이 구현하였다. “휴직 기간에 대하여 알려줘” 질문에 대한 답변을 제공한 것을 보여주고 있으며 추가적으로 “육아 휴직 기간 알려줘” 질문했을 때도 <그림 14>의 ‘휴직 및 복직관리기준.pdf’ 문서 내용이 들어가 있는 벡터 데이터베이스에서 잘 찾아서 답변을 생성해주었다. 또한 시스템 프롬프트에서 ‘답을 모른다면 “모른다”고 말하고 답을 만들어 내려고 하지 말것’이라고 정의함에 따라서 없는 정보에 대한 질문인 “근무복장 규정 알려줘”라고 질문했을 때 답을 모른다고 답변을 생성해주는 것을 확인할 수 있었다.

```
from langchain.prompts.chat import (ChatPromptTemplate,SystemMessagePromptTemplate,HumanMessagePromptTemplate,)
system_template="""다음 내용을 참조하여 사용자의 질문에 간단히 답변할것.
문서와 질문에 대한 요약(summaries)이 주어지면 참조("SOURCES")를 사용하여 최종 답변을 작성할것.
답을 모른다면 "모른다"고 말하고 답을 만들어 내려고 하지 말것.
----------
{summaries}

한국어와 마크다운 형식으로 답변할것:"""

messages = [
    SystemMessagePromptTemplate.from_template(system_template),
    HumanMessagePromptTemplate.from_template("{question}") ]
prompt = ChatPromptTemplate.from_messages(messages)
```

```
from langchain.chains import RetrievalQAWithSourcesChain
chain_type_kwargs = {"prompt": prompt}
chain = RetrievalQAWithSourcesChain.from_chain_type(
    llm=llm, chain_type="stuff", retriever = retriever, return_source_documents=True,chain_type_kwargs=chain_type_kwargs)
```

```
question = "휴직 기간에 대하여 알려줘"
result = chain(question)
result['answer']
```

```
'휴직기간은 휴직 사유에 따라 다음과 같은 기간 내에서 필요에 따라 부여됩니다.₩n₩n1. 육아휴직: 최대 2년, 1회 분할 사용 가능합니다. 상세 내용은 육아휴직 기준을 참조하시면 됩니다.₩n₩n2. 진학: 국내진학은 학위 기간 내 최대 2년, 해외진학은 학위 기간 + 출입국 기간(0.5년)을 포함하여 최대 2.5년까지 가능합니다.₩n₩n3. 어학: 최대 1년까지 가능하며, 국내는 최소 3개월 이상, 해외는 최소 6개월 이상이 필요합니다.₩n₩n이상입니다. 추가로 궁금한 사항이 있으시면 말씀해주세요.'
```

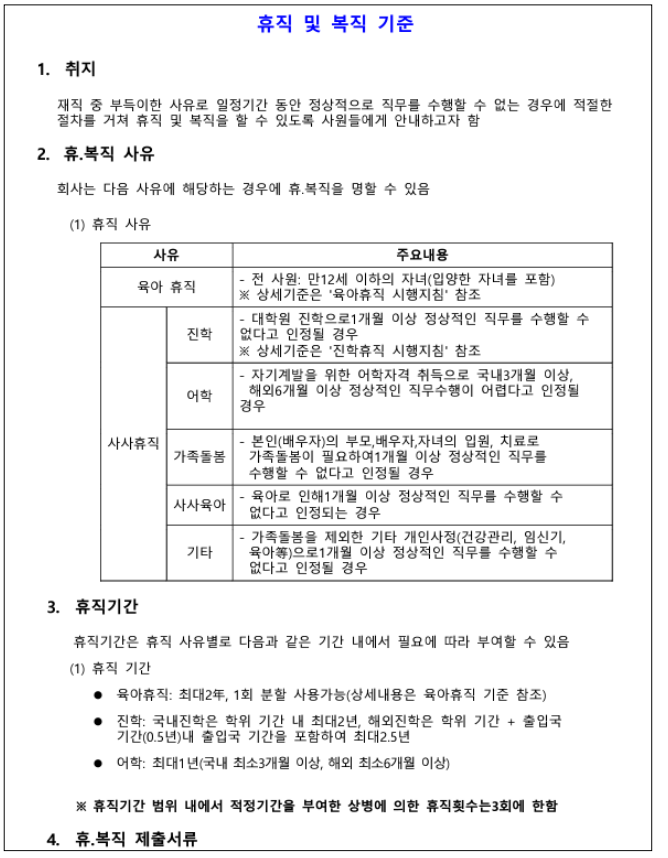
```
question = "육아 휴직 기간 알려줘!"
result = chain(question)
result['answer']
```

```
'육아휴직은 최대 2년까지 가능하며, 1회 분할 사용이 가능합니다. 자세한 내용은 "육아휴직 기준"을 참조해주세요.'
```

```
question = "근무복장 규정 알려줘!"
result = chain(question)
result['answer']
```

```
'근무복장 규정에 대한 내용은 제공되지 않았습니다. 죄송합니다.'
```

〈그림 13〉 LLM을 활용한 답변 생성



휴직 및 복직 기준

1. 취지

재직 중 부득이한 사유로 일정기간 동안 정상적으로 직무를 수행할 수 없는 경우에 적절한 절차를 거쳐 휴직 및 복직을 할 수 있도록 사원들에게 안내하고자 함

2. 휴.복직 사유

회사는 다음 사유에 해당하는 경우에 휴.복직을 명할 수 있음

(1) 휴직 사유

사유		주요내용
육아 휴직		- 전 사원: 만12세 이하의 자녀(입양한 자녀를 포함) ※ 상세기준은 '육아휴직 시행지침' 참조
사사휴직	진학	- 대학원 진학으로1개월 이상 정상적인 직무를 수행할 수 없다고 인정될 경우 ※ 상세기준은 '진학휴직 시행지침' 참조
	어학	- 자기계발을 위한 어학자격 취득으로 국내3개월 이상, 해외6개월 이상 정상적인 직무수행이 어렵다고 인정될 경우
	가족돌봄	- 본인(배우자)의 부모,배우자,자녀의 입원, 치료로 가족돌봄이 필요하여1개월 이상 정상적인 직무를 수행할 수 없다고 인정될 경우
	사사육아	- 육아로 인해1개월 이상 정상적인 직무를 수행할 수 없다고 인정되는 경우
	기타	- 가족돌봄을 제외한 기타 개인사정(건강관리, 임신기, 육아等)으로1개월 이상 정상적인 직무를 수행할 수 없다고 인정될 경우

3. 휴직기간

휴직기간은 휴직 사유별로 다음과 같은 기간 내에서 필요에 따라 부여할 수 있음

(1) 휴직 기간

- 육아휴직: 최대2年, 1회 분할 사용가능(상세내용은 육아휴직 기준 참조)
- 진학: 국내진학은 학위 기간 내 최대2년, 해외진학은 학위 기간 + 출입국 기간(0.5년)내 출입국 기간을 포함하여 최대2.5년
- 어학: 최대1년(국내 최소3개월 이상, 해외 최소6개월 이상)

※ 휴직기간 범위 내에서 적정기간을 부여한 상병에 의한 휴직횟수는3회에 한함

4. 휴.복직 제출서류

〈그림 14〉 '휴직 및 복직관리기준.pdf' 내용

4.2.6. 다양한 소스 데이터 처리 구현

앞에서는 PDF 등 텍스트 문서 파일에 대한 것을 대표적으로 선정하여 각 단계별로 구현된 코드에 대하여 기술을 하였고 다음은 문서 이외의 소스 데이터 정보로 Youtube동영상에 대한 정보와 웹 베이스 정보를 처리하는 코드를 오픈소스로 제공되고 있는 LLM과 또 다른 Vector DB인 FAISS로 구현을 하여 다양한 사례를 제공하고자 하였다.

1) Youtube 데이터 처리

실제 업무에서 기업에 관련된 홍보 자료나 교육자료 등을 Youtube에 올려놓고 공유하는 경우가 많은데 이때 활용될 수 있도록 Youtube에 올라가 있는 정보를 찾거나 찾은 Youtube내용에 대한 질문 시 답변을 받을 수 있도록 오픈소스 기반 벡터 데이터베이스인 FAISS를 활용하여 <그림 15>와 같이 구현을 하였다.

2) 웹 베이스 데이터 처리

업무 관련된 홈페이지나 제품 설명 웹페이지, 회사 포탈 등 웹페이지에 있는 소스 데이터를 활용할 경우에 이용할 수 있도록, 오픈소스 기반

```
!pip install -q openai tiktoken langchain
!pip install -q youtube-transcript-api youtube_search #Youtube data
!pip install -q faiss-cpu #Vector DB

from langchain.llms import OpenAI
from langchain.document_loaders import YoutubeLoader #Youtube Loader
from langchain.tools import YouTubeSearchTool # YouTube Search
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS #Vector store
from langchain.chains import RetrievalQA
import tiktoken   #tiktoken is a fast BPE tokeniser for use with OpenAI's models.

embeddings = OpenAIEmbeddings()
loader = YoutubeLoader.from_youtube_url("https://www.youtube.com/watch?v=jDpyUuspaBg")

documents = loader.load()
llm = OpenAI(model_name="gpt-3.5-turbo", temperature=0)

db = FAISS.from_documents(documents, embeddings)
retriever = db.as_retriever()
qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    return_source_documents=True)
# tool.run("son seungmin") #YouTube Search --> Result : ['/watch?v=I_OLjYhTc64&pp=ygUMc29uIHNlZ5nbWlu', '/shor
query = "손흥민은 몇살에 조국을 떠났으며, 잠은 몇시에 자는지 알려주고, 그가 생각하는 성공에 대한 의견 요약해줘?
result = qa({"query" : query})

print("Question : " + query)
print("Answer : " + result['result'])
```

```
Question : 손흥민은 몇살에 조국을 떠났으며, 잠은 몇시에 자는지 알려주고, 그가 생각하는 성공에 대한 의견 요약해줘?
Answer : 손흥민은 15살 때 조국을 떠났으며, 그는 9~10시 사이에 잠을 자는 것을 좋아한다고 말했습니다. 그는 성공은 언제
```

〈그림 15〉 Youtube 데이터처리(VectorDB - FAISS)

```
!pip install gpt4all  #GPT4All  LLM
!pip install langchain
!pip install chromadb

from langchain.document_loaders import WebBaseLoader  #WebBase URL Data Loader
loader = WebBaseLoader("https://tjdans0721.tistory.com/25") #Web Contents
data = loader.load()

from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=0)
all_splits = text_splitter.split_documents(data)
```

```
from langchain.vectorstores import Chroma
from langchain.embeddings import GPT4AllEmbeddings  #GPT4All Embedding

vectorstore = Chroma.from_documents(documents=all_splits, embedding=GPT4AllEmbeddings())
```

```
100%|██████████| 45.9M/45.9M [00:00<00:00, 87.9MiB/s]
```

```
question = "달의 나이는 어떻게 되나요?"
docs = vectorstore.similarity_search(question)
len(docs)
docs[0]
```

```
Document(page_content='지구와 달 사이의 관계는 수천 년 동안 인간들에게 무한한 호기심을 자아내고, 언제나 새로운 비밀을 풀어가게 합니다. 그동안, 달의 정확한 나이는 미스터리로 남아 있었지만, 최근에 발표된 연구결과는 이 미스터리의 한 조각을 해결해 주었습니다.₩n₩xa0₩n우리의 달은 과연 몇 살일까요? 이 질문은 화성 크기의 물체가 지구와 충돌하여 달이 형성된 지 얼마나 오래됐는지에 대한 질문과 연결되어 있습니다. 이러한 과정은 우리 태양계의 형성과 진화에 관한 중요한 정보를 제공하는 중요한 퍼즐 조각입니다.', metadata={'description': '지구와 달 사이의 관계는 수천 년 동안 인간들에게 무한한 호기심을 자아내고, 언제나 새로운 비밀을 풀어가게 합니다. 그동안, 달의 정확한 나이는 미스터리로 남아 있었지만, 최근에 발표된 연구결과는 이 미스터리의 한 조각을 해결해 주었습니다. 우리의 달은 과연 몇 살일까요? 이 질문은 화성 크기의 물체가 지구와 충돌하여 달이 형성된 지 얼마나 오래됐는지에 대한 질문과 연결되어 있습니다. 이러한 과정은 우리 태양계의 형성과 진화에 관한 중요한 정보를 제공하는 중요한 퍼즐 조각입니다. 연구진은 이러한 나이 미스터리를 해결하기 위해 달의 마그마 바다에서 형성된 결정을 분석했습니다. 이 연구는 스코틀랜드 글래스고대학과 미국 캘리포니아 로스앤젤레스대(UCLA), 노스웨스턴대, 시카고필즈박물관, 시카고대의 연구진으...', 'language': 'ko', 'source': 'https://tjdans0721.tistory.com/25',
'title': '"달의 나이가 44억6000만 년? 새로운 연구결과로 밝혀진 미스터리"'})
```

〈그림 16〉 웹 페이지 데이터처리(LLM - GPT4All)

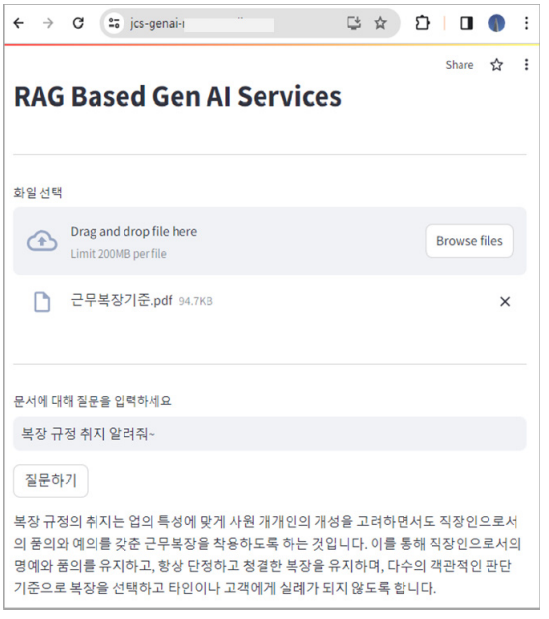
```python
!pip install streamlit
import streamlit as st
import tempfile

st.title("RAG Based Gen AI Services") # 제목
st.write("---")

uploaded_file = st.file_uploader("화일 선택") # 파일 업로드
st.write("---")

pages = pdf_to_document(uploaded_file)

question = st.text_input('문서에 대해 질문을 입력하세요')  # Question

if st.button('질문하기'):
    with st.spinner('Wait for it...'):
        llm = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)
        qa_chain = RetrievalQA.from_chain_type(llm,retriever=db.as_retriever())
        result = qa_chain({"query": question})
        st.write(result["result"])

```

〈그림 17〉 UI 구현 (Streamlit)

LLM인 GPT4All을 사용하였다. LLaMA 7B 모델을 기반 모델로 GPU가 필요하지 않고 CPU에서 실행할 수 있는 장점 있다. <그림 16>은 GPT4All 이용하여 임베딩과 LLM을 활용한 부분으로 웹 페이지의 소스 데이터를 벡터 DB에 넣어서 처리한 것을 보여주고 있다.

4.2.7. 사용자 인터페이스(User Interface) 구현 화면

질문과 답변 채널을 위한 사용자 인터페이스를 구현한 소스의 일부는 <그림 17> 이고, 화면은 <그림 18>과 같다. RAG기반인 지식정보 파일(PDF, TXT, DOCX 등)을 화면에서 선택하고 선택된 지식 내에서 질문에 대한 답을 찾아 답변하는 것을 볼 수 있다.



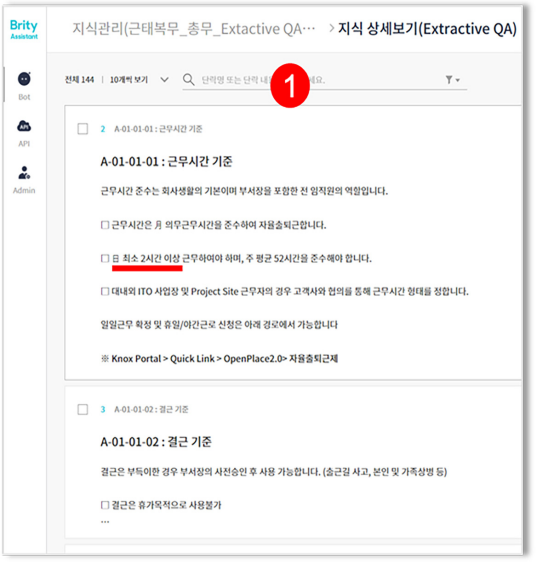
〈그림 18〉 UI 구현화면

QA시스템을 활용한 구현 사례는 <그림 19>, <그림 20>과 같으며, 최종 답변을 LLM을 통해 제공받고 추가로 유사답변도 볼 수 있도록 구현하였다.

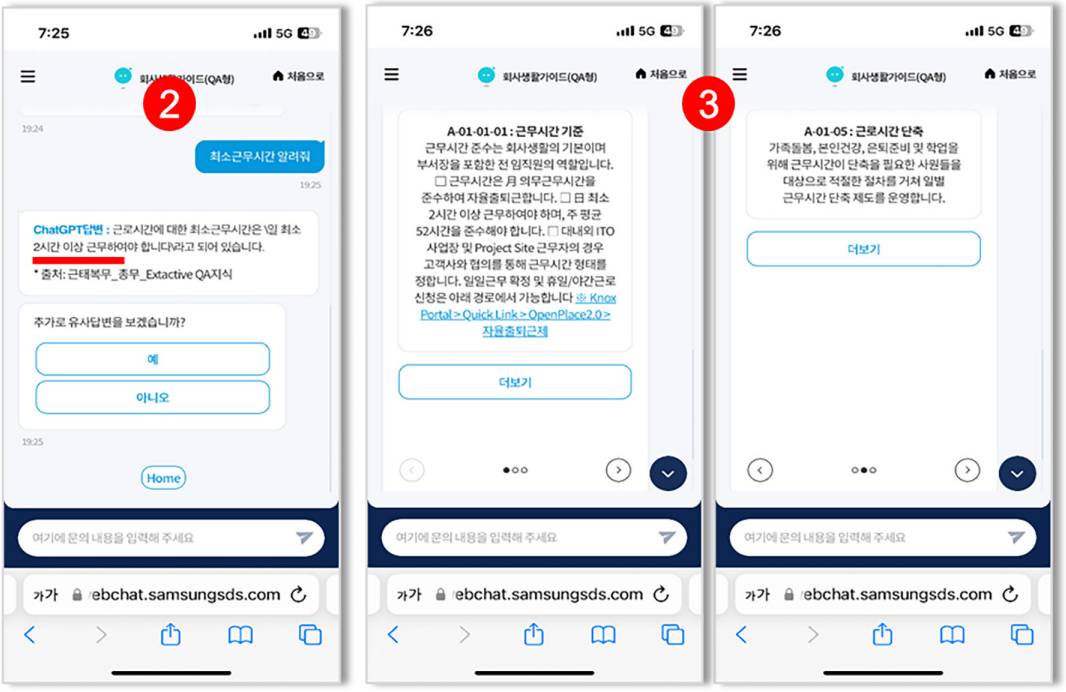
① 근태복무.pdf 화일을 QA 시스템에 업로드 한다(Brity Assistant MRQA 사용).

② QA시스템에서 제공하는 ③번 화면에 나타난 n개의 유사 추천 문맥내용과 질문을 함께 LLM에 전달하여 한개의 답변을 반환한다.

③ QA시스템에서 질문에 대한 n개의 유사 추천 답변 문맥을 사용자가 선택하여 확인할 수도 있다.



〈그림 19〉 QA시스템(Brity Assistant)



〈그림 20〉 QA시스템 활용한 LLM 답변 구현화면

4.2.8. 구현 사례 고찰

본 구현 사례는 Python, LangChain, OpenAI API, ChromaDB, Streamlit, QA 시스템 등을 포함한 다양한 도구를 활용하여 LLM을 기반으로 하는 생성형 AI 서비스를 일련의 구체적 사례로 구현하고, 이를 실 업무에 적용할 수 있는 방법을 제시하였다. 또한 업무 적용을 위한 개발환경에서 운영환경으로 전환할 때 고려해야 할 사항으로는 벡터 데이터베이스의 선택이다. 사례에서는 ChromaDB를 사용했지만 다양한 기능, 성능 및 확장성을 갖춘 수많은 벡터 데이터베이스가 있다. 최적의 성능과 효율성을 위해서는 특정 사용 사례에 적합한 데이터베이스를 신중하게 평가하고 선택하는 것이 필수적이다. 또한 LLM에서의 응답이 예상 범위 내에 유지되고 비즈니스 요구 사항에 부합하는지 확인하기 위해 광범위한 테스트를 수행하는 것이 중요하다. 여기에는 생성형 AI 서비스의 입력 소스 데이터의 품질을 개선하거나, 더 나은 응답 정확도를 달성하기 위해 모델의 매개변수 또는 시스템 프롬프트를 파인튜닝하는 작업이 포함될 수 있다.

본 구현 사례에서 구축된 생성형 AI 서비스는 자체 문서를 사용하여 AI 기반 지원 챗봇의 잠재력을 보여주고 있다. 본 구현 사례 및 앞에서 언급한 고려 사항을 참조한다면 좀 더 쉽게, 기술요소 솔루션을 선택하고 생성형 AI를 업무에 적용할 수 있을 것으로 기대한다. 또한 이것을 지원하는 솔루션을 개발하는데 기여할 것으로 기대한다.

5. 결론 및 논의

본 연구에서는 LLM 애플리케이션 아키텍처를 활용하여 생성형 AI 서비스를 구현하는 방법 및 구현 사례를 제시하였으며 이를 통해 생성형 AI 기술의 발전과 산업적 활용을 촉진할 수 있는 방안을 모색하였다.

LLM 및 생성형 AI 전반에 대해 이론적 배경을 살펴보았으며 이를 통해 생성형 AI 기술의 개념과 LLM 모델의 특징에 대한 이해를 높일 수 있었다. 또한 LLM의 정보 부족 문제를 파인튜닝이나 직접 문서 정보 활용을 통해 극복하는 방법을 검토하고, RAG 모델의 구체적인 작동 방식과 주요 단계에 대해 자세히 알아보았다. 그리고 벡터 DB를 활용한 정보 저장 및 검색 방법에 대해 논의하였다. 최신 LLM 모델과 벡터 DB 기술을 활용하여 어떻게 구현할 수 있는지 비즈니스 도메인에서 실제로 RAG 모델을 적용한 다양한 구현사례를 제시하여 생성형 AI 기술의 현실적인 활용 가능성을 확인하고자 하였으며, 실제 본 연구에서 구현사례를 제시하여 RAG 모델은 검색 기능을 통해 정보를 수집하고, 이를 기반으로 생성된 결과물을 보완하는 방식으로 작동하여, 생성된 결과물이 보다 정확하고 유용한 정보를 제공할 수 있는 것을 확인하였다. 또한 본 연구는 관련 LLM서비스 구현부분에 대한 연구 자료가 많지 않은 상황에서 이를 활용하고자 하는 많은 이해관계자들에게 생성형 AI 기술이 비즈니스 도메인에서 어떻게 구현될 수 있는지에 대한 이해도를 높일 수 있고 참고자료로 활용될 수 있을 것으로 보인다.

본 연구에서는 LLM을 활용하여 기업내 정보를 활용한 생성형 AI 서비스를 구현하는 방법을 제시하였지만, 여전히 몇 가지 한계점이 존재한다. 첫째, LLM 모델의 크기와 복잡성 때문에 모델 학습 및 구현에 많은 시간과 비용이 소요될 수 있다. 둘째, RAG 모델의 생성 결과물이 일관성이 없거나 부적절한 경우가 있을 수 있으며 여

전히, 정보 부족 문제가 여전히 발생할 수 있다. 셋째, 대부분 오픈소스 기반으로 구현된 사례를 제공함으로 기능면에 부족한 부분도 존재하였다.

향후 연구에서는 이러한 한계점을 극복하고 LLM 및 RAG 모델을 더 효율적으로 활용하기 위한 방법을 모색할 필요가 있다. 우선적으로, LLM 모델의 크기와 복잡성을 줄이는 연구가 필요하며 더 작고 더 간단한 모델, 즉 sLLM으로도 효과적인 결과를 얻을 수 있는 방법을 연구할 필요가 있다. 두 번째로, RAG 모델의 검색 기능을 보완하고 정보 부족 문제를 해결할 수 있는 새로운 접근 방법을 개발해서 더 정확하고 효율적인 정보 검색 방식을 적용한다면 모델의 성능을 향상시킬 수 있을 것이다. 세 번째로, RAG 모델이 생성하는 결과물의 일관성과 적절성을 향상시킬 수 있는 방법을 찾아야 한다. 생성된 내용의 품질을 높이기 위한 다양한 기법을 연구하고 적용해보는 것이 중요하다. 또한 기업의 LLM 활용 서비스 업무에 따라 유료화된 고기능/고성능 LLM서비스의 구현 요소 솔루션을 활용하여 산업적 활용 가능성을 향상시키는 것도 고려되어야 한다.

마지막으로 LLM은 다양한 문화와 서로 다른 언어의 맥락을 잘 이해할 수 있는 특화된 부분을 반영한 생성형 AI 서비스를 구현하는 것이 매우 중요하다. 이것은 다양한 지역에서 활용할 수 있는 생성형 AI 기술 발전을 가져오고 산업적 활용을 더욱 촉진시키며 활용 범위를 확장시킬 것이다.

참고문헌(References)

[국내 문헌]

박창호. (2023). 정보화 사회의 문화권력의 변화와 챗GPT의 사이버지배에 대한 탐색적 이해, *사회와 이론*, 2023-2(45), 209-257. https://doi.org/10.17209/st.2023.07.45.209

삼성 SDS. (2023, 4월 28일). ChatGPT 기술 분석 백서 - 2부 ChatGPT 활용. https://www.samsungsds.com/kr/insights/chatgpt_whitepaper2.html

삼성 SDS. (2023, 10월 20일). SAMSUNG SDS Executive Briefing 4분기 2023 생성형 AI 특집호, 마케팅팀.

안정희, 박혜옥. (2023). 생성형 인공지능을 활용한 사례 기반 간호 교육 프로그램 개발, *한국간호교육학회지*, 29(3), 234-246. https://doi.org/10.5977/jkasne.2023.29.3.234

오영환. (2023). Keras 지도학습을 이용한 고교야구 선수의 타율과 OPS 상관 관계 예측 모델, *한국지식정보기술학회 논문지*, 18(4), 935-945. http://dx.doi.org/10.34163/jkits.2023.18.4.014

장민, 안재관. (2023). 프롬프트 엔지니어링, 알투스.

정천수, 정지환. (2020). 포스트 코로나19 언택트 시대 대응을 위한 AI 챗봇 구축방법에 관한 연구, *한국IT서비스학회지*, 19(4), 31-47. https://doi.org/10.9716/KITS.2020.19.4.031

정천수. (2023a). 하이브리드 AI 챗봇 구현을 위한 RPA연계 방안 연구, *정보처리학회논문지/소프트웨어 및 데이터 공학*, 12(1), 41-50. https://doi.org/10.3745/KTSDE.2023.12.1.41

정천수. (2023b). E2E 비즈니스 프로세스 자동화를 위한 하이퍼오토메이션 플랫폼 적용방안 및 사례연구, *경영정보학연구*, 25(2), 31-56. https://doi.org/10.14329/isr.2023.25.2.031

정천수. (2023c). 전통적인 챗봇과 ChatGPT 연계 서비스 방안 연구, *한국정보기술응용학회지*, 3(4), 11-28. https://doi.org/10.21219/jitam.2023.30.4.001

조정임. (2023). 초거대 AI와 생성형 인공지능, *TTA 저널*, 제207호.

최성철. (2023). 기업을 위한 ChatGPT, 프라이빗

챗GPT!, SAMSUNG SDS 인사이트 리포트.

한국정보화진흥원. (2016). 인공지능 기반의 '챗봇(Chat-Bot)' 서비스 등장과 발전 동향, ICT 융합의Issues & Trends, 2016-8월호.

홍기주, 김한준, 전종훈. (2016). 텐서공간모델 기반 시멘틱 검색 기법, 한국전자거래학회지, 21(4), 1-14.

[국외 문헌]

Andreessen Horowitz. (2023, January 19). Who Owns the Generative AI Platform?. https://a16z.com/2023/01/19/who-owns-the-generative-ai-platform/

Arefeen, A., Debnath, B., Chakradhar, S. (2023, September 2). LeanContext: Cost-Efficient Domain-Specific Question Answering Using LLMs. *arXiv preprint arXiv:2309.00841.*

Bommasani, R., et. al. (2021, August 16). On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258.*

Chowdhery, A., et., al. (2022). Palm: Scaling language modeling with pathways. *CoRR*, vol. abs/2204.02311

Devtorium. (2023, July 26). How Vector Databases Can Enhance Custom AI Solutions. https://devtorium.com/blog/how-vector-databases-can-enhance-custom-ai-solutions/

Dilmegani, C. (2023, June 21). Large Language Models: Complete Guide in 2023. https://research.aimultiple.com/large-language-models/

Gartner. (2023, August 23). What's New in the 2023 Gartner Hype Cycle for Emerging Technologies. https://www.gartner.com/en/articles/what-s-new-in-the-2023-gartner-hype-cycle-for-emerging-technologies

Greyling, C. (2023, October 21). Large Language Model (LLM) Disruption of Chatbots. https://cobusgreyling.medium.com/large-language-model-llm-disruption-of-chatbots-8115fffadc22

Huyen, C. (2023, April 21). Building LLM applications for production. https://huyenchip.com/2023/04/11/llm-engineering.html.

IDC. (2023, May). Generative AI Platforms and Applications Market Trends and Forecast 2Q23.

Jeong, C. S. (2023, September 03). A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture. *arXiv preprint arXiv:2309.01105.*

Jeong, J. H. and Jeong, C. S. (2022). Ethical Issues with Artificial Intelligence (A Case Study on AI Chatbot & Self-Driving Car), *International Journal of Scientific & Engineering Research*, 13(1). 468 – 471.

LangChain. (2023, October 28). LangChain Introduction. https://python.langchain.com/docs/get_started/introduction

Mayank, S. (2023, June 30). Generative AI: Empowering Innovation with its Astonishing Capabilities. https://shurutech.com/innovating-with-generative-ai/

Microsoft. (2023, August 01). Retrieval Augmented Generation using Azure Machine Learning prompt flow. https://learn.microsoft.com/en-us/azure/machine-learning/concept-retrieval-augmented-generation?view=azureml-api-2

Nijkamp, E., et., al. (2022). Codegen: An open large language model for code with mtulti-turn program synthesis. *arXiv preprint arXiv:2203.13474.*

Raschka, S. (2023, May 20). Finetuning LLMs Efficiently with Adapters. https://magazine.sebastianraschka.com/p/finetuning-llms-with-adapters

RevFactory. (2023, August 23). RevFactory Project - OpenAI enables GPT-3.5 fine tuning starting

today. https://revf.tistory.com/293

Sánchez-Díaz, X., Ayala-Bastidas, G., Fonseca-Ortiz, P., Garrido, L. (2018). A Knowledge-Based Methodology for Building a Conversational Chatbot as an Intelligent Tutor, *Advances in Computational Intelligence*, Vol. 11289. 165-175. https://doi.org/10.1007/978-3-030-04497-8_14

Sivarajkumar, S., Kelley, M., Samolyk-Mazzanti, A., Visweswaran, S., Wang, Y. (2023, September 18). An Empirical Evaluation of Prompting Strategies for Large Language Models in Zero-Shot Clinical Natural Language Processing. *arXiv preprint arXiv:2309.08008.*

Taylor, R., et., al. (2022). On the Opportunities and Risks of Foundation Models. *CoRR*, vol. abs/2211.09085

Zhao, W. X., et al. (2023, June 29). A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223.*

Abstract

Generative AI service implementation using LLM application architecture: based on RAG model and LangChain framework

Cheonsu Jeong*

In a situation where the use and introduction of Large Language Models (LLMs) is expanding due to recent developments in generative AI technology, it is difficult to find actual application cases or implementation methods for the use of internal company data in existing studies.

Accordingly, this study presents a method of implementing generative AI services using the LLM application architecture using the most widely used LangChain framework. To this end, we reviewed various ways to overcome the problem of lack of information, focusing on the use of LLM, and presented specific solutions. To this end, we analyze methods of fine-tuning or direct use of document information and look in detail at the main steps of information storage and retrieval methods using the retrieval augmented generation (RAG) model to solve these problems. In particular, similar context recommendation and Question-Answering (QA) systems were utilized as a method to store and search information in a vector store using the RAG model. In addition, the specific operation method, major implementation steps and cases, including implementation source and user interface were presented to enhance understanding of generative AI technology. This has meaning and value in enabling LLM to be actively utilized in implementing services within companies.

Key Words : Large Language Model, Generative AI, RAG, LangChain, Vector Store, QA System

Received : October 16, 2023 Revised : November 28, 2023 Accepted : November 29, 2023
Corresponding Author : Cheonsu Jeong

* Corresponding Author: Cheonsu Jeong
SAMSUNG SDS AI Automation Team
Olympic-ro 125, Songpa-gu, Seoul 05510, KOREA
Tel: +82-2-6155-3114, E-mail: csu.jeong@samsung.com

저 자 소 개

정천수

현재 SAMSUNG SDS AI Automation Team에서 부장으로 재직 중이며 고려대학교에서 컴퓨터공학 석사학위와 국민대학교에서 경영정보시스템 박사학위를 취득하였다. 다수의 AI 프로젝트 구축 PM을 하였으며 컴퓨터정보학회논문지, 인터넷정보학회논문지, 정보시스템연구, 지식경영연구, 한국IT서비스학회지, 정보처리학회논문지, 정보기술응용연구, Information Systems Review, AAIML 등을 포함한 다수의 저널에 논문을 게재한 바 있으며 주요 관심분야는 Generative AI, Hyperautomation, Digital Transformation, Conversational AI, Machine Learning, Big Data 등이다.