

지식 기반 QA개선을 위한 Advanced RAG 시스템 구현 방법: Graph Agent 활용

A Graph-Agent-Based Approach to Enhancing Knowledge-Based QA with Advanced RAG

정천수 (Cheonsu Jeong)

SAMSUNG SDS AI Automation Team¹⁾

〈 국문초록 〉

본 연구는 지식 기반 질문-답변(QA) 시스템을 개선하기 위해 기존 RAG(Retrieval Augmented Generation) 모델의 한계를 극복하고, Graph 기반의 향상된 RAG 시스템을 구현하여 품질 좋은 생성형 AI 서비스 개발을 목표로 하고 있다. 기존 RAG 모델은 검색된 정보를 활용해 높은 정확도와 유창성을 보이지만, 한 번 적재된 지식을 재작업 없이 사용해 답변을 생성하기 때문에 정확도가 떨어질 수 있다. 또한, RAG 구성 시점 이후의 실시간 데이터를 반영할 수 없어 맥락 이해 능력이 부족하고 편향된 정보 문제를 야기할 수 있다. 이러한 한계를 개선하기 위해 본 연구에서는 Graph 기술을 활용한 향상된 RAG 시스템을 구현하였다. 이 시스템은 정보를 효율적으로 검색하고 활용할 수 있도록 설계되었다. 특히, LangGraph를 활용하여 검색된 정보의 신뢰성을 평가하고, 다양한 정보를 종합하여 보다 정확하고 향상된 답변을 생성할 수 있도록 하였다. 또한, 구체적인 작동 방식과 주요 구현 단계 및 사례를 구현 코드와 검증 내용을 통해 제시하여 Advanced RAG 기술에 대한 이해를 높였다. 이를 통해 Advanced RAG를 활용한 기업 내 서비스 구현에 실질적인 지침을 제공하여 기업들이 적극적으로 활용할 수 있도록 하는 데 의미가 있다.

주제어: Advance RAG, Agent RAG, 지식 검색, 생성형 AI, LangGraph

1) csu.jeong@samsung.com

1. 서론

최근 AI 기술의 발전과 함께 생성형 AI(Generative AI)가 주목받고 있다. 생성형 AI는 방대한 양의 학습된 데이터 모델을 바탕으로 텍스트, 이미지, 오디오, 비디오와 같은 새로운 콘텐츠를 생성할 수 있는 인공지능의 한 형태로(Jeong, 2023), 일상 대화는 물론 금융, 의료, 교육, 엔터테인먼트까지 다방면에서 적용되고 있으며(안정희, 박혜옥, 2023), 누구나 쉽게 생성형 AI 서비스를 활용할 수 있게 되면서, 생성형 AI 기반 챗봇의 역할이 중요해지고 있다(Adam et al., 2021; Przegalinska et al., 2019; 박은영, 2024). 챗봇은 사용자가 일반적으로 텍스트 또는 음성을 통해 대화를 할 수 있는 지능형 에이전트이다(Sánchez-Díaz et al., 2018; 정천수, 정지환, 2020). 최근 생성형 AI 챗봇은 사람의 감정과 의도를 분석해 답변을 제시하는 수준에까지 이르고 있으며(정천수, 2023a), LLM이 나오면서 자동 대화 생성, 번역 등에 활용할 수 있게 됐다(정천수, 2023b). 하지만, 최신 정보와 상충되는 답변을 생성할 수도 있으며 이전에 학습한 내용을 사용하므로 새로운 문제나 도메인에 대한 이해도가 낮을 수 있다(정천수, 2023c). LLM 출시가 주를 이룬 2023년은 ChatGPT, Llama-2와 같은 기초적인 대형 언어 모델(LLM)들에 관한 해였다면, 2024년은 RAG(Retrieval Augmented Generation; 검색 증강 생성)와 AI Agent의 해가 될 것이라고 다수의 필드 전문가들은 예측하고 있다(스칼터랩스, 2024). 그러나, 기업이 생성형 AI 서비스를 도입하기 위해서는 다양한 검토 사항이 존재한다. ‘기업 내부 데이터를 참조한 정확한 답변이 가능 할 것인지’, ‘사내 데이터가 유출될 가능성은 없는지’, ‘기업 시스템과 생성형 AI를 어떻게 연결 할 것인지’ 등과 같은 기업의 많은 고민이 존재한다. 이에 대한 해결방안으로 도메인 특화 파인튜닝 LLM 활용, 내부정보 이용으로 신뢰성 향상을 위한 RAG 등 다양한 방법이 모색되고 있다(정천수, 2024). GPT-4 LLM을 기

준으로 도메인별 정보를 파인튜닝 한 후 정확도가 75%에서 81%로 올라가고, 여기에 RAG를 추가하면 정확도가 86%까지 올라가는 경우도 확인되었다(Angels et al., 2024). 이렇게 RAG 모델은 내부 지식 검색과 생성을 결합하여 더욱 정확한 답변을 생성하는 데 효과적인 모델로 알려져 있다. 특히, 기업내·외 지식 베이스 연계를 통해 출처 기반한 사실(Fact) 제공 및 최신 데이터(Data Freshness) 이슈를 해결하는 장점이 있다.

따라서, 현재 많은 기업들이 LLM과 RAG 기술을 활용하여 고객 서비스, 정보 검색 등 다양한 분야에 AI 시스템을 도입하고 있다. 하지만 이러한 시스템은 데이터 편향, 환각 문제, 실시간 데이터 처리의 어려움, 특정 분야에 대한 지식 부족 등의 한계를 가지고 있다. 특히, 사용자의 질문에 시스템이 답변을 찾지 못할 때 ‘죄송합니다, 모르겠습니다’라고 명확하게 답변하지 못하고, 관련 없는 정보를 마치 사실인 것처럼 답변을 제시하는 환각(Hallucination)이 발생한다. 또한, 정확한 답변이 존재하지만 검색 결과의 순위가 낮아 답변에서 누락되는 경우도 발생한다. 특히, 기업의 사례를 보면, 금융 회사의 경우 과거 데이터로 인한 잘못된 예측으로 손실을 입고, 전자 상거래 기업은 실시간으로 재고 현황이나 배송 상태와 같은 정보를 반영하지 못해 고객 만족도가 크게 하락하는 문제가 발생한다. 또한 제조업체는 기술 지원을 위한 RAG 기반 시스템을 도입했으나, 시스템이 복잡한 기술 용어와 전문 지식을 제대로 이해하지 못해 고객들이 잘못된 정보를 제공받는 사례가 발생하게 된다(Scott et al., 2024; 정천수, 2024).

이렇게 RAG모델의 효과는 고품질의 데이터베이스에 크게 의존하고 있기 때문에 데이터의 질이 모델 성능에 직접적인 영향을 미친다(김종철, 2024). 기존의 RAG 모델은 한 번에 지식을 적재한 후, 재작업 없이 답변을 생성하므로 정확도 저하 문제와 RAG구성 시점 이후의 실시간 데이터를 볼 수 없다. 이렇게 처음에 한번 벡터화한

지식정보를 한 번의 답변 생성 과정을 거치기 때문에, 답변이 부정확할 경우에 지식정보를 새로운 정보로 재 작업하는 과정 없이 답변을 제공하여 답변의 정확도가 저하되는 경우가 발생할 수가 있다. 특히 복잡한 질문에 대한 답변을 생성할 때 이러한 문제가 생길 수 있으며, 기존 RAG 모델은 다양한 질문 유형에 효과적으로 대응하기 어려운 측면이 있었다. 잘못된 검색 전략으로 인해 관련 없는 문서가 질문에 답변하는 데 사용될 수 있고, 여전히 LLM에서 나타나는 환각을 겪거나 질문에 대답하지 못하는 경우가 발생하는 한계점을 가지고 있다(전준영 등, 2024)

본 연구는 이러한 전통적인 QA 시스템에서 발생하는 정보 검색 및 환각 문제를 해결하기 위한 새로운 접근 방식을 제안하고자 한다. 특히, 실시간 데이터 접근과 문서 검증 프로세스를 통합하여 질문에 대한 정확한 답변을 제공할 수 있도록 하는 것을 목표로 한다. 실시간 데이터에 접근하고, 검색된 문서가 실제로 질문에 응답하는 데 관련이 있는지 확인한다. 이를 통해 RAG 시스템은 최근 이벤트와 실시간 데이터에 대한 질문에 답할 수 있고 환각에 덜 취약하게 향상된 RAG 시스템을 구현함으로써 생성형 AI 서비스의 품질 및 성능을 향상시키는 데 기여할 것이다.

본 논문의 서론에서는 연구의 배경과 목적, 기존 RAG 모델의 한계, 연구의 중요성 및 기여도, 논문의 구조에 대해 설명한다. 이론적 배경에서는 RAG 모델의 개요와 Advanced RAG 방안, 기존 연구 개선 유형 사례를 검토하고, Advanced RAG 모델의 설계에서는 Advanced RAG의 구성 흐름과 Agent RAG 구성, 기타 향상된 기능에 대해 다룬다. 시스템 구현에서는 LangGraph의 개요 및 적용 방법, 시스템 구현 과정 및 결과를 기술하고, 테스트에서는 구현된 코드의 개선된 결과값을 제시하고 있다. 마지막으로 결론에서는 연구 결과의 요약, 연구의 한계 및 향후 연구 방향에 대해 다루고 있다.

2. 이론적 배경

본 연구를 위해 RAG 모델과 관련된 자료에 대하여 최근 발표된 주요 연구 논문, 저널 기사 등을 면밀히 분석하고 조사하였다. 본 장에서는 RAG 모델의 개념부터 최신 연구 동향인 Advanced RAG까지 다루고 있다.

2.1. RAG 모델의 개요

RAG 모델은 질문에 대한 답변을 생성하기 위해 검색과 생성이라는 두 가지 과정을 결합한 모델이다(Lewis et al., 2020). 마치 사람이 궁금한 것을 해결하기 위해 책이나 인터넷을 검색하고, 찾은 정보를 바탕으로 답변을 구성하는 것과 비슷하다. 이 모델은 질문에 대한 답변을 생성하기 위해 먼저 관련 문서를 검색하고, 이를 바탕으로 답변을 생성한다. 이 과정은 질문에 대한 정확한 답변을 생성하는 데 도움이 된다. RAG 모델은 다양한 질문 유형에 대응할 수 있으며, 특정 도메인에 대한 지식이 부족할 경우에도 효과적으로 답변을 생성할 수 있다. 따라서 기존 생성 모델에 비해 정보의 정확성과 일관성을 높일 수 있다는 장점을 가지고 있다.

RAG 모델은 아래와 같이 두 단계로 구성된다.

- **검색 단계:** 주어진 질문과 관련된 정보를 지식에서 검색 엔진을 통해 검색한다.
- **생성 단계:** 검색된 정보를 기반으로 답변을 생성한다.

2.1.1. RAG 모델 구현 흐름

RAG 모델은 텍스트 생성 작업을 수행하는 모델로 주어진 소스 데이터로부터 정보를 검색하고, 해당 정보를 활용하여 원하는 텍스트를 생성하는 과정을 수행한다(정천수, 2023d). RAG 사용을 위한 데이터처리 과정은 원본 데이터를 청크(Chunk)단위의 작은 조각으로 나누고 텍스트 데이터를 숫자인 벡터로 변환하는 임베딩(Embedding)

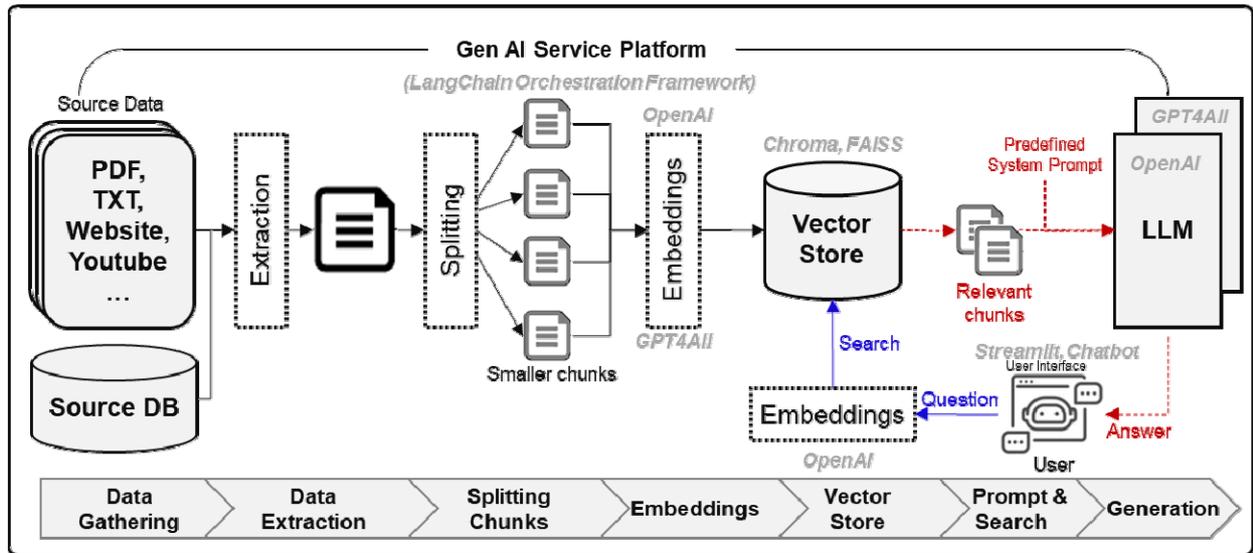
을 거쳐 벡터 저장소에 저장된다(Microsoft, 2023). 이러한 RAG모델 기반 생성형 AI 서비스 구현 흐름은 <그림 1>과 같다(정천수, 2023d).

2.1.2. RAG기반 Vector Store 유형

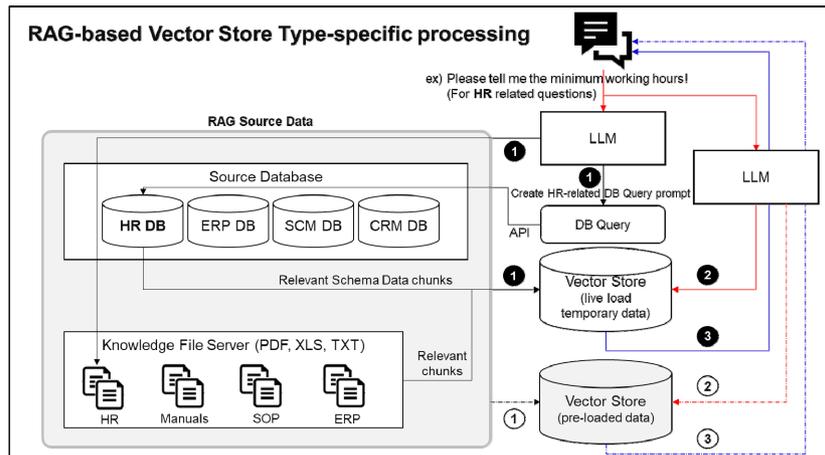
RAG시스템을 구축하기 위해서는 지식이 저장되는 벡터 데이터베이스를 사용하게 되는데, 벡터 데이터베이스에 대한 일반적인 벡터 파이프라인은 인덱싱(Indexing), 조회(Querying), 사후처리(Post Processing)와 같은 3단계

를 거친다(Devtorium, 2023). 특히, RAG기반 벡터 저장소(Vector Store) 저장 유형은 <그림 2>와 같이 모든 소스 데이터를 사전에 Vector Store에 저장하는 경우와 질문 시 실시간으로 넣는 경우로 나눌 수 있다(정천수, 2023d).

기업에서는 내부 지식을 Open LLM을 통해 서비스하게 되었을 때 보안 이슈 때문에 Local LLM을 사용하는 것이 중요한 이슈이다(정천수, 2023d). 이때 각 구간별로 가장 잘 처리할 수 있는 Local LLM을 여러 개 구성하여 사용하는 것이 효과적인데, <그림 2>에서 여러 개의 소



<그림 1> RAG기반 생성형 AI 서비스 구현 흐름



<그림 2> RAG기반 Vector Store 구성 유형 및 처리절차

스 데이터베이스에서 질문에 맞는 데이터베이스의 내용을 가져오는 DB Query를 생성하는데 특화된 LLM이나, 특정지식에서 답변을 생성해줄 수 있는 도메인 특화된 LLM을 별도 구축할 수도 있다(정천수, 2023d).

2.2. Advanced RAG 선행 연구

2.2.1. RAG 성능을 향상시키는 방안

RAG는 외부 저장소에서 검색하는 질문 처리 결과에 따라 프롬프트로 구성할 수 있는 데이터의 질에 영향을 받는다. 최근에는 이러한 일반적인 RAG의 한계를 해결하기 위해 다양한 Advanced RAG 방안들이 제시되고 있

다. Advanced RAG는 기존 RAG 기법의 진화된 형태로, 일반적인 RAG의 한계를 극복하고자 다양한 최적화 기법을 적용한 접근 방식이다. 최근 Yunfan G., et al. (2024) 연구자들은 Advanced RAG를 검색 부분을 사전검색(Pre-Retrieval), 검색(Retrieval), 사후검색(Post-Retrieval) 단계로 나누고 각 단계의 최적화를 실시해 정보의 정확성과 처리 효율을 크게 향상시키는 최적화 전략을 제시하고 있다. 또한 적절한 컨텍스트를 찾고, 적절한 응답을 생성하는 RAG 시스템의 품질 향상을 위한 전략으로 관련성(Relevance)에 따라 다시 랭킹(Re-Ranking)하여 정확도를 향상하는 방법(장동진, 2024) 등과 같은 다양한 개선 방법들이 <표 1>과 같이 제시되고 있다(Matt, 2023).

<표 1> RAG 성능을 향상시키는 방법

방법	내용
Clean your data	정보가 상충되거나 중복되는 데이터를 사용하는 경우 검색 시 올바른 컨텍스트를 찾는 데 어려움을 겪게 됨 - Query에 대한 대답을 제대로 할 수 있도록 문서 자체를 잘 구축해 놔야 함, 모든 문서의 Summary를 만들어서 context 로 사용하는 것이 하나의 방법
Explore different index types	임베딩 기반의 유사도 검색 방식은 보통은 잘 동작하지만, 항상 Best 인 것은 아님 - 전자상거래 에서 제품과 같은 특정 항목을 찾는 것에 대한 검색 은 키워드 검색 방식이 더 적합할 수 있고, 많은 시스템에서 하이브리드 방식을 사용(예; 특정 상품을 찾는 것은 키워드 기반 검색, 고객 정보, 지원에 대한 일반적인 것은 임베딩 기반의 검색)
Experiment with your chunking approach	Chunk 사이즈는 매우 중요하며, 일반적으로 작은 Chunk 일 때 더 성능이 좋으나 주변 정보의 부족한 문제가 발생함 - 일반적으로 청크 크기가 작을수록 검색 시스템이 관련 컨텍스트 부분을 찾는 데 도움이 됨
Play around with your base prompt	할루시네이션을 줄이기 위해 주어진 컨텍스트정보에서 만 답변을 하도록 프롬프팅 - 예: ‘당신은 고객지원 상담원이다. 사실에 근거한 정보만 제공하면서 최대한 도움을 주도록 설계되었다. 사전학습 지식이 아닌 Given the context information를 바탕으로 질의에 답하세요.’
Try meta-data filtering	청크에 관련성 있는 메타데이터 태그를 추가한 다음 이를 사용하여 결과를 처리(문서명, 페이지, 이메일, 날짜 등)
Use query routing	하나 이상의 인덱스를 갖는 것이 유용한 경우가 많음. 그런 다음 쿼리가 들어올 때 적절한 인덱스로 라우팅 - 예: 요약 질문을 처리하는 인덱스 하나, 지적 질문을 처리하는 인덱스 하나, 날짜에 민감한 질문에 적합한 인덱스가 있을 수 있음
Look into reranking	랭킹 재지정을 사용하면 검색 시스템이 컨텍스트에 대한 유사한(Similarity) 상위 노드를 가져온 후, 다음 관련성(Relevance)에 따라 다시 랭킹하여 정확도 향상
Consider query transformations	첫 질문에 대한 관련 컨텍스트를 찾지 못하면, 질문 변경을 해서 다시 시도하여 답변 정확도 향상 - 쿼리가 여러 질문으로 분해되도록 RAG 시스템에 이를 구축할 수 있음
Fine-tune your embedding model	Context, Domain 에 맞지 않는 경우 Embedding model을 fine-tuning 을 진행하여 성능을 높일 수 있음 - 예: Domain specific 용어
Start using LLM dev. tools	Llamaindex 또는 LangChain을 사용하여 RAG 시스템을 구축 시 문서, 컨텍스트 출처 등을 알 수 있는 디버깅 도구 사용

LangChain 또는 LlamaIndex 프레임워크는 이러한 전략을 구현할 수 있도록 각 항목별 라이브러리를 제공하고 있어 구현을 좀더 쉽게 할 수 있도록 하고 있다.

2.2.2. Advanced RAG 유형 연구 및 개선 방향

현재 연구되고 있는 대표적인 Advanced RAG 방안을 살펴보고, 각 유형의 강점과 약점을 분석하여 본 연구의 방향성을 제시하고자 한다.

- **Self-RAG:** 이 방식은 생성된 답변을 다시 검색하여 관련 정보를 찾고, 이를 기반으로 답변을 개선하는 방법이다. 이를 통해 답변의 정확도와 유창성을 향상시킬 수 있다(Asai et al., 2023). 그러나 이 접근법은 검색된 정보의 품질에 크게 의존하며, 정보의 신뢰성을 평가하는 메커니즘이 부족하다는 단점이 있다. 또한, 반복적인 검색과 생성 과정에서 많은 리소스가 소모되므로, 이를 보완하기 위해 초기 검색 단계에서 더 정교한 필터링을 통해 검색 범위를 줄이고, 반복적인 처리를 최소화하는 최적화 기법이 필요하다.
- **Corrective RAG:** 이 방식은 생성된 답변의 오류를 수정하기 위해 Corrective Agent를 사용하는 방법이다. Corrective Agent는 답변의 오류를 식별하고, 이를 수정하기 위한 정보를 검색한다. 이를 통해 답변의 신뢰성을 높일 수 있다(Yan et al., 2024). 그러나 초기 검색 결과가 부정확할 경우 이를 수정하는 과정이 명확하게 정의되지 않아, 추가적인 처리 시간이 필요하게 되며, 실제 적용 시 비효율성을 초래할 수 있다. 이를 보완하기 위해, 수정 프로세스를 구체화하고, 사전 검토 단계에서 잠재적인 오류를 미리 감지하여 이를 기반으로 답변을 생성하도록 유도하는 기법을 도입하면, 보다 효과적인 정보 검색 및 응답 생성이 가능해질 것이다.

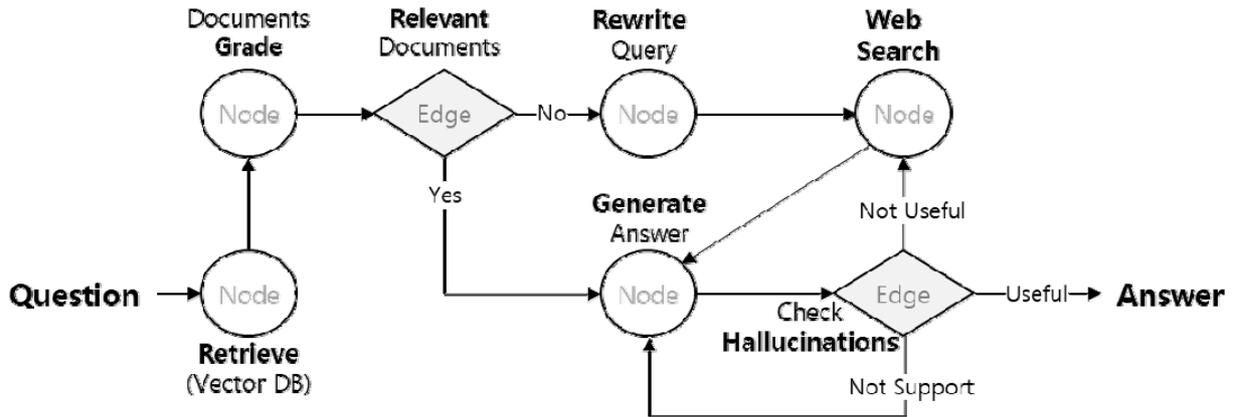
- **Adaptive RAG:** 이 방식은 질문의 유형에 따라 적절한 RAG 방식을 선택하여 적용하는 방법이다. 예를 들어, 사실적 질문에는 Self-RAG 방식을, 의견 질문에는 Corrective RAG 방식을 사용하는 등 질문 유형에 따라 최적의 방식을 선택함으로써 답변의 정확도를 높일 수 있다(Jeong et al., 2024). 그러나 현재 연구에서는 피드백 루프의 설계와 구현에 대한 구체적인 사례가 부족하다는 한계가 있다. 이를 개선하기 위해서는 질문 유형을 정확히 분류하는 것이 핵심 과제로, 분류의 정밀도를 높이기 위해 추가적인 학습 데이터를 활용하거나, 특정 도메인에 맞춘 분류 기법을 도입하는 방법을 고려할 수 있다.

본 연구에서는 LangGraph와 같은 그래프 기반 기술 및 에이전트를 활용하여, 검색된 정보의 신뢰성을 효과적으로 평가하고, 사전에 잠재적인 오류를 미리 감지함으로써 응답 품질을 개선할 수 있는 방법을 제시하고자 한다.

3. Advanced RAG 모델의 설계

본 장에서는 기존 연구에서 제안된 다양한 Advanced RAG 방안들을 검토하고, 이를 기반으로 향상된 RAG 시스템을 설계한다. 특히, Self-RAG, Corrective RAG, Adaptive RAG 등의 방안들을 면밀히 분석하고, 이를 통해 얻은 개선점을 바탕으로 <그림 3>과 같이 구현 모델을 제시한다. Agent RAG 시스템에 대한 구현은 Corrective RAG를 기본으로 하고 Self-RAG, Adaptive RAG를 참조하고 있다.

일반적인 RAG 시스템을 향상시키기 위한 워크플로는 기존처럼 벡터 데이터베이스에서 문서 청크를 검색한 다음 LLM을 사용하여 검색된 각 문서 청크가 입력 질문과



〈그림 3〉 Agent 기반 Advanced RAG Workflow

관련이 있는지 확인하는 것이다. 검색된 모든 문서 청크가 관련이 있는 경우, 표준 RAG 파이프라인 과 같은 정상적인 응답 생성을 위해 LLM으로 이동 한다. 그러나 일부 검색된 문서가 입력 질문과 관련이 없다고 가정한다. 이 경우 입력 쿼리를 다시 표현하고 웹을 검색하여 입력 질문과 관련된 새로운 정보를 검색한 다음 LLM으로 보내 응답을 생성하며 이러한 워크 플로우는 Graph 기반으

로 Advanced RAG 시스템을 구현하게 된다.

3.1. Advanced RAG 모델 설계 절차

Advanced RAG 모델 설계 절차는 <표 2>와 같이 초기 설계단계를 거쳐 데이터 수집 및 전처리, 그래프 구조 설계, LangGraph 통합으로 구성된다.

〈표 2〉 Advanced RAG 모델 설계 절차

Phase	Task	Description
초기 설계 단계	RAG 시스템 요구 사항 정의	RAG 시스템이 충족해야 할 요구 사항을 명확히 하여, 정보 검색의 정확성, 신뢰성, 실시간 데이터 반영 등을 고려한다.
	LangGraph 모듈의 기능 정의	LangGraph의 기능을 명확히 정의하여 텍스트 데이터를 그래프 형태로 변환하는 방법과 그 이점을 이해한다. 예를 들어, 노드와 엣지의 정의, 그래프의 시각화 방법 등을 포함한다(LangGraph, 2024).
소스 데이터 수집 및 전처리 단계	데이터 수집	다양한 출처에서 텍스트 데이터를 수집한다. 이 데이터는 고객 문의, 문서, 웹 페이지 등에서 가져올 수 있다(정천수, 2023d).
	데이터 정제	수집된 데이터를 정제하여 노이즈를 제거하고, 그래프 생성을 위한 형식으로 변환한다. 이 과정에서는 불필요한 정보 제거, 텍스트 정규화 등이 포함된다.
그래프 구조 및 데이터 흐름 설계 단계	그래프 구조 설계	노드와 엣지의 구조를 설계하여 정보의 관계를 명확히 한다. 예를 들어, 주제 별로 노드를 그룹화하고, 관련된 정보 간의 관계를 엣지로 연결한다 (LangGraph, 2024).
	데이터 흐름 설계	데이터가 시스템 내에서 어떻게 흐름지를 정의한다. 이는 데이터 수집, 전처리, 그래프 생성, 검색 및 응답 생성의 흐름을 포함한다.
LangGraph 통합 단계	노드 및 엣지 생성	정제된 데이터를 기반으로 LangGraph를 사용하여 노드와 엣지를 생성한다. 이 과정에서 각 노드의 속성과 엣지의 관계를 정의한다.
	시스템 성능 평가	생성된 그래프의 성능을 평가하기 위한 기준을 설정하고, 이를 통해 시스템의 정확성과 신뢰성을 검증한다.

3.2. Advanced RAG Graph구성 흐름

Advanced RAG 모델은 기본적인 RAG 모델의 흐름을 유지하면서, 검색 이후 단계에서 추가적인 과정을 거쳐 답변의 정확도와 일관성을 높인다. Advanced RAG 모델의 구성 흐름은 다음과 같다.

- ① 질의 처리: 사용자의 질의를 입력 받고, 의도 파악 및 분석을 통해 질의의 정확한 의미를 파악한다.
- ② 검색: 질의와 관련된 정보를 탐색하기 위해 검색 엔진을 활용하여 다양한 정보원에서 검색을 수행한다.
- ③ 검색 결과 후보 선정: 검색된 결과 중에서 질의와의 관련성이 높고 신뢰도가 높은 정보들을 후보로 선정한다.
- ④ 후보 랭킹: 선정된 후보들을 질의와의 부합도, 정보의 신뢰도, 다양성 등을 고려하여 랭킹을 정한다.
- ⑤ 답변 생성: 랭킹된 후보 정보들을 기반으로 텍스트 생성 모델을 활용하여 답변을 생성한다.
- ⑥ 답변 갱신: 답변 생성 후에도 지속적으로 새로운 정보를 수집하고 답변을 갱신하여 최신 정보를 제공한다.

Advanced RAG 모델의 작동 방법은 다음과 같다.

- ① 질문을 입력한다.
- ② 검색 엔진을 통해 질문과 관련된 정보를 검색한다.
- ③ 정보 추출기를 통해 검색된 정보에서 질문에 대한 답변과 관련된 정보를 추출한다.
- ④ 추출된 정보를 기반으로 답변을 생성한다.
- ⑤ Agent가 답변의 정확도와 유창성을 향상시키기 위해 답변을 수정한다.
- ⑥ 최종 답변을 출력한다.

3.3. Agent RAG 적용 방식

향상된 수행 절차를 적용하기 위해 LLM이 Agent도구

를 사용하여 RAG를 선택하게 구성하는 Agent RAG는 답변 생성 과정에 Agent라는 개념을 도입하여 답변의 정확도와 일관성을 더욱 향상시키는 방식이다. Agent는 답변 생성 과정에서 다음과 같은 역할을 수행하는 핵심 요소이다.

- **답변 평가:** 생성된 답변의 정확도, 유창성, 신뢰성 등을 평가한다.
- **답변 개선:** 평가 결과를 바탕으로 답변을 개선한다.
- **정보 검색:** 답변을 개선하기 위해 필요한 정보를 검색한다.

3.4. LangGraph 모듈 적용

LangGraph는 LangChain에서 출시한 모듈로 LLM을 사용하여 상태가 있는 다중 Actor 애플리케이션을 빌드하기 위한 라이브러리이다. Agent 및 다중 Agent 워크플로우를 만드는 데 사용되며 LangGraph를 사용하면 대부분의 Agent 아키텍처에 필수적인 사이클을 포함하는 흐름을 정의할 수 있고, 신뢰할 수 있는 Agent를 만드는 데 중요한 애플리케이션의 흐름과 상태를 모두 세부적으로 제어할 수 있다(LangGraph, 2024).

LangChain 위에 구축된 LangGraph는 LLM으로 구동되는 AI 에이전트를 개발할 때 순환 그래프를 만드는 것을 용이하게 한다. LangGraph는 에이전트 워크플로우를 순환 그래프 구조로 처리하며, 특히 LangGraph Conversational Retrieval Agent는 언어 처리, AI 모델 통합, 데이터베이스 관리, 그래프 기반 데이터 처리 등 다양한 기능을 제공한다. 이러한 기능은 언어 기반 AI 애플리케이션 개발에 필수적인 도구로, LangGraph는 노드(Node), �지(Edge), 상태(State)로 구성되며 각 구성 요소의 역할은 다음과 같다.

- **노드:** 개별 작업 또는 프로세스를 나타내며, 에이전트의 주요 단계 또는 상태를 정의

- **엣지:** 노드 간의 연결을 담당하며, 에이전트가 특정 조건에 따라 다른 단계로 전환되는 경로를 형성
- **상태:** 각 노드가 수행한 작업들의 현재 상태를 기억하며, 워크플로우 내에서 에이전트가 어느 단계에 있는지를 나타냄.

이러한 구조로 LangGraph는 복잡한 AI 에이전트의 워크플로우를 효율적으로 관리하고 실행할 수 있다. 또한, LangGraph의 그래프 생성 과정은 다음과 같다

- **키워드 추출:** 텍스트에서 중요한 키워드를 식별
- **관계 정의:** 추출된 키워드 간의 관계를 정의하여 엣지를 생성. 이 과정은 자연어 처리(NLP) 기법을 활용하여 의미적 유사성을 평가할 수 있음.
- **그래프 구축:** 노드와 엣지를 기반으로 그래프를 구축하고 그래프는 정보 검색 및 분석에 활용.

이러한 LangGraph는 정보 검색, 질의 응답 시스템, 추천 시스템, 고객 지원 시스템 등에서 사용자 질문과 관련된 정보를 그래프 형태로 시각화 하여 보다 효율적인 검색과 응답을 제공하는 데 사용될 수 있다.

Graph는 다음과 같은 중요한 역할을 수행한다:

- **복잡한 워크플로우 관리:** 상태 기반 워크플로우를 구성하고 관리하는 데 매우 유용하며, 각 단계의 상태 전환과 분기 처리가 명확하게 이루어진다.
- **명확한 흐름 제어:** 각 노드와 엣지 간의 관계를 명확하게 정의할 수 있어, 복잡한 조건부 로직과 상태 전환을 쉽게 구현할 수 있다.
- **확장성:** 새로운 노드와 엣지를 쉽게 추가함으로써 워크플로우를 확장할 수 있으며, 다양한 조건부 로직을 포함한 복잡한 시스템을 유연하게 구성할 수 있다.

본 연구에서는 에이전트 기반의 Advanced RAG 구현에 적합한 도구로서, 이러한 많은 기능을 제공하는 LangGraph를 활용한 구현 방법을 제시하고자 한다.

4. 시스템 구현

본 장에서는 3장에서 소개한 Advanced RAG 설계 모델을 기반으로, 기업 내부에서 사용할 수 있는 데이터를 활용하여 RAG 모델을 LangChain 프레임워크를 통해 구현한 사례를 다룬다. 특히, 에이전트 구현에 적합한 LangGraph를 사용한 방법과, 이를 구축할 때 고려해야 할 사항들을 구체적인 구현 사례를 통해 제시한다.

4.1. 구현개요 및 개발 환경

구현에 사용된 RAG 지식 문서는 기업 내부에서 관리되는 문서 중에서 공통적으로 활용할 수 있는 “근무 복잡 규정 관련 문서”를 수집하여 활용하였다. 이 문서에서 유의미한 정보를 추출한 후, 이를 전처리 및 정제 과정을 거쳐 사용하였다. <그림 3>에 제시된 구현 모델에 따라 노드와 엣지의 구조를 설계하고, LangGraph를 통해 노드와 엣지를 통합하는 단계를 진행하였다. 또한, Advanced RAG 구성 흐름에서 검색결과 후보를 랭킹할 때 고려해야 할 부합도, 정보의 신뢰도, 다양성 등의 측정 기준은 LangChain 프레임워크에서 제공하는 다양한 라이브러리(예: Embeddings, Similarity_threshold_retriever 등)를 활용하여 구현하였다.

본 사례에 적용할 솔루션과 개발 플랫폼의 프레임워크는 <그림 1>에 제시된 것을 기본으로 하여, LangGraph와 OpenAI LLM을 활용한 구현 방법을 제시하고자 한다. 기본적으로 문서를 청킹하고, 임베딩하여 Chroma DB에 저장한다. 다음으로 문서 내용 검색을 위해 검색기(Retriever)로 변환, 검색결과 판단하는 단계를 거치며 이를 위해 Agent RAG Graph를 정의하고 구현하게 된다. 개발 언어는 AI 개발에 필요한 다양한 라이브러리가 있는 Python 활용하여 서비스를 구현하였다. <표 3>은 각 구현 요소별 개발 환경이다.

<표 3> 구현 요소별 개발 환경

구현 요소	개발 환경 및 도구
Orchestration Framework	LangChain
Agent Graph Workflow	LangGraph
Workflow Trace	LangSmith
Data Loader & Chunking	LangChain Package
Embedding	Text-embedding-3-small
Vector Store	Chroma
LLM	GPT-4-turbo
Python 개발 환경	Google Colab

4.2. 시스템 구현 과정 및 결과

4.2.1. 기본 라이브러리 설치 및 API Key 설정

기본적으로 데이터 분할 등 개발 전반을 오케스트레이션(Orchestration) 하는 LangChain과 OpenAI, RAG 지식 저장을 위한 Chroma DB, 그리고 웹 검색, 워크플로우 구성을 위한 LangGraph 등 기본 라이브러리를 설치한다. 또한 Colab에서 Google드라이브를 연동해서 지식 파일을 쉽게 관리할 수 있게 하였다. 사용되는 다양한 모듈들의 Key값은 보안을 위해 .env 환경 파일에 별도로 등록하였으며, Key 값을 포함한 .env 파일이 정상적으로 로드된 것을 <그림 4>에서 확인할 수 있다.

```

1 | pip install -q langchain_openai langchain_community langchainhub langchain langgraph
2 | pip install -q tiktoken chromadb pypdf tavily-python
3 | pip install -q python-dotenv # .env file load

1 # Google Drive Connect
2 from google.colab import drive
3 drive.mount('/content/drive')

Mounted at /content/drive

1 cd /content/drive/MyDrive/RAG
/content/drive/MyDrive/RAG

1 # api key
2 from dotenv import load_dotenv # Add
3 load_dotenv() # load .env

True
    
```

<그림 4> 기본 라이브러리 설치 및 API Key 설정

4.2.2. 검색기(Retriever) 구현

기업 내부 정보로 관리되는 문서 중 ‘근무복장기준.pdf’ 문서를 PyPDFLoader를 사용하여 주어진 위치에서 관련 문서를 로드 한다. 문서를 청크로 분할하는 과정에서, 표와 같은 문서의 특성을 고려하여 단순히 하나의 구분자(예: 줄바꿈)를 기준으로 분할하는 방식인 CharacterTextSplitter 대신, 문맥 유지를 효율적으로 하기 위해 chunk_size와 chunk_overlap을 조정하는 RecursiveCharacterTextSplitter를 사용했다. 이 방법은 문서의 구조를 더 잘 반영하며, 분할된 청크들 사이의 문맥 연결성을 높여준다. 분할된 문서는 Chroma 벡터 저장소에 저장되고, OpenAIEmbeddings를 활용해 임베딩 작업이 이루어진다. 이후, 저장된 문서는 검색을 용이하게 하기 위해 retriever로 변환되어 <그림 5>와 같이 활용된 것을 확인할 수 있다.

```

1 from langchain.text_splitter import RecursiveCharacterTextSplitter
2 from langchain_openai import OpenAIEmbeddings
3 from langchain_community.vectorstores import Chroma
4 from langchain_openai import ChatOpenAI
5
6 llm = ChatOpenAI(model="gpt-4-turbo", temperature=0)
7 openai_embed_model = OpenAIEmbeddings(model="text-embedding-3-small")

1 from langchain.document_loaders import PyPDFLoader
2 loader = PyPDFLoader("/content/drive/MyDrive/RAGdata/근무복장기준.pdf")
3 docs = loader.load()
4
5 text_splitter = RecursiveCharacterTextSplitter(chunk_size=300, chunk_overlap=30)
6 chunked_docs = text_splitter.split_documents(docs)
7 chunked_docs[:2]

1 # Load the documents to vectorstore
2 vectorstore = Chroma.from_documents(documents=chunked_docs, collection_name="rag_pdf_db",
3 embedding=openai_embed_model,)
4 retriever = vectorstore.as_retriever()
5 print(retriever)

tags=['Chroma', 'OpenAIEmbeddings'] vectorstore=<langchain_community.vectorstores.chroma.Chroma object
    
```

<그림 5> Data Load 및 임베딩

4.2.3. 검색결과 판단

LLM 자체를 사용하여 검색된 문서가 주어진 질문과 관련이 있는지 평가하기 위해 <그림 6>과 같이 구현한다. 검색된 문서를 사용자 질문과 연결하여 문서에 사용자 질문과 관련된 키워드가 포함되어 있는 경우, 관련성 있는지 여부를 판단하는 단계로 목표는 잘못된 검색을 필터링하는 것이다. <그림 7>과 같이 문서가 질문과 관련이 있

는지 여부를 나타내기 위해 “GRADE: binary_score=’yes’”
처럼 이진 점수로 ‘yes’ 또는 ‘no’를 부여한다.

```
1 ## Implement the Retrieval Grader
2 from langchain_core.prompts import ChatPromptTemplate
3 from langchain_core.pydantic_v1 import BaseModel, Field
4 from langchain_openai import ChatOpenAI
5 # Data model for LLM output format
6 class GradeDocuments(BaseModel):
7     """검색된 문서에 대한 관련성 평가를 위한 바이너리 점수."""
8     binary_score: str = Field(
9         description="문서가 질문과 관련이 있는지, 'yes' 또는 'no'."
10    )
11 # LLM for grading
12 llm = ChatOpenAI(model="gpt-4-turbo", temperature=0)
13 structured_llm_grader = llm.with_structured_output(GradeDocuments)
14 # Prompt template for grading
15 SYS_PROMPT = """당신의 임무는 검색된 문서와 사용자 질문의 관련성을 평가하는 것입니다. 평가하려면 다음 지침을 따르세요
16 - 문서에 질문과 관련된 키워드나 의미론적 의미가 포함되어 있는 경우 'yes'로 관련성 평가를 매깁니다.
17 - 당신의 평가용 문서는 질문과 관련이 있는지 여부를 나타내는 'yes' 또는 'no'여야 합니다. """
18 # Create the prompt
19 grade_prompt = ChatPromptTemplate.from_messages(
20     [
21         ("system", SYS_PROMPT),
22         ("human", """검색된 문서: {document}
23 사용자 질문: {question}"""),
24     ]
25 )
26 # Build grader chain
27 doc_grader = (grade_prompt | structured_llm_grader)
28 print(doc_grader)
29 print(ChatPromptTemplate(input_variables={'document', 'question'}, messages=[SystemMessagePromptTemplate(prompt=grade_prompt, input_variables=
```

<그림 6> 검색결과 판단 구현

```
1 query = "근무 목적 선정시 착안사항 알려줘"
2 topk_docs = similarity_threshold_retriever.invoke(query)
3 for doc in topk_docs:
4     print(doc.page_content)
5     print("-----")
6     print('GRADE:', doc_grader.invoke({'question': query,
7                                       'document': doc.page_content}))
8     print("-----")
```

목적 기준

- 취지
 - 업의 특성에 맞도록 사원 개인의 개성을 고려하되 직장인으로서 품의와 메의를 갖춘 근무목적들 익종도록 목적 기준을 운영함
- 목적 선정시 착안사항
 - (1) 직장인으로서 명예와 품의가 유지되도록 함
 - (2) 항상 단정하고 정결한 목적을 유지해야 함
 - (3) 다수의 객관적인 판단기준으로 목적을 선택함
 - (4) 타인 또는 고객에게 실례가 되지 않도록 목적을 착용해야 함
- 대상
 - (1) 전 임직원

```
GRADE: binary_score='yes'
```

<그림 7> 검색결과 관련성 판단

다음으로는 <그림 8>과 같이 전통적인 RAG 시스템처럼 질의응답 RAG 체인을 구축하여 AI 에이전트와 연계 되도록 구현한다.

구성된 질의응답 RAG 체인을 이용해서 <그림 9>와 같이 연관된 답변 결과를 가져오는 것을 볼 수 있다.

하지만 <그림 10>과 같이 지식에서 벗어난 질문을 시도하면 “RAG에 관련정보가 없습니다”라고 답변 받아 질문에는 답할 수 없다는 것을 확인할 수 있다.

이를 개선하기 위해 사용자 질문을 웹 검색에 최적화된 더 나은 버전으로 다시 표현하는 <그림 11>과 같이

```
1 # Build QA_RAG Chain
2 from langchain_core.prompts import ChatPromptTemplate
3 from langchain_openai import ChatOpenAI
4 from langchain_core.output_parsers import StrOutputParser
5 from operator import itemgetter
6 # 응답 생성을 위한 RAG 프롬프트
7 prompt = """당신은 질의 응답 작업의 Assistant 입니다.
8 다음 검색된 Context 조각을 사용하여 질문에 답을 할것.
9 맥락이 없거나 답을 모르는 경우에는 답을 모른다고 답변할것.
10 제공된 Context에 해당하지 않는 한 답변을 구성하지 마십시오.
11 답변구성시 Context값이 Null이면 'RAG에 관련정보가 없습니다'라고 답변을 할것.
12 질문에 대한 자세하게 요약한 답변을 해주세요.
13
14     Question: {question}
15     Context: {context}
16     Answer:
17     """
17 prompt_template = ChatPromptTemplate.from_template(prompt)
18 # GPT-4-turbo 연결 초기화
19 llm = ChatOpenAI(model_name='gpt-4-turbo', temperature=0)
20 # 새 줄로 컨텍스트 문서를 분리하는 데 사용
21 def format_docs(docs):
22     return "\n\n".join(doc.page_content for doc in docs)
23 # QA RAG chain 생성
24 qa_rag_chain = (
25     {
26         "context": (itemgetter('context')),
27         "question": itemgetter('question')
28     } | prompt_template | llm | StrOutputParser())
```

<그림 8> 질의응답 RAG 체인

```
1 query = "근무 목적 선정시 착안사항 알려줘"
2 topk_docs = similarity_threshold_retriever.invoke(query)
3 result = qa_rag_chain.invoke(
4     [{"context": topk_docs, "question": query}]
5 )
6 print(result)
```

근무 목적을 선정할 때 착안해야 할 사항은 다음과 같습니다:

- 직장인으로서의 명예와 품의 유지: 목적은 직장인으로서의 품위와 예의를 반영해야 하며, 개인의 명예와 단정함과 정결 유지: 목적은 항상 정결하고 정결해야 하며, 관련이 잘 되어 있어야 합니다.
- 객관적인 판단 기준에 따른 선택: 목적 선택 시 다수의 객관적인 기준을 고려하여 결정해야 합니다.
- 타인 또는 고객에게 불쾌감을 주지 않도록 함: 목적은 타인이나 고객에게 불쾌감을 주지 않도록 세심

이러한 사항들은 모든 임직원에게 적용되며, 업무의 특성과 개인의 개성을 존중하면서도 직장인으로서의

<그림 9> 연관된 질문에 대한 답변 처리

```
1 query = "백록담이 있는 산의 높이를 알려줘"
2 topk_docs = similarity_threshold_retriever.invoke(query)
3 print(topk_docs)
4 result = qa_rag_chain.invoke(
5     [{"context": topk_docs, "question": query}]
6 )
7 print(result)
```

```
usr/local/lib/python3.10/dist-packages/langchain_core/vectorstores/base.py:784: UserWarning: Relevance scores
  warnings.warn(
usr/local/lib/python3.10/dist-packages/langchain_core/vectorstores/base.py:796: UserWarning: No relevant docs
  warnings.warn(
RAG에 관련정보가 없습니다.
```

<그림 10> 연관되지 않은 질문에 대한 답변 처리

질문을 재작성(question rewriting)하여 웹에서 질문에 대한 더 나은 컨텍스트 정보를 얻는 데 도움을 받을 수 있다. “백록담이 있는 산의 높이를 알려줘” 처럼 불완전한 질문에 대하여 개선된 질문으로 “백록담이 위치한 한라산의 정확한 높이는 얼마인가요?”와 같이 작성된 것을 확인할 수 있다.

```

1 # 질문 rewrting - LLM : 입력된 사용자 쿼리를 웹 검색에 최적화된 질문으로 재작성
2 llm = ChatOpenAI(model="gpt-4-turbo", temperature=0)
3 # 질문 rewrting - Prompt template
4 SYS_PROMPT = """질문 재작성자 역할을 하고 다음 작업을 수행:
5     - 다음 입력 질문을 웹 검색에 최적화된 더 나은 버전으로 변환하세요.
6     - 다시 작성할 때 입력 질문을 살피고 기본 의미론적 외도/외미에 대해 추론하세요.
7     """
8 re_write_prompt = ChatPromptTemplate.from_messages(
9     [
10         ("system", SYS_PROMPT),
11         ("human", """추가 질문:
12             {question}
13             개선된 질문 작성.
14             """,
15         ),
16     ]
17 )
18 # rephraser 체인 생성
19 question_rewriter = (re_write_prompt | llm | StrOutputParser())

1 query = "백록담이 있는 산의 높이를 알려줘"
2 question_rewriter.invoke({"question": query})
백록담이 위치한 함란산의 높이는 얼마인가요?
    
```

<그림 11> 질문 재 작성 및 개선된 질문

또한, <그림 12>와 같이 생성된 답변이 환각이 있는지 여부를 판단한다.

```

1 ### Implement the hallucination grader
2 from langchain.prompts import PromptTemplate
3 from langchain_core.output_parsers import JsonOutputParser
4 prompt = PromptTemplate(
5     template=""" 환각이 있는지 여부를 평가하는 평가자입니다.
6     환각 여부를 나타내기 위해 이진수 'yes' 또는 'no' 를 부여합니다.
7     환각이 있는 경우 'yes'로 표기하고 없는 경우는 'no'로 표기 하겠.
8     Provide the binary score as a JSON with a
9     single key 'score' and no preamble or explanation.
10    Here are the facts:
11    {documents}
12
13    Here is the answer: {generation} """,
14    input_variables=["generation", "documents"],
15 )
16
17 hallucination_grader = prompt | llm | JsonOutputParser()
18
19 # Run
20 generation = rag_chain.invoke({"context": docs, "question": query})
21 hallucination_grader_response = hallucination_grader.invoke({"documents": docs, "generation": generation})
22 print(hallucination_grader_response)
{"score": "no"}
    
```

<그림 12> 환각 여부 판단

이후, 답변이 질문을 해결하는 데 유용한지 여부를 <그림 13>과 같이 평가하는 단계를 거친다. 답변이 질문을 해결하는 데 유용한지 여부를 나타내기 위해 이진 점수 'yes' 또는 'no'를 부여한다.

```

1 ### Implement the Retrieval Grader : 문서에 사용자 질문과 관련된 키워드가 포함되어 있는 경우
2 prompt = PromptTemplate(
3     template=""" 당신은 관련성을 평가하는 평가자입니다.
4     검색된 문서를 사용자 질문에 연결합니다. 문서에 사용자 질문과 관련된 키워드가 포함되어 있는 경우,
5     관련성 있는 것으로 등급을 매기세요. 일관된 문구는 없습니다. 목표는 잘못된 검색을 필터링하는 것입니다.
6     문서가 질문과 관련이 있는지 여부를 나타내기 위해 이진 수 'yes' 또는 'no' 를 부여합니다.
7
8     Provide the binary score as a JSON with a single key 'score' and no preamble or explanation.
9     {documents}
10
11     Here is the answer: {generation} """,
12     input_variables=["generation", "documents"],
13 )
14 retrieval_grader = prompt | llm | JsonOutputParser()
15
16 # Run
17 result = rag_chain.invoke({"context": docs, "question": query})
18 retrieval_grader_response = retrieval_grader.invoke({"documents": docs, "generation": result})
19 print(retrieval_grader_response)
{"score": "yes"}
    
```

<그림 13> 답변의 적정성 여부 판단

4.2.4. Agent RAG Graph 정의

더 나은 답변을 검색하기 위해 웹 검색에 Tavily API 를 사용하기위해 API에 대한 연결을 로드 한다. 그리고 <그림 14>와 같이 Agent의 그래프 상태를 정의하는데 GraphState()는 그래프의 각 노드에 전달되는 상태 정보를 담은 객체로 각 노드의 상태를 정의하는 역할을 한다.

```

1 # 웹 검색 도구 로드
2 from langchain_community.tools.tavily_search import TavilySearchResults
3 web_search_tool = TavilySearchResults(max_results=2, search_depth='advanced', max_tokens=5000)

1 # Define the Graph State
2 from typing_extensions import TypedDict
3 from typing import List
4
5 ### State
6 class GraphState(TypedDict):
7     """
8     그래프의 상태.
9     Attributes:
10     question: 질문
11     generation: LLM 응답 생성
12     web_search_add: 웹 검색 추가 여부를 나타내는 flag - yes 또는 no
13     documents: 컨텍스트 문서 목록
14     """
15     question: str
16     generation: str
17     web_search_add: str
18     documents: List[str]
    
```

<그림 14> 웹 검색도구 및 Agent Graph State 정의

Agent RAG 그래프는 Retrieve, grade_documents, rewrite_query, web_search_add, generate_answer 등의 노드로 구성할 수 있다. 이러한 노드들은 각각 문서 검색(retrieve), 문서 평가(grade_documents), 질문 재작성(rewrite_query), 웹 검색(web_search_add), 답변 생성(generate_answer)이라는 특정 기능을 수행한다. State는 다양한 노드를 거쳐 전달되는 메시지의 집합으로, 에이전트 그래프의 현재 상태를 나타내는 역할을 한다. <그림 15>에서 보듯이,

```

1 ##### 노드 정의 #####
2 # 벡터스토어에서 문서 검색
3 from langchain.schema import Document
4 def retrieve(state):
5     """
6     벡터스토어에서 문서 검색
7     Args:
8     | state (dict): 현재 그래프 상태
9     Returns:
10    | state (dict): 검색된 문서가 포함된 상태, 문서에 새 키 추가.
11
12    print("---VectorDB에서 검색---")
13    question = state["question"]
14    # Retrieval
15    documents = similarity_threshold_retriever.invoke(question)
16    return {"documents": documents, "question": question}
    
```

<그림 15> Retrieve Node Graph 구현 예

Retrieve 노드는 벡터 데이터베이스에서 관련 컨텍스트 문서를 가져오는 기능을 정의하고 있다. 마찬가지로, 나머지 노드들도 각각의 기능에 맞는 클래스로 정의한다.

4.2.5. Agent RAG Graph 구현

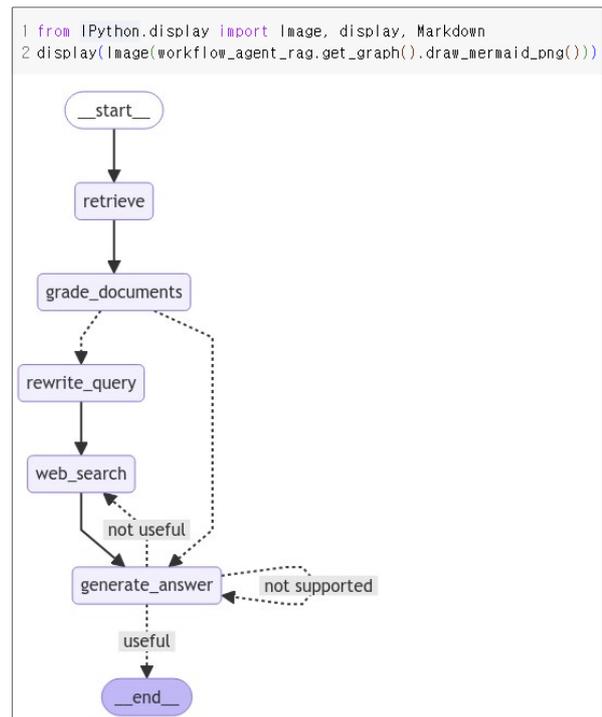
Agent RAG 그래프는 LangGraph를 사용하여 구현하였으며 이전 섹션에서 정의한 함수들을 사용하여 그래프를 구성하고, 각 노드에 필요한 기능을 할당했다. 구체적으로, Retrieve 노드는 문서 검색 기능을 담당하며, 이 노드를 시작점으로 하여 문서 평가, 질의 재작성 등의 다양한 작업을 수행하는 노드들이 순차적으로 연결된다. 이러한 노드들 간의 연결 관계는 정의된 워크플로우에 따라 설정되며 Agent는 Retrieve 노드를 호출하는 action을 수행하고, 그 결과를 State에 추가하여 다음 노드로 전달하는 방식으로 동작한다. <그림 16>에서 보듯이, StateGraph 클래스는 이러한 State 기반 그래프를 관리하고, Agent의 실행 흐름을 제어하는 데 사용된다. 즉, 주어진 코드는 Agent가 문서 검색 작업을 수행하고, 그 결과에 따라 다음 수행할 작업을 결정하는 전체적인 흐름을 정의한다.

```

1 from langgraph.graph import END, StateGraph
2 workflow_agent_rag = StateGraph(GraphState)
3 # Define the nodes
4 workflow_agent_rag.add_node("retrieve", retrieve) # retrieve
5 workflow_agent_rag.add_node("grade_documents", grade_documents) # grade documents
6 workflow_agent_rag.add_node("rewrite_query", rewrite_query) # transform_query
7 workflow_agent_rag.add_node("web_search", web_search) # web search
8 workflow_agent_rag.add_node("generate_answer", generate_answer) # generate answer
9 # Build graph
10 workflow_agent_rag.set_entry_point("retrieve")
11 workflow_agent_rag.add_edge("retrieve", "grade_documents")
12 workflow_agent_rag.add_conditional_edges("grade_documents", decide_to_generate,
13 | | ("rewrite_query": "rewrite_query", "generate_answer": "generate_answer"),)
14 workflow_agent_rag.add_edge("rewrite_query", "web_search")
15 workflow_agent_rag.add_edge("web_search", "generate_answer")
16 workflow_agent_rag.add_conditional_edges(
17 | | "generate_answer",
18 | | grade_generation_v_documents_and_question,
19 | | {
20 | |     "not supported": "generate_answer",
21 | |     "useful": END,
22 | |     "not useful": "web_search",
23 | | },
24 | )
25 # Compile
26 workflow_agent_rag = workflow_agent_rag.compile()
    
```

<그림 16> 답변생성을 위해 조건분기 할 Node정의 및 Graph 생성

Retrieve 노드 이후에는 문서의 평가(grade_documents)를 통해 generate 또는 web_search 작업을 결정하고, 각각의 경로를 따라 최종적으로 작업을 마무리하거나 Agent로 돌아가는 과정을 포함한다. 이 과정은 복잡한 결정과 작업의 흐름을 관리하는 데 사용될 수 있으며 최종적으로 워크플로우를 컴파일하여 실행 가능한 애플리케이션을 생성한다. 이렇게 생성된 워크플로우는 <그림 17>과 같이 IPython.display를 사용하여 시각화할 수 있다.



<그림 17> 답변 생성과정 Graph

5. 테스트 및 결과

구현된 Agent RAG 워크플로우는 다양한 질문으로 테스트하여 답변의 정확도를 높일 수가 있으며 stream 메소드를 통해 질문에 대한 응답의 스트리밍하는 과정을 확인할 수 있다.

5.1. RAG 지식 정보에 있는 질문 검증

RAG 지식 정보에 있는 ‘근무복장기준.pdf’ 문서의 <그림 18>에 대해 “근무 복장 선정 시 착안사항 알려줘” 라는 질문을 입력했을 때 답변의 정확도를 높이기 위해, 질문을 “근무 복장을 선택할 때 고려해야 할 주요 요소는 무엇인가요?”로 수정하여 질의한 결과, 보다 정확한 답변을 얻을 수 있었다. 이 과정은 <그림 19>와 같이 stream 메소드를 통해 질문에 대한 응답을 스트리밍하는 과정을 통해 확인할 수 있었다.

RAG 지식 범위내 질문에 대한 검증 결과는 <표 4>에서 확인할 수 있듯이, 모델은 정확한 답변을 생성하는 것으로 나타났다.

5.2. RAG 지식 정보에 없는 질문 검증

RAG 지식 정보에 없는 질문인 “백록담이 있는 산의 높이를 알려줘” 라고 입력했을 때 “모든 Vector RAG문서는 질문과 관련이 없어 웹 검색을 결정”이라고 결정하여

근무 복장 기준

- 취지**
업의 특성에 맞추어 사원 개인의 계성을 고려하되 직장인으로서 품위와 예의를 갖춘 근무복장을 착용토록 복장 기준을 운영함
- 복장 선정시 착안사항**
(1) 직장인으로서 명예와 품의가 유지되도록 함
(2) 항상 단정하고 청결한 복장을 유지해야 함
(3) 다수의 객관적인 판단기준으로 복장을 선택함
(4) 타인 또는 고객에게 실례가 되지 않도록 복장을 착용해야 함
- 대상**
(1) 전 임직원
(2) 고객사 근무의 경우 고객사 복장 기준을 따름
- 복장 기준**
(1) 단정하고 간소한 비즈니스 캐주얼을 기본으로 하며 청바지, 운동화 가능
(2) 노출이 심하거나 화려한 복장은 피하도록 함
(3) 반바지는 6~9월에 한하여 착용 가능
- 직장인으로서의 예타격을 감안하여, 반바지는 4~5부 길이와, 정장 또는 면 소재의 단정한 디자인을 착용
- 과하게 짧은 바지(일어선 자세에서 허벅지가 노출되는 길이), 트레이닝복, 지나치게 화려한 무늬 등의 반바지는 착용을 지양
- 신발의 경우, 슬리퍼 착용은 불가하며, 로퍼, 운동화 등 반바지와 어울리는 단정한 디자인의 신발을 착용

□ 남자 사원

구분	비즈니스 캐주얼	피해야할 복장
자켓	· 플리셔, 정장 자켓 * 화절기 탈의 가능	· 진(Jean)류 자켓, 점퍼류, 운동복
셔츠	· 드레스셔츠(반팔 셔츠可) · 깃이 있는 캐주얼셔츠, 티셔츠 · 깃 없는 단정한 니트 가능 * 노타이(No-Tie) 가능	· 후드 티, 민소매, 라운드 티 등 · 노출이 심한 복장 · 성적/정치적 공격적인 내용이 프린트된 티셔츠 등
바지	· 정장 바지류 · 단정한 면바지 * 화절기 반바지 가능	· 윗어린 진(Jean)류, 카고 팬츠(견장바지) 등 · 원색계통의 면바지
신발	· 정장 구두류 · 단정한 캐주얼 슈즈	· 샌들, 부츠, 워커, 슬리퍼류
기타	· 벨트 착용 · 양말(옷과 어울리는 색)	· 과도한 장신구, 모자

□ 여자 사원

구분	비즈니스 캐주얼	피해야할 복장
상의	· 면안하게 상·하의를 조합할 수 있는 단정한 복장 (정장 자켓, 블라우스, 등) · 깃 없는 블라우스, 니트 가능	· 진(Jean)류 자켓, 점퍼운동복 · 노출이 심하거나 타이트한 복장(민소매, 탱크 탑 등) · 성적/정치적으로 공격적인 내용이 프린트된 티셔츠 등

<그림 18> ‘근무복장기준.pdf’ 내용

<표 4> RAG 지식정보에 있는 질문-답변

No	최초질문	재작성된 질문	답변
1	여자사원 피해야 할 복장	여자 사원이 피해야 할 복장은 무엇입니까?	여자 사원은 노출이 심하거나 타이트한 복장, 성적/정치적으로 공격적인 내용이 프린트된 티셔츠 등을 피해야 합니다.
2	남자직원 복장 알려줘	남자 사원의 복장 기준은 무엇입니까?	남자 사원은 비즈니스 캐주얼을 기본으로 하며, 정장 바지류와 단정한 면바지를 착용해야 합니다.
3	남자 피해야 할 복장은?	남자 사원이 피해야 할 복장은 무엇입니까?	남자 사원은 너무 캐주얼한 복장이나 지나치게 화려한 복장을 피해야 합니다.
4	복장 기준의 취지	복장 기준의 취지는 무엇인가요?	복장 기준은 직장인으로서 품의와 예의를 갖춘 근무복장을 착용하도록 운영됩니다.
5	반바지 착용 가능 기간이 무엇인가?	반바지는 언제 착용할 수 있습니까?	반바지는 6월부터 9월까지 착용할 수 있습니다.
6	신발 착용 기준이 뭐야?	복장 기준에서 신발 착용에 대한 규정은 무엇입니까?	슬리퍼 착용은 불가하며, 로퍼나 운동화 등 단정한 디자인의 신발을 착용해야 합니다.
7	복장 선정 시 고려사항이 있는지~	복장 선정 시 어떤 사항을 고려해야 하나?	직장인으로서 명예와 품의 유지, 항상 단정하고 청결한 복장 유지 등을 고려해야 합니다.
8	여름철 복장 규정	여름철에 허용되는 복장은 무엇입니까?	여름철에는 하절기 반바지와 단정한 복장이 허용됩니다.
9	복장 기준의 대상 알려줘..	복장 기준의 적용 대상은 누구입니까?	복장 기준은 전 임직원에게 적용됩니다.
10	장신구도 착용 가능해?	복장 기준에서 과도한 장신구에 대한 규정은 무엇입니까?	과도한 장신구 착용은 피해야 하며, 가벼운 악세서리 활용이 권장됩니다.

```

1 query = "근무 복장 선정시 착안사항 알려줘"
2 response = workflow_agent_rag.invoke({"question": query})
3 print("최초 질문 : "+ query)
4 print("재작성된 질문 : "+question_rewriter.invoke({"question": query}))
5 display(Markdown(response['generation']))

---VectorDB에서 검색---
---질문과 문서의 관련성을 체크---
---GRADE: 관련 있는 문서임---
---GRADE: 관련 있는 문서임---
---등급이 매겨진 문서를 평가---
---DECISION: 등급 생성---
---답변 생성---
---CHECK HALLUCINATIONS---
---DECISION: 문서에 기반을 둔 답변 생성---
---GRADE GENERATION vs QUESTION---
---DECISION: 질문에 맞는 답변 생성---
* 최초질문 : 근무 복장 선정시 착안사항 알려줘
* 재작성된 질문 : 개선된 질문:
근무 복장을 선택할 때 고려해야 할 주요 요소는 무엇인가요?
근무 복장을 선정할 때 착안해야 할 사항은 다음과 같습니다.

1. 직장인으로서의 명예와 품의 유지: 근무복장은 직장인으로서의 품위와 예의를 반영해야 하며, 직업의 명예를 유지할 수 있도록 적절해야 합니다.
2. 단정함과 청결 유지: 복장은 항상 깔끔하고 청결해야 하며, 관리가 잘 되어 있어야 합니다.
3. 객관적인 판단 기준에 따른 선택: 복장 선택 시 다수의 객관적인 기준을 고려하여 결정해야 합니다. 이는 주관적인 판단보다는 일반적으로 수용될 수 있는 기준을 따르는 것을 의미합니다.
4. 타인 또는 고객에게 실례가 되지 않도록 함: 복장은 타인이나 고객에게 불편감을 주지 않도록 적절해야 하며, 모든 상황에서 예의를 갖추어야 합니다.

이러한 사항들은 모든 임직원에게 적용되며, 업무의 특성과 개인의 개성을 고려하면서도 직장인으로서의 기본적인 예의와 품위를 유지하는 데 중점을 두어야 합니다.
    
```

<그림 19> RAG 지식에 있는 정보의 답변 생성 과정

‘WEB SEARCH’를 한다. 불안정한 최초 질문인 “백록담이 있는 산의 높이가 알려줘”를 완성된 재 작성된 질문인 “백록담이 위치한 한라산의 높이는 얼마인가요?”로 재 작성한 후 질의해서 “한라산의 해발 고도는 1,947미터입니다. 이는 남한에서 가장 높은 산입니다.”라고 정확한 답변이 나온 결과를 확인할 수가 있다. 이 과정은 <그림

```

1 query = "백록담이 있는 산의 높이를 알려줘"
2 response = workflow_agent_rag.invoke({"question": query})
3 print("최초 질문 : "+ query)
4 print("재작성된 질문 : "+question_rewriter.invoke({"question": query}))
5 display(Markdown(response['generation']))

---VectorDB에서 검색---
/usr/local/lib/python3.10/dist-packages/langchain_core/vectorstores/base.py:784: UserWarning: Relevance scores
warnings.warn(
/usr/local/lib/python3.10/dist-packages/langchain_core/vectorstores/base.py:796: UserWarning: No relevant docs
warnings.warn(
---질문과 문서의 관련성을 체크---
---검색된 문서가 없음---
---등급이 매겨진 문서를 평가---
---DECISION: 모든 Vectore RAG문서는 질문과 관련이 없어 웹 검색을 결정---
---관련 재작성---
---WEB SEARCH---
---답변 생성---
---CHECK HALLUCINATIONS---
---DECISION: 문서에 기반을 둔 답변 생성---
---GRADE GENERATION vs QUESTION---
---DECISION: 질문에 맞는 답변 생성---
* 최초질문 : 백록담이 있는 산의 높이를 알려줘
* 재작성된 질문 : 백록담이 위치한 한라산의 높이는 얼마인가요?
한라산의 해발 고도는 1,947미터입니다. 이는 남한에서 가장 높은 산으로, 정상에는 백록담이 위치해 있습니다.
    
```

<그림 20> RAG 지식에 없는 정보의 웹 검색을 통한 답변 생성 과정

20>과 같이 스트리밍하는 과정을 통해 확인할 수 있다.

RAG 지식에 없는 정보를 웹 검색을 통한 답변 생성하는 과정은 <그림 21>과 같이 실행 내용을 LangSmith를 통해서 추적해보면 설계된 Agent 워크플로우(retrieve, grade_documents, rewrite_query, web_search, generate_answer) 데로 진행된 것을 확인할 수 있다.

RAG 지식 범위를 벗어난 질문에 대한 검증 결과는 <표 5>에 제시되어 있다. 1번부터 5번 질문은 복장 관련 주제지만 RAG에 학습되지 않은 내용을 묻는 질문들이

<그림 21> LangSmith를 통한 Graph 워크플로우 추적

〈표 5〉 RAG 지식정보에 없는 질문-답변

No	최초질문	재작성된 질문	답변
1	결혼식 복장 알려줘	결혼식에 참석할 때 적합한 복장은 무엇 입니까?	남성은 정장이나 블레이저와 슬랙스를, 여성은 원피스나 격식있는 정장차림이 적합합니다. 결혼식 테마나 드레스 코드에 따라 복장이 달라질 수 있습니다.
2	봄에 어울리는 어린이 옷 추천해줘	봄철에 어린이들이 입기에 적합한 옷은 어떤 종류 입니까?	밝은 색상의 가벼운 재킷, 면 소재 티셔츠, 얇은 스웨터, 그리고 편안한 청바지나 면바지가 좋습니다.
3	축구할 때 적당한 신발 알려줘	축구 할 때 착용할 수 있는 적합한 신발은 어떤 것입니까?	축구는 주로 인조 잔디용, 천연 잔디용, 그리고 실내용으로 나뉘며, 각 표면에 맞는 밀착과 쿠션이 있는 것이 좋습니다.
4	등산할 때 입으면 좋은 옷은 뭐야?	등산 시 편안하고 기능적인 옷차림은 무엇입니까?	방수 기능이 있는 재킷, 통기성 좋은 티셔츠, 등산화, 그리고 트레킹 팬츠가 적합합니다.
5	비오는 날 입으면 좋은 옷은?	비가 오는 날씨에 적합한 옷차림은 무엇 입니까?	방수 재킷, 레인부츠, 그리고 우산이 필요합니다.
6	프리미어리그 한국인 주장이 있는 팀 알려줘	프리미어리그에서 한국인 선수가 주장으로 활동하고 있는 팀은 어디 입니까?	토틀넘 홋스퍼 FC입니다.
7	백록담이 있는 산의 높이는?	백록담이 위치한 한라산의 높이는 얼마인가요?	한라산의 해발 고도는 1,947미터입니다. 이는 남한에서 가장 높은 산입니다.
8	세계에서 가장 높은 건물이 있는 나라의 유명한 도시는?	세계에서 가장 높은 부르즈 할리파가 위치한 아랍에미리트의 도시는 어디입니까?	세계에서 가장 높은 건물인 **부르즈 할리파(Burj Khalifa)**는 아랍에미리트의 **두바이(Dubai)**에 위치해 있습니다.
9	유명한 뉴턴이 발견한 법칙 중 하나는 무엇인가요?	뉴턴이 발견한 중력 법칙의 주요 내용은 무엇인가요?	뉴턴의 중력 법칙은 두 물체 간의 인력이 질량에 비례하고 거리의 제곱에 반비례한다는 내용입니다.
10	세계에서 가장 큰 바다의 넓이 알려줘	가장 큰 바다인 태평양의 면적은 얼마나 되나요?	태평양의 면적은 약 168,723,000 평방킬로미터입니다.

며, 6번부터 10번 질문은 완전히 다른 주제의 질문들이 다. 두 경우 모두 RAG 지식정보에 정확한 답변이 없었기 때문에 ‘WEB SEARCH’를 통해 필요한 정보를 찾아 정확한 답변을 제공한 것을 확인할 수 있었다.

과적으로 평가할 수 있음을 확인하였다. 이러한 결과는 RAG 모델의 성능을 개선할 수 있는 효과적인 방법을 제시하며, Graph 기반 접근 방식의 유용성을 입증하였다.

6.2. 시사점

본 연구의 학술적 시사점은 RAG 모델에 Agent와 Advanced RAG의 개념을 도입함으로써 이론적 기초를 확장하고, Graph 기술을 활용한 정보 검색 및 평가 방법의 가능성을 제시했다는 점이다. 이는 기존 문헌에서 다루지 않았던 새로운 연구 방향으로, RAG 관련 연구에서 Graph 기반 접근 방식의 중요성을 강조할 수 있는 기초 자료로 활용될 것이다. 실무적으로는 본 연구에서 구현한 Advanced RAG 시스템이 기업과 기관의 업무 효율성을 크게 높일 수 있는 잠재력을 지니고 있으며, 고객 서비스, 의료, 법률 등 다양한 분야에서 실시간 데이터를

6. 결론 및 논의

6.1. 결론

본 연구는 기존 RAG 모델의 한계를 극복하고자 Graph 기반의 Advanced RAG 시스템을 설계 및 구현하여 지식 기반 질문-답변(QA) 시스템의 성능을 향상시키는 새로운 접근 방식을 제안하였다. 연구 결과, 제안된 시스템은 사용자 질문에 대한 응답의 정확도와 관련성을 크게 향상시켰으며, LangGraph를 활용하여 정보의 신뢰성을 효

바탕으로 신뢰할 수 있는 정보를 제공하여 변화하는 환경에 빠르게 대응할 수 있는 능력을 강화할 수 있다.

예를 들어, 금융 분야에서는 실시간 시장 데이터를 분석하여 투자 의사 결정을 지원하고, 의료 분야에서는 최신 의학 정보를 제공하여 정확한 진단과 치료를 돕는 등 다양한 응용이 가능하다. 또한, 문서 검증 기능을 통해 법률 자문이나 학술 연구에서 필요한 정보의 신뢰성을 높여 더욱 정확한 의사 결정을 지원할 수 있다. 이처럼 본 연구는 고객 만족도를 높이고, 기업의 데이터 품질을 향상시키며, 나아가 기업의 의사 결정 과정을 혁신하는데 기여할 수 있다. 또한 RAG 기반 생성형 AI 기술 발전에 중요한 기반을 제공하며, 관련 분야 연구자와 실무자들에게 유용한 지침이 될 것으로 기대한다.

6.3. 한계 및 향후 연구

본 연구에는 몇 가지 한계점이 존재한다. 첫째, LangGraph 기반 시스템이 특정 도메인만 적용되어 있어 다른 분야에 적용할 경우 성능 저하가 발생할 가능성이 있다. 둘째, 시스템의 복잡성으로 인해 구현 및 유지 관리에 추가적인 자원이 필요할 수 있으며, 이는 실무 적용 시 부담으로 작용할 수 있다. 셋째, 실시간 데이터의 정확성과 신뢰성을 보장하기 위한 추가 검증 과정이 요구되며, 이는 시스템 전체 성능에 영향을 미칠 수 있다. 마지막으로, 그래프 데이터의 처리와 유지 관리에 있어 데이터의 양과 복잡성에 따른 성능 문제도 발생할 수 있다.

향후 연구에서는 Graph 기반 RAG 시스템의 범용성을 높이기 위한 방법을 모색할 필요가 있다. 다양한 도메인에 대한 테스트와 최적화를 통해 시스템의 적용 가능성을 확장하고, 실시간 데이터의 신뢰성을 높이기 위한 알고리즘 개발 및 검증 방법론을 연구하여 시스템 전반적인 성능을 더욱 향상시킬 수 있는 방안을 모색해야 한다. 또한, RAG 기술의 발전은 급속도로 발전하기 때문에 단

순히 기술적인 진보를 따라하기 보다는 어떻게 하면 더 효과적으로 정보를 검색하고, 더 정확하고 신뢰할 수 있는 답변을 생성할 수 있을지에 대한 깊은 이해와 지속적인 개선 노력이 수반되어야 할 것이다. 이는 기술의 발전 뿐 아니라, 실제 응용에서의 신뢰성과 사용 편의성을 높이는 방향으로 발전할 필요가 있다.

〈참고문헌〉

[국내 문헌]

1. 김종철 (2024). RAG 기법을 활용한 LLM 서비스 품질 향상을 위한 데이터 Chunking 전략에 대한 연구. 석사학위논문, 고려대학교, 서울.
2. 박은영 (2024). 고객의 조절초점 성향과 생성형 AI 기반 챗봇에 대한 친숙도가 개인정보 제공의도에 미치는 영향: 프라이버시 계산이론을 중심으로. *지식경영연구*, 25(2), 49-68. <https://doi.org/10.15813/kmr.2024.25.2.003>
3. 안정희, 박해옥 (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>
4. 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>
5. 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>
6. 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>
7. 정천수 (2023c). 전통적인 챗봇과 ChatGPT 연계 서비스 방안 연구. *한국정보기술응용학회지*, 3(4), 11-28. <https://doi.org/10.21219/jitam.2023.30.4.001>
8. 정천수 (2023d). LLM 애플리케이션 아키텍처를 활용한 생성형 AI 서비스 구현: RAG모델과 LangChain 프레임워크 기반. *지능정보연구*, 29(4), 129-164. <https://dx.doi.org/10.13088/jiis.2023.29.4.129>
9. 정천수 (2024). 도메인 특화 LLM: Mistral 7B를 활용한 금융 업무분야 파인튜닝 및 활용 방법. *지능정보연구*, 30(1), 93-120. <https://dx.doi.org/10.13088/jiis.2024.30.1.093>
10. 전준형, 김상철, 김주철, 윤성준 (2024, 1월 31일). 웹 애플리케이션 서버(WAS)에서의 검색 증강 생성(RAG) 기술을 이용한 지식 기반 QA 문제 해결. *한국통신학회 학술대회논문집*.
11. Adam, M., Wessel, M., & Benlian, A. (2021). AI-based Chatbots in customer service and their effects on user compliance. *Electronic Markets*, 31(2), 427-445.
12. Angels, B., Vinamra, B., Renato, L., et al. (2024, January 30). RAG vs Fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*.
13. Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2023, October 17). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.
14. Jeong, C. S. (2023). A study on the implementation of generative ai services using an enterprise data-based LLM application architecture. *Advances in Artificial Intelligence and Machine Learning*, 3(4), 1588-1618. <https://dx.doi.org/10.54364/AAIML.2023.1191>
15. Jeong, S., Baek, J., Cho, S., Hwang, S. J., & Park, J. C. (2024, March 28). Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.
16. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.
17. Przegalinska, A., Ciechanowski, L., Stroz, A., Gloor, P., & Mazurek, G. (2019). In bot we trust: A new methodology of Chatbot performance measures. *Business Horizons*, 62(6), 785-797.
18. 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
19. Scott, B., Stefanus, K., Srikanth, T., Zach, B., & Mohamed, A. (2024). Seven failure points when engineering a retrieval augmented generation system. *2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI (CAIN)*, 194-199. <https://doi.org/10.1145/3644815.3644945>
20. Yan, S. Q., Gu, J. C., Zhu, Y., & Ling, Z. H. (2024, March 27). Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.
21. Yunfan, G., Yun, X., Xinyu, G., Kangxiang, J., Jinliu, P., Yuxi, B., Yi, D., Jiawei, S., Meng, W., & Haofen,

[국외 문헌]

W. (2024, March 27). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[URL]

22. 스킨터랩스 (2024). <https://www.skelterlabs.com/blog/2024-year-of-the-rag>
23. 장동진 (2024). <https://aws.amazon.com/ko/blogs/tech/korean-reranker-rag/>
24. Devtorium (2023). <https://towardsdatascience.com/10-ways-to-improve-the-performance-of-retrieval-augmented-generation-systems-5fa2cee7cd5c>
25. LangGraph (2024). <https://langchain-ai.github.io/langgraph/>
26. Matt A. (2023). <https://towardsdatascience.com/10-ways-to-improve-the-performance-of-retrieval-augmented-generation-systems-5fa2cee7cd5c>
27. Microsoft (2023). <https://learn.microsoft.com/en-us/azure/machine-learning/concept-retrieval-augmented-generation?view=azureml-api-2>
28. Wikipedia (2016). <http://www.wikipedia.org/>

● **저 자 소 개** ●



정 천 수 (Cheonsu Jeong)

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, 지능정보연구, Information Systems Review, AAIML 등을 포함한 다수의 국내.외 저널에 논문을 게재한 바 있으며 주요 관심분야는 Generative AI, LLM, RAG, Hyperautomation, Digital Transformation, Conversational AI 등이다.

〈 Abstract 〉

A Graph-Agent-Based Approach to Enhancing Knowledge-Based QA with Advanced RAG

Cheonsu Jeong^{*}

This research aims to develop high-quality generative AI services by overcoming the limitations of existing Retrieval-Augmented Generation (RAG) models and implementing an enhanced graph-based RAG system to improve knowledge-based question answering (QA) systems. While traditional RAG models demonstrate high accuracy and fluency by utilizing retrieved information, their accuracy can be compromised due to the use of pre-loaded knowledge without rework. Additionally, the inability to incorporate real-time data after the RAG configuration leads to a lack of contextual understanding and potential biased information. To address these limitations, this study implements an enhanced RAG system utilizing graph technology. This system is designed to efficiently search and utilize information. In particular, LangGraph is employed to evaluate the reliability of retrieved information and to generate more accurate and improved answers by integrating various information. Furthermore, the specific operation method, key implementation steps, and case studies are presented with implementation code and verification results to enhance understanding of Advanced RAG technology. This research provides practical guidelines for actively implementing enterprise services utilizing Advanced RAG, making it significant.

Key words: Advance RAG, Agent RAG, Knowledge retrieval, Generative AI, LangGraph

^{*} SAMSUNG SDS AI Automation Team