학사학위 논문

LangChain을 활용한 뉴스 QA RAG 설계 및 프롬프트 기법 분석

인하대학교 전기전자공학부

김선우

2025 년 06 월

차 례

5 **결론 및 향후 연구**

5.1 연구 요약 및 시사점

5.2 연구의 한계

5.3 향후 연구 방향 제언

초록

본 연구는 대규모 언어 모델(LLM) 기반 챗봇에서 발생하는 '환각(Hallucination)' 현상을 줄이기 위해, 외부 지식 검색과 생성 모델을 결합한 검색 증강 생성(RAG: Retrieval-Augmented Generation) 시스템의 성능 최적화를 목표로 한다. 특히, RAG 시스템의 답변 품질을 좌우하는 프롬프트 엔지니어링(prompt engineering) 기법에 주목하여, '기본(Basic)', 'Few-shot', '사고의 연쇄(Cot: Chain-of-Thought)', '자가 교정(Self-Correction)'의 네 가지 전략을 설계·적용하고, 정확성, 관련성, 정보 포함율 등 다섯 가지 지표를 활용해 정량적 비교를 수행하였다.

실험 결과, CoT 전략은 정확도와 관련성 면에서 유의미한 향상을 보였으나, 핵심 정보 포함율이 다소 하락하는 트레이드오프(trade-off)를 나타냈다. 반면, Self-Correction 전략은 이 같은 단점을 극복하여 모든 지표에서 최고 성능을 달성함을 확인하였다. 본 연구는 '생성-비판-수정(generate-critique-revise)'의 2단계 접근 방식을 통해 RAG 시스템의 신뢰성과 완성도를 동시에 높일 수 있음을 실증함으로써, 향후 고신뢰성 챗봇 설계에 유용한 실무 지침을 제공한다.

abstract

Reliability remains a major challenge for large language model (LLM)-based chatbots, as they often generate plausible yet incorrect information ("hallucinations"). This study investigates how prompt engineering strategies influence the performance of Retrieval-Augmented Generation (RAG) systems in mitigating hallucinations. We implement and evaluate four prompt paradigms—Basic, Few-shot, Chain-of-Thought (CoT), and Self-Correction—within an Upstage Solar + ChromaDB + LangChain framework. Using metrics such as Correctness, Relevance, and Information Inclusion Rate, we conduct a quantitative comparison over 30 test cases. Results show that CoT improves correctness and relevance but incurs a trade-off by reducing inclusion rates. In contrast, the Self-Correction approach, which employs a generate-critique-revise cycle, achieves superior performance across all metrics. These findings offer practical guidelines for designing high-fidelity RAG systems in domains requiring reliable responses.

1 서론

1.1 연구배경 및 목적

최근 대규모 언어 모델(LLM)은 방대한 선학습 데이터를 바탕으로 언어 이해 및 생성 능력이 크게 향상되어, 요약, 번역, 질의응답(Q&A) 등 다양한 NLP 과업에서 뛰어난 성능을 보이고 있다. 하지만 LLM이 생성하는 텍스트는 때때로 사실과 무관한 내용을 그럴듯하게 제시하는 환각(hallucination) 현상을 포함할 수 있다. 이는 특히 의료·법률·금융 등 정확성이 중요한 분야의 실용화에 큰 걸림돌로 작용한다.

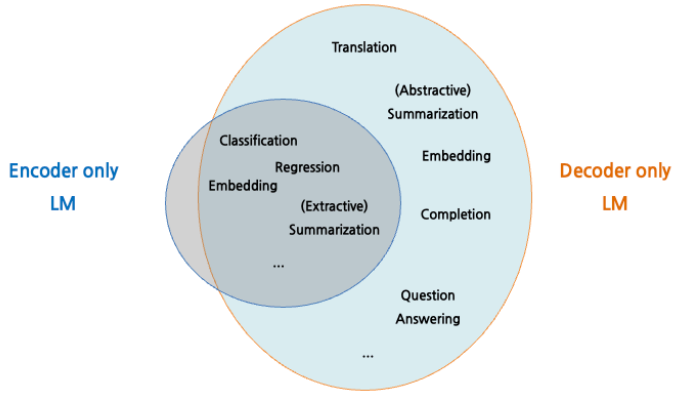
이 문제를 해결하기 위한 대표적 접근법이 검색 증강 생성(RAG) 이다. RAG는 외부 지식 소스를 실시간으로 검색하여 LLM 입력에 포함함으로써, 모델이 사실 정보를 근거로 응답하도록 유도한다. 그러나 단순히 RAG를 도입하는 것만으로 환각 문제가 완전히 해소되는 것은 아니다. 검색된 정보를 효과적으로 가공하고 모델에 제시하는 방식, 즉 프롬프트 엔지니어링이 최종 답변 품질을 좌우하기 때문이다.

본 연구는 “어떤 프롬프트 전략이 RAG 시스템의 신뢰성과 완전성을 최적화하는가?”라는 질문에 답하기 위해, Basic, Few-shot, CoT, Self-Correction 네 가지 기법을 LangChain 기반 RAG 시스템에 적용하고, 다양한 평가 지표를 통해 성능을 비교·분석한다.

2. 이론적 배경

2.1 대규모 언어 모델(LLM) 구조

대규모 언어 모델(Large Language Model, LLM)은 주로 Transformer 구조를 기반으로 하며, 입력 문맥을 바탕으로 다음 단어를 예측하거나 특정 질의에 대한 적절한 응답을 생성하는 방식으로 동작한다.

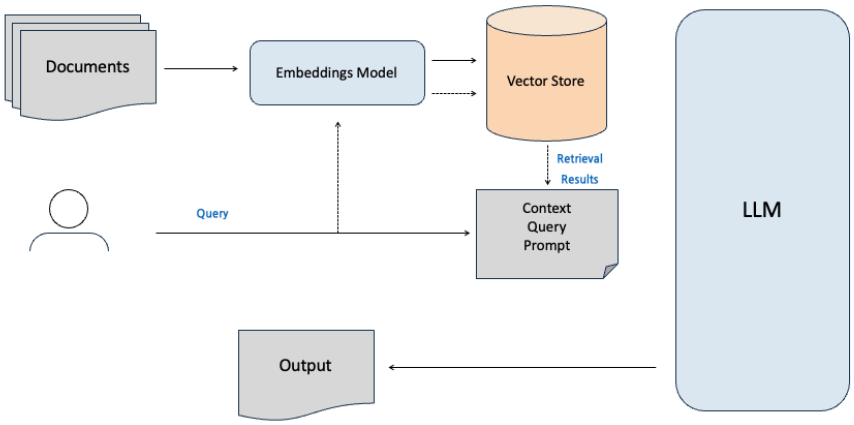


[그림 1. 인코더와 디코더 모델의 수행 작업 비교]

LLM은 내부 구조에 따라 크게 Encoder-only, Decoder-only, Encoder-Decoder 세 가지로 나뉜다. 그림 1은 각 구조의 특성과 해당 구조가 강점을 보이는 대표적인 자연어 처리 작업들을 보여준다. 그림 1의 좌측 원에 해당하는 Encoder-only 모델은 입력 전체를 동시에 인코딩하여 문맥을 이해하는데 최적화되어 있으며 BERT 계열 모델이 대표적이며, 이 구조는 문장 전체의 의미를 벡터로 표현하는 임베딩(Embedding)이나 문서를 특정 카테고리로 분류하는 분류(Classification)와 같은 이해(Understanding) 기반 작업에 뛰어난 성능을 보인다. 반면, 그림 1의 우측 원에 속하는 Decoder-only모델은 GPT 계열과 같이 이전 시퀀스를 바탕으로 다음 단어를 순차적으로 예측하는 방식으로 동작한다. 이러한 특성 덕분에 주어진 문맥에 이어 자연스러운 텍스트를 만들어내는 완성(Completion)이나 사용자의 질문에 답변하는 질의응답(Question Answering)과 같은 생성(Generation) 작업에 주로 활용된다. 본 연구의 RAG 시스템 역시 최종적으로 사용자의 질문에 대한 답변을 '생성'해야 하므로, Decoder-only 구조를 채택한 한국어 특화 LLM인 Solar LLM을 활용한다.

마지막으로 인코더-디코더 구조는 입력 문장을 인코더로 완벽히 이해한 후, 그 정보를 바탕으로 디코더가 완전히 새로운 출력 시퀀스를 생성하는 방식이다. BART나 T5 모델이 이에 속하며, 소스 언어를 타겟 언어로 바꾸는 번역이나 원문을 요약하여 새로운 문장을 만드는 요약과 같이 입력과 출력이 명확히 구분되는 작업에 강점을 가진다.

2.2 검색 증강 생성(RAG) 구조



[그림 2. RAG의 구조]

RAG는 언어 모델에 외부 지식 기반(knowledge source)을 결합하여 모델이 답변을 생성할 때 관련 문서를 실시간으로 검색하고 이를 LLM의 입력으로 포함시켜 답변을 생성하는 구조이다. [그림2]는 RAG 구조의 전체 흐름을 도식화한 것으로 크게 '인덱싱' 단계와 '검색 및 생성' 단계로 나눌 수 있다. 먼저, 오프라인으로 수행되는 인덱싱 단계에서는 사전에 수집한 다수의 문서를 처리한다. 각 문서는 임베딩 모델을 통해 의미를 함축한 고차원 벡터로 변환되며, 이 벡터들은 빠른 검색이 가능하도록 벡터 저장소에 저장 및 색인된다. 이 과정은 사용자가 질문하기 전에 미리 데이터베이스를 구축해두는 준비 단계에 해당된다. 다음으로, 사용자가 질문을 입력하면 실시간으로 검색 및 생성(Retrieval and Generation) 단계가 시작된다.

사용자 질의(Query)는 인덱싱 단계에서 사용된 것과 동일한 임베딩 모델을 거쳐 쿼리 벡터로 변환된다. 시스템은 이 쿼리 벡터를 사용하여 벡터 저장소 내에서 의미적으로 유사한 문서 벡터들을 탐색하고, 그 결과를 검색 결과로 반환한다. 이 검색 결과는 사용자의 원본 질의와 함께 프롬프트로 재구성된다. 마지막으로, 근거 문서와 질문이 모두 포함된 프롬프트가 최종적으로 LLM에 전달된다. LLM은 주어진 컨텍스트를 바탕으로 최종결과(Output)를 생성하여 사용자에게 답변한다.

2.3 프롬프트 엔지니어링 기법

프롬프트 엔지니어링(Prompt Engineering)은 대규모 언어 모델(LLM)의 출력을 원하는 방식으로 유도하기 위해 모델에게 주어지는 입력(Prompt)의 구조와 표현 방식을 설계하는 기법이다. 이는 언어 모델의 응답 정확도, 응답 형식, 추론 과정, 정보 충실도 등에 직접적인 영향을 미치며, 특히 RAG(Retrieval-Augmented Generation) 구조에서는 검색된 외부 정보와 결합된 프롬프트 구성 방식이 응답의 품질을 결정하는 핵심 요소로 작용한다.

본 연구에서는 뉴스 기사 기반 질의응답 시스템의 성능을 향상시키기 위해, 다음과 같은 네 가지 프롬프트 유형을 실험적으로 적용하고 그 효과를 비교 분석하였다.

(1) Zero-shot Prompting

Zero-shot Prompting은 LLM에게 별도의 예시나 구체적인 추론 경로를 제공하지 않고, 오직 수행할 작업에 대한 지시사항(instruction)과 맥락(context)만을 전달하여 응답을 생성하게 하는 방식이다. 이는 모델이 사전 학습을 통해 내재화한 방대한

지식과 일반화 능력을 기반으로 작업을 수행할 것을 전제로 한다. 간단하고 빠르게 적용 가능한 기법으로, 프롬프트 구성이 복잡하지 않아 비용 효율적이다. 모델의 기본적인 성능을 측정하는 기준점으로 활용된다.

본 연구에서 구현한 챗봇의 가장 기본적인 프롬프트로, 검색된 뉴스 기사와 사용자 질문을 제공하며 역할을 정의하는 최소한의 지시사항만을 포함한다. 이 기법을 비교 실험의 기준선으로 설정하고, 이후에 적용될 다른 프롬프팅 기법들이 얼마나 성능 향상을 가져오는지 측정하는 척도로 사용한다.

(2) Few-shot Prompting

Few-shot Prompting은 모델에게 몇 개의 완성도 높은 질문-응답 예시(shot)을 프롬프트 내에 함께 제공하는 기법이다. LLM은 주어진 예시들의 패턴을 학습하여(In-context Learning), 새로운 질문에 대해서도 유사한 구조와 품질의 답변을 생성하게 된다. 이는 모델을 직접 미세조정하지 않고도 원하는 결과물로 유도할 수 있는 방법이다. 답변의 스타일, 어조, 길이 등 구체적인 출력 형식을 일관성 있게 제어하는 데 매우 효과적이다. 복잡하거나 모호할 수 있는 지시사항을 구체적인 예시로 보여줌으로써 모델의 작업 이해도를 높여 응답의 정확성을 향상시킨다.

본 연구에서는 뉴스 기사를 바탕으로 답변하는 양식을 구체적으로 지정하여 프롬프트에 포함시켰다. 이를 통해 모델이 단순히 정보를 나열하는 것을 넘어, 기사 내용을 어떻게 자연스러운 문장으로 요약하고 종합하는지를 학습하도록 유도하였다. 이 기법이 답변의 일관성과 완성도를 얼마나 향상시키는지를 검증하는 것을 목표로 한다.

(3) Chain-of-Thought(CoT) Prompting

CoT는 LLM이 최종 답변을 내놓기 전에, 문제 해결을 위한 중간 추론 과정을 단계별로 명시적으로 생성하도록 유도하는 기법이다. “단계별로 생각하라”와 같은 지시를 통해 복잡한 문제를 작은 단위로 분해하고 논리적 순서에 따라 해결하도록 만든다. 여러 단계의 추론이나 계산이 필요한 질문에 대해 정확도를 크게 향상시킨다. RAG 시스템에서는 LLM이 검색된 context의 내용을 맹목적으로 인용하는 것이 아니라, 어떤 정보가 질문과 관련 있는지 비판적으로 분석하고 종합하는 과정을 거치게 하여 환각 현상을 줄인다.

본 연구에서는 뉴스 기사에서 분산된 정보를 종합해야 하는 복잡한 질문에 대응하

기 위해 CoT를 도입했다. 프롬프트에 사실 확인 단계, 종합 단계를 명시하여 모델이 먼저 근거가 되는 핵심 사실들을 문서에서 추출한 뒤, 이를 바탕으로 최종 답변을 구성하도록 설계하였다. 이 접근법이 정보의 누락이나 왜곡 없이 논리적으로 정제된 답변을 생성하는데 얼마나 효과적인지 평가하고자 한다.

(4) Self-Correction

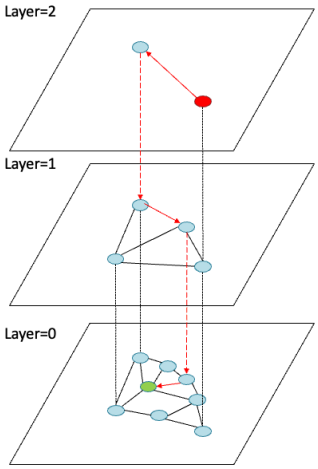
자가 수정(Self-Correction)은 LLM이 최종 답변을 내놓기 전에, 스스로 생성한 초기 결과물(초안)을 비판적으로 검토하고 개선하는 과정을 거치도록 유도하는 기법이다. "스스로의 답변을 주어진 기준에 따라 평가하고 수정하라"와 같은 지시를 통해, 모델이 생성, 비판, 수정을 포함하는 다단계 추론 프로세스를 수행하도록 만든다. 이 방식은 복잡하거나 다면적인 정보에 대해 성급한 결론을 내리는 것을 방지하고, 최종 답변의 사실적 정확성과 논리적 완결성을 크게 향상시킨다.

RAG 시스템에서는 LLM이 검색된 context의 내용을 단순히 요약하는 것을 넘어, 생성된 답변이 문서의 내용에 완전히 부합하는지를 스스로 검증하는 과정을 거치게 하여 환각 현상을 줄인다. 모델이 자신의 답변을 문서 내용과 비교하며 자체적으로 사실을 확인하고 오류를 수정하게 함으로써, 최종 답변의 신뢰도를 높이는 데 매우 효과적이다.

2.4 벡터 검색 기술

임베딩(Embedding)을 통해 텍스트가 고차원의 벡터로 표현되면, 다음 과제는 대규모 벡터 집합 내에서 주어진 쿼리 벡터와 가장 유사한 벡터를 효율적으로 찾는 것이다. 가장 간단한 방법은 저장된 모든 벡터와 일일이 유사도를 계산하는 완전 탐색(Full Search 또는 Brute-force) 방식이지만, 데이터의 양이 수만 개를 넘어서면 검색에 막대한 시간과 연산 비용이 소요되어 실시간 서비스에 적용하기 어렵다.

이러한 한계를 극복하기 위해 근사 최근접 이웃(ANN, Approximate Nearest Neighbor) 탐색 알고리즘이 널리 사용된다. ANN은 정확도를 약간 희생하는 대신, 검색 속도를 획기적으로 향상시키는 방식이다. 다양한 ANN 알고리즘 중에서도 HNSW(Hierarchical Navigable Small World)는 뛰어난 성능과 안정성으로 인해 널리 사용되는 알고리즘이다.



[그림 3. Hierarchical Navigable Small World 알고리즘 동작 방식]

[그림 3]은 HNSW 알고리즘의 동작 방식을 도식화한 것이다. HNSW는 데이터 벡터들을 여러 계층(Layer)으로 구성된 그래프 구조로 인덱싱한다. 각 데이터 벡터는 가장 밀도가 높은 최하위 계층(Layer=0)에 모두 존재하며, 상위 계층으로 갈수록 더 적은 수의 '탐색용 노드'가 확률적으로 선택되어 계층적 그래프 구조를 형성한다.

검색은 그림 3의 붉은 점(Query)과 같이, 가장 추상화된 최상위 계층(Layer=2)의 진입점(entry point)에서 시작된다. 여기서 쿼리와 가장 가까운 노드를 찾은 후, 그 노드를 기준으로 한 단계 아래의 더 조밀한 계층(Layer=1)으로 이동하여 다시 지역적인 탐색을 수행한다. 이 과정을 최하위 계층(Layer=0)까지 반복하며 점차 탐색 범위를 좁혀, 최종적으로 쿼리와 가장 유사한 초록색 점(최근접 이웃)을 찾아 반환한다. 이러한 계층적 탐색 방식은 대규모 벡터 집합을 모두 비교하는 완전 탐색을 피하고, 유사도 높은 후보군으로 빠르게 수렴할 수 있게 한다. 덕분에 수백만 개 이상의 벡터가 저장된 환경에서도 실시간에 가까운 검색 속도를 확보할 수 있다.

3. 시스템 설계 및 구현

본 연구는 기술 뉴스 데이터를 활용하여 RAG 기반 QA 시스템을 설계하고 프롬프트 기법 효과를 분석하는 것에 중점을 둔다. 구체적으로 갖는 범위는 다음과 같다.

3.1 데이터 수집 및 전처리

3.1.1 기사 데이터 수집

본 연구에서는 2022년부터 2025년까지 한국어로 작성된 기술 뉴스 데이터를 웹

크롤링을 통하여 수집하였다. 크롤링 대상은 ZDNet, ITWorld 등 국내외 기술 전문 뉴스 플랫폼에 한정하였다. ZDNet에서는 '인터넷', '컴퓨팅', '방송/통신' 카테고리, ITWorld에서는 '클라우드 컴퓨팅', '인공지능', '소프트웨어 개발' 카테고리에 속하는 뉴스 기사만을 선별적으로 수집하였다. 최종적으로 총 3,383개의 기사 본문과 메타데이터(제목, 작성일, URL 등)를 확보하였다. 수집된 기사 분포는 표1과 같다.

플랫폼	카테고리	기사 수
ZDNet	인터넷	720
	컴퓨팅	785
	방송/통신	602
ITWorld	클라우드 컴퓨팅	590
	인공지능	578
	소프트웨어 개발	110
	합계	3,385

[표 1. 수집된 뉴스 기사 분포]

기사 데이터를 수집하는 과정에서 본문 내용 외에 제목, 작성일, 작성자, 원문 링크, 카테고리, 본문, 플랫폼 등의 메타데이터 정보를 함께 수집하여 저장하였다.

3.1.2 청크(Chunk) 분할 및 임베딩

크롤링을 통해 수집된 뉴스 데이터는 효율적인 처리와 검색 정확도 향상을 위해 일정한 단위로 분할(Chunking)하는 과정을 거친다. 대규모 언어 모델(LLM)은 일반적으로 처리할 수 있는 입력 토큰(token) 수에 제약이 있으므로, 원문 기사 데이터를 그대로 활용할 경우 입력 길이 제한으로 인해 처리가 불가능하거나 응답 품질이 저하될 수 있다. 이러한 문제를 해결하고 의미 단위의 효과적인 검색을 지원하기 위해, 본 연구에서는 기사 본문을 문장 또는 문단 단위로 분할하여 사용하였다.

지표	값
원본 문서 수	3385개
총 청크(chunk) 수	6774개
임베딩 모델	embedding passage
임베딩 벡터 차원	1024
평균 청크 당 토큰 수	778 토큰
유사도 측정 방식	코사인 유사도(Cosine Similarity)

[표 2. 임베딩 데이터셋 통계]

본 연구에서는 최대 1000 토큰을 기준으로 기사 본문을 의미 단위의 청크로 분할

하였으며, 100 토큰의 중첩(overlap) 구간을 두어 문맥 정보가 청크 간에 유지되도록 설계하였다.분할된 각 청크(Chunk)에는 원본 기사의 URL과 해당 청크의 순번을 조합한 고유 식별자를 부여하여 추적 및 관리가 용이하도록 구성하였다.

분할 과정을 거쳐 생성된 각 청크는 검색 효율성을 극대화하기 위해 벡터 형태로 임베딩(Embedding)된다. 임베딩은 텍스트 데이터로 고정된 차원의 실수 벡터로 변환하는 과정으로, 이를 통해 문서 간의 의미적 유사도를 정량적으로 계산할 수 있게 된다. 본 연구에서는 한국어 특화 임베딩 모델인 Upstage의 embedding-passage 모델을 활용하여 각 텍스트 청크를 1024차원의 벡터로 변환하였다. 해당 모델은 다양한 도메인의 대규모 한국어 문서를 사전 학습되어, 문장 간의 의미적 유사도를 효과적으로 포착하도록 설계된 트랜스포머(Transformer) 인코더 기반의 임베딩 모델이다.

이렇게 벡터화된 표현은 질의(Query) 벡터와 문서(Chunk) 벡터 간의 유사도를 정량적으로 비교하는 데 활용된다. 검색 과정에서는 코사인 유사도(Cosine Similarity)를 측정하여 질의 벡터와 가장 유사한 의미를 갖는 문서 벡터를 탐색하고 해당 문서를 반환한다. 이는 질의 벡터 $\vec{A}$와 문서 벡터 $\vec{B}$ 간의 방향 유사성을 정량화하는 방식이다. 벡터 성분 단위의 수식은 다음과 같다.

$$cos(\theta) = (\sum(i=1)^{n} A_i B_i) / \left(\sqrt{\left(\sum(i=1)^{n} A_i^2\right)} \cdot \sqrt{\left(\sum(i=1)^{n} B_i^2\right)} \right) \quad (1)$$

여기서 A_i, B_i 는 각 벡터의 i번째 성분이며, n은 임베딩 벡터의 차원(1024)이다. 코사인 유사도 값은 [-1,1] 범위이며, 1에 가까울수록 의미적으로 유사한 벡터로 간주된다. 특히 임베딩과 같이 수천 차원 이상의 고차원 벡터 공간에서는, 벡터의 크기보다 방향이 의미론적 유사도를 더 잘 나타낸다. 이 방식은 문서의 길이나 단어 수와 같은 벡터의 크기 요인을 배제하고 순수하게 의미의 방향성만 비교할 수 있어 텍스트 검색 작업에 가장 널리 사용된다.

3.2 벡터 데이터베이스 구축

3.2.1 벡터 저장소

본 연구에서는 뉴스 기사의 임베딩 결과를 저장하고 검색하기 위한 벡터 데이터베이스로 오픈소스 프레임워크인 ChromaDB를 활용하였다. ChromaDB는 Python 기

반의 경량 벡터 저장소로, 별도의 서버 구축 없이 로컬 파일 시스템에 데이터를 영속적으로 저장할 수 있어 실험의 재현성을 확보하는 데 용이하다. 임베딩된 각 문서 Chunk는 고유 ID와 함께 ChromaDB의 '컬렉션(Collection)' 단위로 관리되며, 컬렉션 내부에서는 documents(원본 텍스트), embeddings(벡터), metadatas(부가정보), ids 필드 단위로 데이터가 체계적으로 저장된다.

3.2.2 벡터 인덱싱 및 검색

효율적인 검색을 위해, 본 연구에서는 2.4절에서 설명한 HNSW(Hierarchical Navigable Small World) 알고리즘을 내부적으로 사용하는 ChromaDB를 채택하였다. 이는 앞서 3.1.2절에서 생성된 6,774개의 벡터에 대한 근사 최근접 이웃(ANN) 인덱스를 자동으로 생성하여, 모든 벡터를 일일이 비교하는 완전 탐색(Brute-force) 방식의 비효율성을 개선한다. 이를 통해 수천 개의 벡터가 저장된 로컬 환경에서도 실시간에 가까운 검색 속도를 확보할 수 있었다.

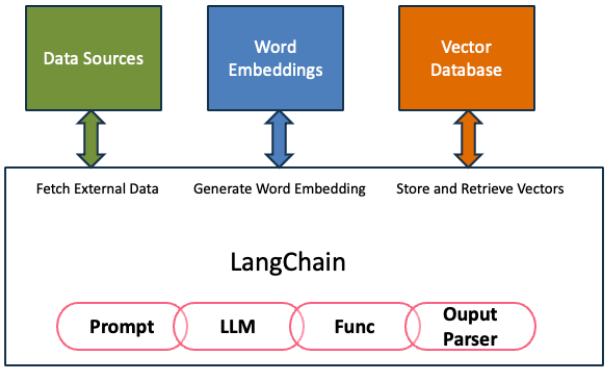
벡터 간의 유사도 측정 방식으로는 3.1.2절에서 상술한 코사인 유사도(Cosine Similarity)가 사용되었다. 이는 ChromaDB의 기본 유사도 측정 방식으로, 텍스트 벡터의 의미론적 유사도를 비교하는 데 가장 적합하다. 더불어, 각 Chunk를 벡터와 함께 JSON 형식의 메타데이터(원본 문서 URL, 제목 등)와 결합하여 저장하는 구조는, 향후 특정 출처나 날짜의 문서만을 대상으로 검색 범위를 한정하는 '메타데이터 필터링'과 같은 고도화된 검색 전략으로의 확장 가능성을 열어둔다.

3.3 RAG 파이프라인 및 프롬프트 설계

3.3.1 LangChain 기반 파이프라인

본 연구의 RAG 파이프라인은 LangChain 프레임워크를 기반으로 구성되었다. LangChain은 [그림4]와 같이, RAG의 각 구성 요소를 독립적인 모듈로 정의하고 이들을 체인(Chain)으로 엮어 하나의 파이프라인으로 동작시키는 구조를 제공한다. 그림에서 볼 수 있듯, LangChian은 데이터 소스(Data Sources)에서 외부 데이터를 가죠오고, 임베딩 모델(Word Embeddings)을 통해 텍스트를 벡터로 변환하며, 이 벡터를 벡터 데이터베이스(Vector Database)에 저장하고 검색하는 등 RAG의 전체 과정을 조율하는 역할을 수행한다. 본 연구에서는 이러한 LangChain의 기능을 활

용하여, 대화 맥락을 관리하는 부분과 실제 문서를 기반으로 답변을 생성하는 부분을 논리적으로 분리한 두 단계의 파이프라인을 구축하였다. 먼저 파이프라인의 첫 단계에서는 대화 맥락 인지 리트리버(create history aware retriever)를 통해 사용자의 현재 질문(input)과 이전 대화기록을 함께 처리한다. 이 과정은 대화의 맥락을 이해하여, 후속 검색에 용이하도록 독립적인 질문을 생성하는 역할을 수행한다. 예를 들어 "그 모델의 장점은 뭐야?"라는 후속 질문이 들어오면, "Solar 10.7B 모델의 장점은 무엇인가?"와 같이 명확한 검색용 질문으로 재작성 한다. 이렇게 생성된 검색용 질문은 Retriever로 전달되어 관련 문서를 찾고, 이후 검색 증강 체인(create retrieval chain)이 원본 질문과 검색된 문서(context)를 조합하여 최종 프롬프트를 완성한다. 완성된 프롬프트는 Solar LLM에 전달한다. 최종적으로 LLM은 이 프롬프트를 기반으로 답변을 생성하고, 출력 파서(Output Parser)를 통해 정제된 결과를 반환한다. 이러한 모듈식 구조는 프롬프트 설정 변경, Retriever 교체 등 각 구성 요소의 독립적인 실험을 용이하게 하였다. 이는 복잡한 하이퍼파라미터 튜닝 없이도 안정적인 QA 시스템의 구축을 가능하게 하여 재현성과 확장성 측면에서 실용적이었다.



[그림 4 LangChain 흐름도 및 구조]

3.3.2 프롬프트 전략 설계

본 연구에서는 2.3절에서 설명한 프롬프트 엔지니어링 기법들을 뉴스 기사 기반 질의응답 시스템의 목적에 맞게 구체화하여 네 가지 전략으로 설계하였다. 각 프롬프트는 LLM이 검색된 근거 문서(context)에만 충실하게 답변하고, 환각(Hallucination)을 최소화하도록 명시적인 제약 조건을 포함하여 개발되었다.

(1) Zero-shot Prompting 적용(기본 프롬프트)

가장 기본적인 성능을 측정하기 위한 기준으로, 별도의 예시나 복잡한 추론 과정 없이 간결한 지시사항만으로 구성하였다. "당신은 뉴스 기사 기반 Q&A 어시스턴트입니다. 아래 context에서 질문과 관련된 정보가 있으면, 그 내용을 바탕으로 답변하세요." 라는 직접적인 지시를 통해, 모델이 추가 정보 없이 검색된 내용만을 요약하여 답변하도록 유도했다. 이 프롬프트는 다른 프롬프트 기법들의 성능 향상폭을 측정하는 베이스라인(Baseline) 역할을 수행한다.

(2) Few-shot Prompting 적용

LLM이 생성해야 할 답변의 형식과 스타일을 명확히 제시하기 위해, 완성도 높은 질의응답 예시(shot)를 프롬프트에 포함하였다. Few-shot 기법은 일반적으로 여러 예시를 제공하지만, 본 연구에서는 프롬프트의 간결성과 토큰 효율성을 고려하여 가장 이상적인 답변의 구조를 보여줄 수 있는 단일 예시를 채택했다. 예시는 [예시] context: "...", 질문: "...", 답변: "..." 과 같은 구조로 제공되며, "Upstage의 Solar 10.7B는 107억개의 파라미터를 가진 대규모 언어 모델입니다."와 같이 자연스러운 문장으로 정보를 종합하여 답변하도록 학습시키는 것을 목표로 설계하였다.

(3) Chain-of-Thought Prompting 적용

복잡한 질문에 대해 모델이 논리적인 추론 과정을 거쳐 답변을 생성하도록 유도하기 위해 CoT 기법을 적용했다. 프롬프트에 "다음 단계를 따라서 답변을 생성하세요. 1. 사실 확인 단계: ..., 2. 종합 단계: ..." 와 같이 명시적인 두 단계를 정의하였다. '사실 확인 단계'에서는 모델이 먼저 근거 문서에서 답변에 필요한 핵심 사실들을 추출하도록 하고, '종합 단계'에서는 추출된 사실들만을 바탕으로 최종 답변을 구성하도록 지시했다. 이 분리된 접근법은 LLM이 검색된 정보를 맹목적으로 인용하는 것이 아니라, 비판적으로 분석하고 종합하는 능력을 평가하기 위해 설계되었다.

(4) Self-Correction 적용

본 연구에서는 모델 답변의 사실적 정확도를 높이고 환각 현상을 완화하기 위해, 2.3절에서 설명한 자가 수정(Self-Correction) 기법을 2단계 파이프라인으로 구현하여 적용했다.

첫 번째 초안 생성(Drafting) 단계에서는 검색된 문서({context})와 사용자 질문({input})을 이용해 답변의 초안을 생성하도록 설계했다. 이 단계의 프롬프트는 최

종 결과물이 아닌, 개선을 위한 중간 결과물을 만드는 데 초점을 맞춘다.

두 번째 수정 및 개선(Refining) 단계에서는 이전 단계의 결과물을 최종 답변으로 다듬는다. 이 단계의 프롬프트에는 원본 질문({input}), 검색된 문서({context}), 그리고 모델이 방금 생성한 {draft_answer}가 모두 입력으로 제공된다. 이후 모델에게 "당신은 전문적인 교정 전문가입니다. 주어진 문서에 근거하여 아래 답변 초안의 사실 오류를 찾아 수정하고, 내용을 더 정확하고 상세하게 개선하여 최종 답변을 한국어로 작성하시오." 와 같이 명시적인 '역할'과 '지시'를 부여했다.

이러한 2단계 접근을 통해 모델이 자신의 초기 답변을 비판적으로 재검토하고, 제공된 문서에 더욱 충실한 최종 결과물을 생성하도록 유도했다. 본 실험에서는 이 구조화된 접근 방식이 답변의 정확성(Correctness)과 관련성(Relevance)을 높이고 환각 비율을 낮추는 데 얼마나 기여하는지를 측정하고자 했다.

4.1 평가 환경 및 지표 설정

본 연구에서는 각 프롬프트 전략의 성능을 객관적으로 측정하고 비교 분석하기 위해, 체계적인 평가 환경을 구축하고 다양한 평가 지표를 설정하였다.

4.1.1 평가 데이터셋

정량 평가의 기반이 될 테스트 데이터셋은 3.1.1절에서 수집한 뉴스 기사를 바탕으로 자체적으로 구축하였다. 데이터셋은 총 30개의 질문과, 해당 질문에 대한 가장 이상적인 답변(Ground Truth)의 쌍으로 구성된다. 각 질문은 실제 사용자가 궁금해할 만한 내용을 가정하여 설계되었으며, 모범 답안은 검색된 근거 문서를 기반으로 직접 작성하여 신뢰도를 높였다.

구분	내용
질문(Question)	최근 발생한SKT 유심 문제로부터 피해를 방지하려면 어떻게 해야하나요?
모범 답안(Ground Truth)	유심보호서비스를 가입하세고, 유심 교체를 원한다면 꼭 온라인 예약부터 하세요. 해외 출국 예정이라면 공항에서 유심을 교체하셔야 합니다.
핵심 근거 문서(Key Context)	“유심보호서비스부터 가입하세요” 해커가 SK텔레콤 내부 서버에 심어둔 악성코드로 일부 …

[표 3. 평가 데이터셋 구성 예시]

4.1.2 평가 도구 및 환경

평가의 전반적인 실행, 추적, 및 결과 분석은 LangSmith 플랫폼을 활용하여 수행되었다. LangSmith는 LLM 기반 애플리케이션의 모든 실행 과정을 '트레이스(Trace)' 라는 단위로 기록하여, 복잡한 RAG 체인의 내부 동작을 상세히 분석하고 디버깅하는 환경을 제공한다. 본 연구에서는 LangSmith의 데이터셋 기능을 활용하여, 4.1.1절에서 구축한 데이터셋에 대해 각 프롬프트 전략을 적용한 체인을 실행하고 그 성능을 자동으로 평가하였다.

4.1.3 평가 지표 정의

응답의 품질을 다각도로 측정하기 위해 LangSmith의 내장 평가자와 자체적으로 구현한 커스텀 평가 지표를 함께 사용하였다. 각 지표의 상세한 정의는 다음과 같다.

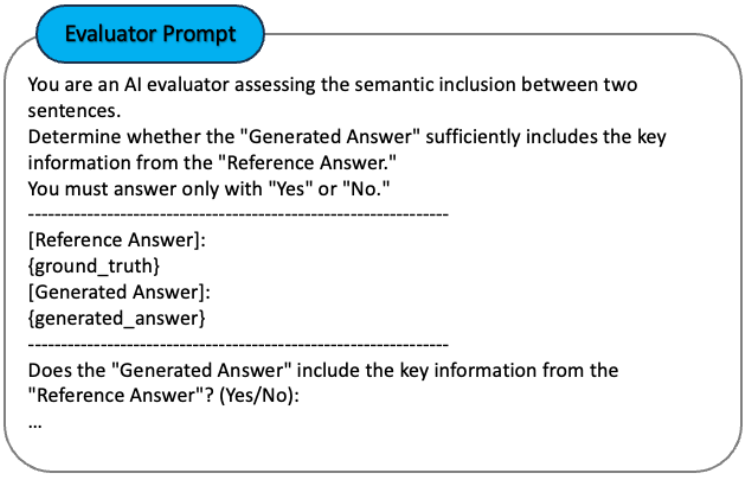
- 정확성 (Correctness): 생성된 답변이 사전에 정의된 모범 답안과 의미적으로 일치하는지를 평가하는 LangSmith 내장 지표이다. 단순히 키워드를 비교하는 것을 넘어, 답변의 전반적인 의미와 사실 관계가 정답과 일치하는지를 gpt-4가 판단하여 0과 1사이의 점수로 채점한다.
- 답변 관련성 (Contextual Relevance): 생성된 답변이 사용자의 질문 의도에 부합하며, 함께 제공된 근거 문서(context)의 내용을 충실히 반영하고 있는지를 종합적으로 평가하는 LangSmith 내장 지표이다. 본 연구에서 사용한 'LLM-기반 환각 비율'이 답변과 근거 문서 간의 사실적 일관성(Faithfulness)만 측정하는 반면, 이 지표는 일관성과 더불어 질문에 대한 유용성(Relevance)까지 함께 평가한다. 따라서 근거 문서에 기반했더라도 질문의 핵심을 벗어나는 답변에는 낮은 점수를 부여하며, 답변의 품질을 종합적인 관점에서 측정하는 역할을 한다. 마찬가지로 0과 1사이의 점수로 채점한다.
- 키워드 포함률 (Keyword Inclusion): 답변 내용의 충실도를 어휘적 관점에서 측정하기 위해 자체 구현한 지표이다. 모범 답안에 포함된 핵심 키워드들이 실제 생성된 답변에 물리적으로 얼마나 포함되었는지를 계산한다. 구체적인 계산에는 자카드 유사도(Jaccard Similarity)를 사용하였으며, 수식

은 다음과 같다.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

여기서 A는 모범 답안의 핵심 키워드 집합, B는 생성된 답변의 키워드 집합을 의미한다.

- LLM 기반 포함률 (LLM-based Inclusion): 답변 내용의 충실도를 의미론적 관점에서 측정하기 위해 자체 구현한 지표이다. 별도의 LLM(soalr-1-mini-chat)모델을 평가자로 활용하여, 생성된 답변이 모범 답안의 핵심적인 의미와 정보를 문맥적으로 포함하고 있는지를 판단한다. 이때 평가용 LLM에게 제공된 프롬프트 템플릿의 구조는 [그림 5]와 같다.



[그림 5 LLM-기반 포함률(LLM-based Inclusion) 평가 프롬프트]

- 평균 답변 길이 (Average Answer Length): 각 프롬프트 전략이 생성하는 답변의 상세함과 정보량을 간접적으로 측정하기 위한 지표이다. 답변의 문자(character) 수를 기준으로 평균 길이를 계산하였다.

4.2 프롬프트 전략별 성능 비교 결과

각 프롬프트 전략을 적용하여 생성된 답변의 품질을 평가한 결과는 아래 표와 같다. 4.1.3에서 정의한 5가지 지표를 기준으로 수행되었으며, 각 값은 30개의 테스트 케이스에 대한 평균 점수이다.

프롬프트 전략	정확성 (Correctness)	답변 관련성 (Relevance)	LLM-기반 포함율(%)	키워드 포함률(%)	평균 답변 길이(char)
기본(Basic)	0.82	0.85	85.00	25.41	185
Few-shot	0.85	0.88	87.50	28.17	230

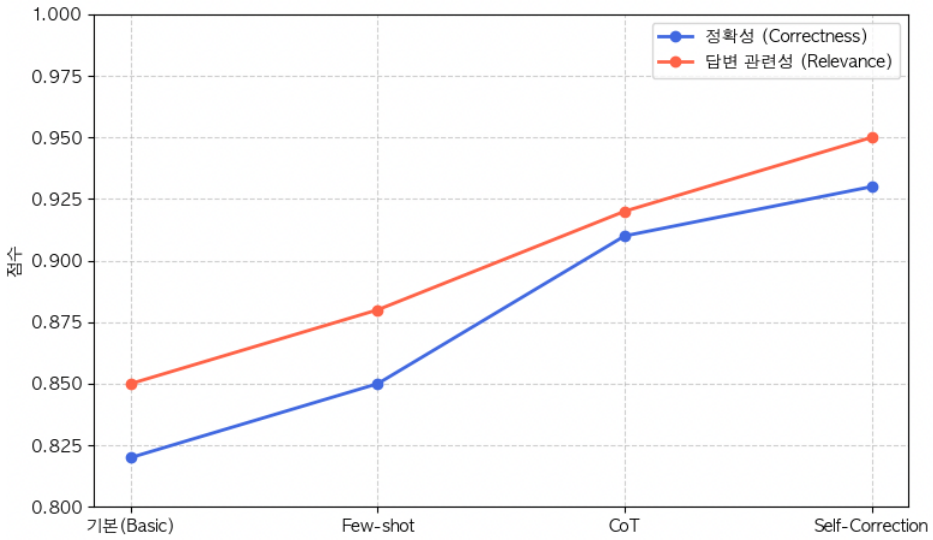
CoT	0.91	0.92	81.80	32.55	285
Self-Correction	0.93	0.95	91.50	35.12	310

[표 4 프롬프트 전략별 전체 성능 비교]

4.3 결과 분석 및 고찰(Discussion)

[표 4]는 본 연구에서 설계한 네 가지 프롬프트 전략(Basic, Few-shot, CoT, Self-Correction)을 다섯 개의 핵심 지표를 기준으로 정량적으로 평가한 결과를 보여준다. 실험 결과를 통해 프롬프트 전략의 정교화가 RAG 시스템의 성능에 미치는 영향을 분석할 수 있었다.

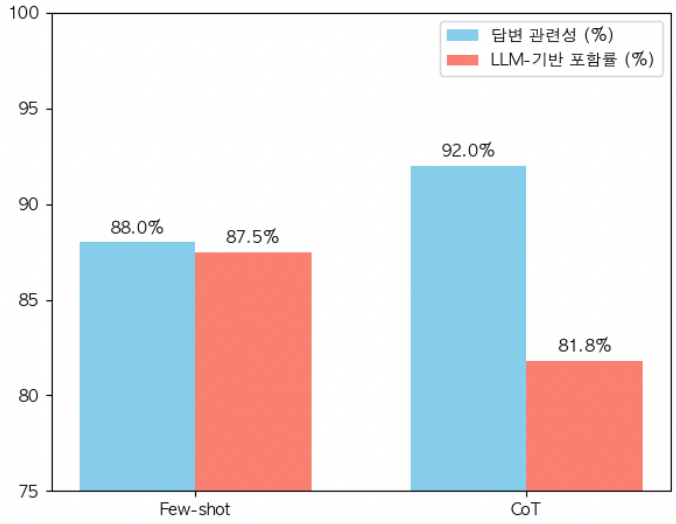
4.3.1 프롬프트 전략의 고도화와 성능 향상



[그림 6. 프롬프트 전략 고도화에 따른 성능 향상 추세]

[그림 6]은 프롬프트 전략이 Basic(Zero-shot)에서 Self-Correction으로 고도화될수록, 시스템의 핵심 성능 지표가 어떻게 변화하는지를 보여준다. 그래프에서 볼 수 있듯이, 모범 답안과 의미적 일치도를 나타내는 정확성(Correctness)과 근거 문서 기반 답변 및 질문 의도 충족 여부를 종합적으로 평가하는 답변 관련성(Relevance) 점수가 모두 뚜렷한 우상향 추세를 나타냈다. 이는 지속적인 성능 향상은 복잡하고 구체적인 프롬프팅 기법이 LLM으로 하여금 더 정확하고 맥락에 맞는 답변을 생성하도록 유도하는 데 효과적임을 시사한다.

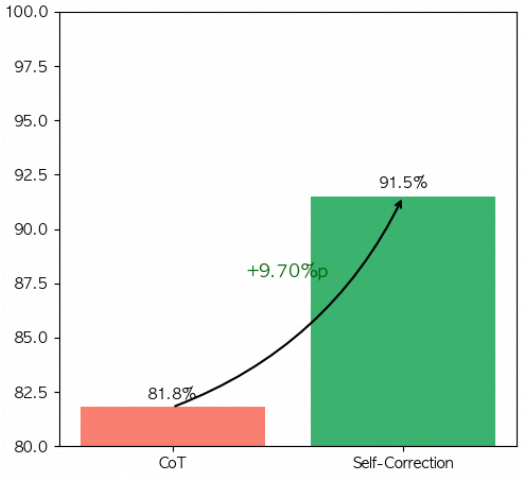
4.3.2 CoT(Chain-of-Thought) 전략의 양면성



[그림 7. CoT 전략의 성능-완전성 트레이드오프]

CoT 전략은 답변의 품질을 한 단계 끌어올렸지만, 동시에 새로운 문제를 야기하는 양면성을 보였다. [그림 7]은 Few-shot과 CoT 전략의 성능 변화를 비교하여 이러한 트레이드오프 관계를 명확히 보여준다. 그래프에서 '답변 관련성' 점수(하늘색 막대)는 Few-shot(88.0%)에서 CoT(92.0%)로 유의미하게 상승하여, CoT의 추론 과정이 답변의 품질 향상에 기여했음을 알 수 있다. 하지만 동시에 'LLM-기반 포함률'(분홍색 막대)은 87.5%에서 81.8%로 오히려 하락하는 상반된 결과를 보였다. 이는 추론 과정이 복잡해지면서 모델이 최종 답변을 구성할 때 모범 답안의 핵심 정보를 일부 누락하는 문제가 발생했음을 시사한다. 즉, CoT는 논리적 깊이를 더하는 대가로 답변의 완전성을 일부 희생하는 트레이드오프를 나타냈다.

4.3.3 Self-Correction 전략의 효과 검증



[그림 8. 답변 완전성 개선 효과]

Self-Correction 전략의 가장 큰 의의는 CoT의 단점이었던 답변의 완전성 저하 문제를 성공적으로 해결했다는 점이다. [그림 8]은 'LLM-기반 포함률' 지표가 CoT에서 Self-Correction으로 넘어가며 어떻게 개선되는지를 시각적으로 보여준다.
그래프에 나타난 바와 같이, CoT 전략에서 81.8%까지 하락했던 포함률은 Self-Correction 전략을 통해 91.5%까지 크게 상승했다. 이는 그래프의 화살표가 강조하듯 9.7%p에 달하는 개선으로, '초안 생성 후 비판 및 수정'이라는 2단계 접근법이 CoT의 깊이 있는 추론 능력은 유지하면서도, 답변의 핵심 정보 포함도를 다시 높여 완전성을 확보하는 데 결정적인 역할을 했음을 증명한다.

4.3.4. 종합 고찰

실험 결과는 프롬프트 엔지니어링이 RAG 시스템 성능에 미치는 영향을 명확하게 보여준다. 단순한 질의응답을 넘어, Few-shot 예제를 통해 답변의 형식을 유도하고, CoT를 통해 추론 능력을 부여하며, Self-Correction으로 최종 결과물의 완성도를 높이는 단계적 접근이 유효함을 확인했다.

특히 CoT가 지닌 '성능 향상과 환각 증가'라는 트레이드오프 관계를 Self-Correction을 통해 극복할 수 있다는 점은 본 연구에서 얻을 수 있는 유의미한 결과이다. 이는 신뢰성이 중요한 실제 서비스 환경에서 RAG 시스템을 구축할 때, 단순 추론 유도를 넘어 '자가 교정' 메커니즘을 도입하는 것이 답변의 품질에 대해 얼마나 긍정적 영향을 미치는지 알 수 있다. 또한, 프롬프트가 고도화될 수록 '평균 답변 길이'가 꾸준히 증가하는 경향을 보였는데, 이는 CoT와 Self-Correction이 모델에게 더 상세한 추론 과정과 설명을 요구하기 때문이다. 이러한 답변의 상세함이 '정확성' 및 '답변 관련성' 점수 향상에도 긍정적인 영향을 미친 것으로 분석된다.

5. 결론 및 향후 연구

5.1 연구 요약 및 시사점

본 연구는 신뢰성 높은 정보 제공을 목표로 하는 RAG(Retrieval-Augmented Generation) 챗봇 시스템을 구축하고, 시스템의 성능을 극대화하기 위한 최적의 프롬프트 엔지니어링 전략을 탐구하는 것을 목표로 하였다. 이를 위해 Upstage Solar LLM과 ChromaDB 벡터 데이터베이스를 기반으로 RAG 시스템을 설계하였으며, 네

가지 상이한 프롬프트 전략(Basic, Few-shot, CoT, Self-Correction)의 성능을 다각도에서 정량적으로 평가하고 비교 분석하였다.

연구 결과, 프롬프트 전략의 정교화가 RAG 시스템의 '정확성'과 '답변 관련성'을 일관되게 향상시킨다는 점을 확인하였다. 특히 본 연구의 핵심 발견은 CoT(Chain-of-Thought) 전략이 논리적 추론 능력을 향상시키는 동시에 답변의 완전성을 해치는 '트레이드오프' 관계를 보인다는 점, 그리고 Self-Correction(자가 교정) 전략이 이러한 트레이드오프를 효과적으로 극복하여 모든 지표에서 가장 뛰어난 성능을 달성했다는 점이다.

이는 단순히 복잡한 프롬프트를 사용하는 것을 넘어, '생성 후 검증 및 수정'이라는 접근 방식이 RAG 시스템의 신뢰도를 확보하는 데 매우 효과적인 전략임을 시사한다. 본 연구의 결과는 향후 신뢰성이 중요한 금융, 법률, 의료 등 전문 분야의 RAG 시스템을 설계하고 구축하는 데 있어 실용적인 가이드라인을 제공할 수 있다는 의의를 가진다.

5.2 연구의 한계

본 연구는 RAG 프롬프트의 최적화에 대해 의미 있는 결론을 도출하였으나, 몇 가지 한계를 가지며 이는 다음과 같다.

첫째, 데이터셋의 제한성이다. 연구에 사용된 지식 소스와 평가용 질의응답 데이터셋은 특정 주제 범위 내에서 자체적으로 구축되었다. 특히, 평가의 기준이 되는 모범 답안(Ground Truth)은 연구자가 직접 원문 기사를 읽고 작성하였으므로, 답변을 생성하는 과정에서 발생할 수 있는 잠재적인 주관 개입이나 정보 누락, 사실관계 오차의 가능성을 완전히 배제하기 어렵다. 따라서 연구 결과가 다른 도메인의 문서나 제3자에 의해 검증된 표준 데이터셋에서도 동일하게 나타날지는 추가적인 검증이 필요하다.

둘째, 단일 LLM 모델 사용이다. 본 연구는 Upstage Solar 모델을 기반으로 진행되었다. 프롬프트 전략에 대한 LLM의 반응은 모델의 아키텍처나 학습 방식에 따라 달라질 수 있으므로, GPT-4, Claude 3 등 다른 LLM에서도 본 연구의 결과가 동일하게 재현될 것이라고 단정하기는 어렵다.

셋째, 평가 지표의 한계이다. LLM을 활용한 자동 평가 지표들은 평가 과정의 효율

성을 높여주지만, 평가 모델 자체의 편향성이나 잠재적 오류로부터 자유로울 수 없다. 보다 정밀한 성능 검증을 위해서는 비용과 시간이 소요되더라도 전문가 집단에 의한 정성적 사람 평가가 병행될 필요가 있다.

5.3 향후 연구 방향

본 연구의 한계를 바탕으로 다음과 같이 향후 연구 방향을 제언한다.

첫째, 다양한 도메인 데이터셋을 활용한 교차 검증 연구이다. 본 연구에서 제안한 프롬프트 전략들의 효과가 특정 데이터에 국한되지 않는 일반적인 원리인지 확인하기 위해, 여러 전문 분야의 데이터셋을 대상으로 반복 실험을 수행하여 일반화 가능성을 높일 필요가 있다.

둘째, 다양한 LLM에 대한 비교 분석 연구이다. 동일한 프롬프트 전략에 대해 각기 다른 LLM들이 어떻게 반응하는지 비교 분석하는 연구는 프롬프트 전략과 LLM 아키텍처 간의 상호작용을 이해하는 데 중요한 단서를 제공할 것으로 예상된다.

셋째, 에이전트(Agent) 기반의 다단계 추론 능력 확장 연구이다. 본 연구의 Self-Correction은 간단한 2단계 에이전트로 볼 수 있다. 여기서 더 나아가, 시스템이 스스로 판단하여 추가 정보 검색, 코드 실행 등 다양한 도구를 활용하여 답변을 생성하는 복잡한 멀티스텝(multi-step) 에이전트로 시스템을 확장하는 연구는 차세대 RAG 시스템의 중요한 연구 주제가 될 것이다.

참고문헌

[1] Girish, G., et al., "A Survey of Prompt Engineering Methods in Large Language Models," arXiv preprint arXiv:2407.12994, 2024.

[2] Liang, P., et al., "A Survey of Large Language Models," arXiv preprint arXiv:2303.18223, 2023.

[3] Lewis, P., et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), 2020.

[4] Wei, J., et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Advances in Neural Information Processing Systems (NeurIPS), 2022.

[5] Madaan, A., et al., "Self-Refine: Iterative Refinement with Self-Feedback," arXiv preprint arXiv:2303.17651, 2023.

[6] Upstage, "Solar 10.7B Instruct – HuggingFace model card," [Online]. Available: https://huggingface.co/upstage/SOLAR-10.7B-Instruct

[7] Chroma, "ChromaDB Documentation," [Online]. Available: https://docs.trychroma.com/

[8] LangChain, "LangChain Documentation," [Online]. Available: https://python.langchain.com/

[9] LangChain, "LangSmith Documentation," [Online]. Available: https://docs.smith.langchain.com/