논문 분석

Retrieval-Augmented Generation for Large Language Models

(Yunfan Gao, Yun Xiong, Xinyu Gao, ...)

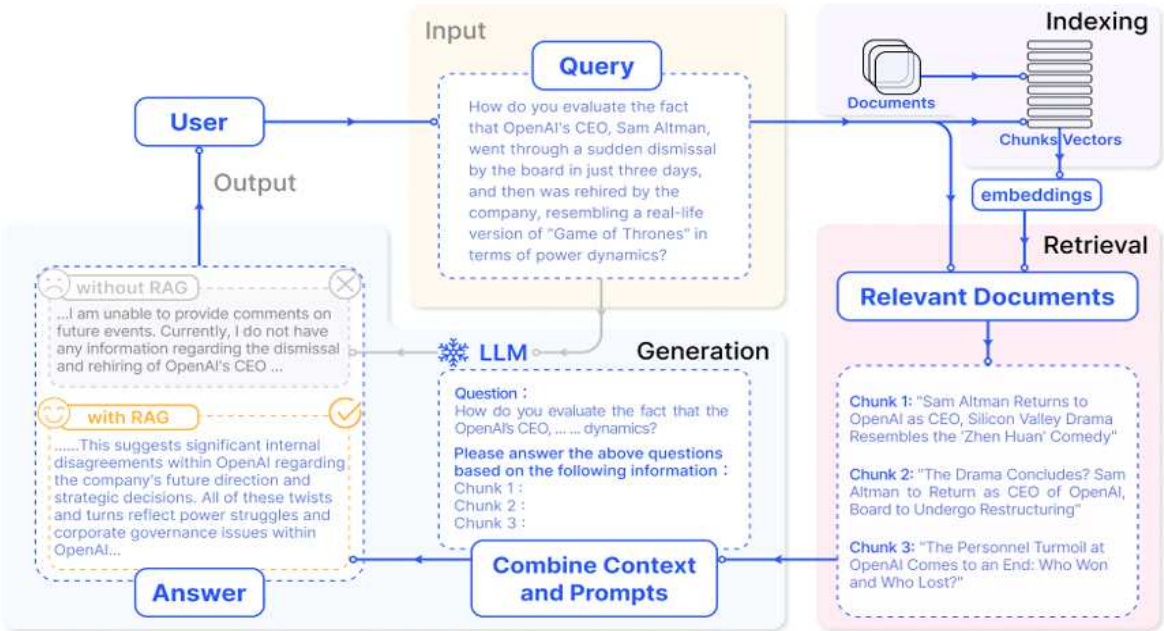
1. Introduction

LLM은 괄목할만한 성과를 보여줬지만, 상당한 제한점들이 존재한다. 특히 특정한 분야, 혹은 전문 지식이 필요한 분야에 말이다. 현재 정보나 training data를 넘어서는 queries를 다룰 때 “hallucinations(환각)” 현상이 일어나기 때문이다.

이를 극복하기 위해, Retrieval-Augmented Generation(RAG)를 활용해 LLMs를 강화시킬 수 있다. RAG는 관련된 외부 지식들을 문서의 chunck 단위로 retrieving 하는데, 관련된 문서를 찾기 위해 semantic similarity calculation을 활용한다.

RAG를 활용하면 외부 지식들을 참조하면서, 사실이 아닌 내용을 output으로 내놓는 경우를 효과적으로 줄어들게 만들 수 있는데, 이는 chatbots를 발전시키는 데 핵심적인 역할을 했을 뿐만 아니라 LLM을 현실 세계에 더욱 잘 활용할 수 있도록 만들어 주었다.
RAG 연구는 LLM가 더 복잡하고, 전문 지식이 필요한 분야에도 잘 답변할 수 있도록 발전시켜줌.

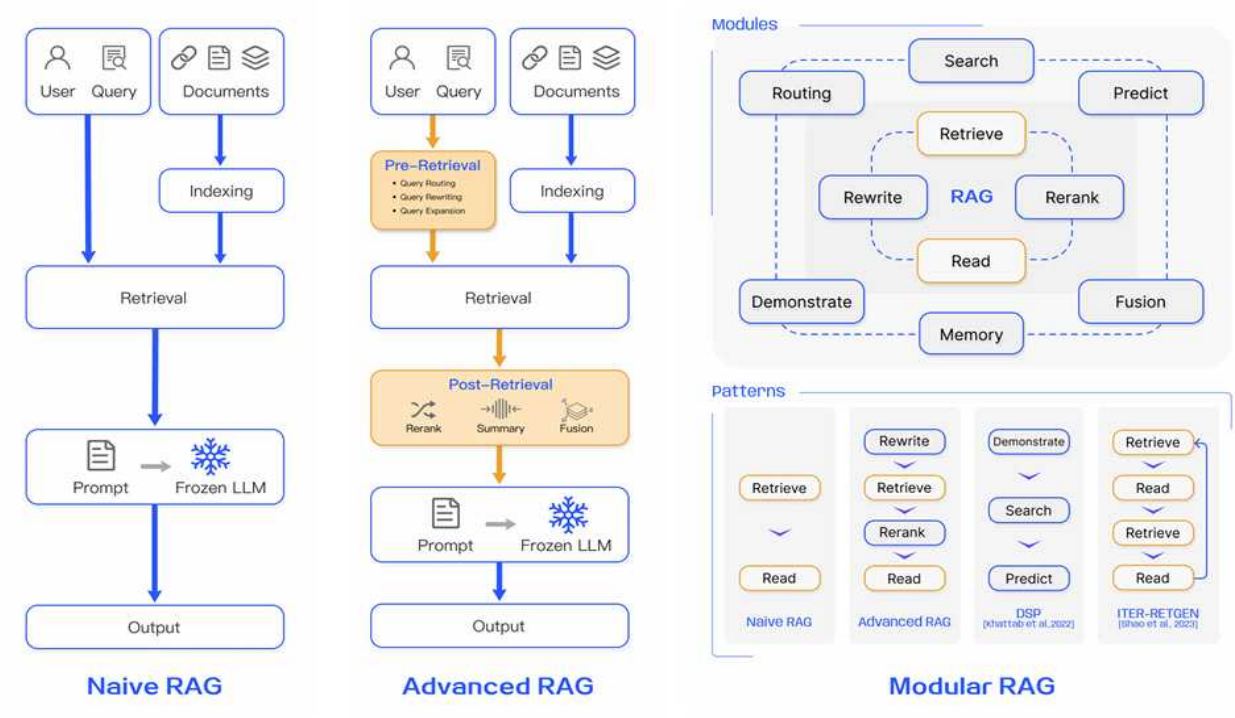
이 논문은 100개 이상의 RAG resarch들로부터 기술적 패러다임, research 도구, 그리고 Retrieval, Generation, Augmentation의 core state를 분석했고, 나아가 RAG에 대한 downstream tasks, datasets, benchmarks, evaluation applicable 들을 소개한다.



2. Overview of RAG

ChatGPT를 예로 들면, RAG는 pre-training data에 의존하는 LLM 모델의 특징에 기인하는 업데이트된 정보를 제공하는 능력의 부재의 간극을 채워주는 역할을 한다.
이는 external databases로부터 정보들을 sourcing하고 incorporating 함으로써 이루어지는데, 즉 RAG는 user의 query와 관련이 있는, 새로운 article 들을 가져와 기존의 질문과 함께 합쳐 comprehensive한 prompt를 만들어 LLMs가 well-informed 한 대답을 하도록 강화해준다.

RAG는 시간이 지남에 따라 계속해서 발전하고 있는데, 크게 다음 3가지 스텝으로 구분할 수 있다.
Naive RAG -> Advanced RAG -> Modular RAG
(뒤에 두 모델은 Naive RAG의 단점들에 대응되어 나온 것들)



A. Naive RAG

가장 초기의 방법론으로, indexing, retrieval, generation이라는 전통적인 처리 과정을 포함한다. (as known as Retrieve-Read framework)

(1) Indexing

(a) raw data(PDF, HTML, Word 등)를 cleaning하고, extraction 한다.
(b) uniform plain text format으로 변환시킨다.
(c) 그 후, LLM 모델의 context limitation을 수용하기 위해, text들을 작은 chunks들로 분해한다.
(d) embedding model에 의해 vector representations로 인코딩 된 후, vector database에 저장된다.

(2) Retrieval

(a) query를 vector representations로 인코딩하기 위해서 Indexing 과정에서 사용한 embedding model과 동일한 모델을 사용해 인코딩한다.
(b) query vector와 indexd corpus 안에 있는 vector of chunks의 유사도(similarity score)를 계산한다.
(c) System은 가장 좋은 유사도를 보이는 top K chunks를 우선시해서 retrieves 한다.
(d) 이 chunk들은 prompt의 context를 확장하는 데 사용된다.

(3) Generation

(a) 제안된 Query와 선택된 documents들은 일관성 있는 prompt로 합성된다.
(b) LLM이 답변을 생성하기 위해 사용된다.

Naive RAG 방법은 다음과 같은 약점들이 존재한다.

(1) Retrieval Challenges : retrieval pahse에서 misaligned 하거나 관계없는 chunks를 가져오거나, 혹은 중요한 정보를 놓칠 수 있다.

(2) Generation Difficulties : 답변을 생성할 때, 환각 문제가 발생할 수 있다. 즉, retrieved 된 context에 관계없는 내용을 답변할 수 있다. 또한 관계없고, 치명적이고, 편향된 결과를 내놓을 수 있다.

(3) Augmentation Hurdles : 복수의 retrieval로부터 retrieved 된 information 들을 통합하는 과정이 쉽지 않고, 때때로 disjointed, incoherent 한 답변을 야기한다.
-> 반복적인 답변을 야기할 수도 있음! 그렇다고 single retrieval을 사용하기에는 적절한 context information을 습득하기에 충분하지 않음!

B. Advanced RAG

Naive RAG의 한계점을 극복하기 위해 나온 모델로,
(1) retrieval의 질적 강화를 위해 pre-retrieval과 post-retrieval 전략을 채택한다.
(2) Naive RAG의 Indexing 문제를 해결하기 위해, sliding sindow 접근, fine-grained segmentation, incorporation of metadata를 활용한다.
(3) retrieval process를 간소화하기 위해 여러 가지 optimization method 들을 통합한다.

(a) Pre-Retrieval Process : 가장 중요한 요점은 indexing 구조와 기존(original)의 query를 최적화하는 것이다.
- Indexing 구조 최적화 : data granularity(세분성) 강화, index 구조 최적화, metadata 추가, alignment 최적화, mixed retrieval 활용
- 기존 query 최적화 : 유저의 기존 질문을 retrieval task에 맞게 더 확실하고, 적절한 쿼리로 변환하는데, 이때 query expansion과 다른 기술들이 활용된다.

(b) Post-Retrieval Process : chunk 들을 rerank(재등급) 시키고 context compressing(압축) 시키는 것으로, 가장 관련 있는 내용을 prompt의 edge에 재위치 시키기 위해 retrieved 된 정보들을 re-ranking 하는 것이 핵심 개념이다.
-> LlambaIndex, LangChain, HayStack 같은 framework 들로 위 내용 구현 가능!

모든 연관된 문서들을 LLM에게 전달하는 것은 정보의 overload로 이어져 질문과 상관없는 대답을 할 가능성을 야기하기 때문에, post-retrieval은 필수적인 정보만을 선택하고, 중요한 section을 강조하고, 처리될 context를 줄이는 것을 목표로 삼음.

C. Modular RAG

Modular RAG는 이전 두 RAG보다 이식성, 다재다능성 측면에서 더욱 강화된 모델로, similarity searches를 위한 search 모듈을 더하거나, fine-tuning 과정을 통해 retriever를 refining 하는 등 다양한 전략들을 통합시켰다.
또한, 구체적인 난관들을 해결하기 위해 기존의 RAG 모듈을 restructured하고, RAG pipelines을 rearranged 하는 등의 혁신을 도입했다.

(a) New Modules : retrieval과 processing 능력을 강화하기 위해 추가적인 요소를 더했다.

(1) Search module: 구체적인 시나리오에 적응하면서 LLM-generated code나 query languages를 사용해 search 엔진, databases, knowledge graph 등 다양한 데이터 소스로부터 직접적인 접근이 가능하게 만든다.

(2) RAG-Fusion : 전통적인 search limitations를 해결하기 위해 등장한 모듈로, user의 쿼리를 다양한 관점으로 확대해주는 multi-query 전략을 도입하면서 parallel vector searches와 intelligent re-ranking을 활용해 명백하고 변환적인 지식을 모두 드러내도록 한다.

(3) Memory module : LLM의 memory를 활용해 retrieval을 가이드할 뿐만 아니라. unbounded 된 memory pool을 만들고 반복적인 self-enhancement를 통해 text를 데이터 분포와 더 가깝게 정렬한다.

(4) Routing in the RAG system : 다양한 데이터 소스를 탐색하며, 쿼리에 대해 최적의 경로를 선택하는데, 이 경로에는 요약, 특정 데이터베이스 검색, 여러 정보 스트림을 통합하는 작업을 포함한다.

(5) Predict module : LLM을 통해 직접적으로 문맥을 생성해 중복과 노이즈를 줄이는 것을 목표로 삼으며, 이를 통해 정보의 관련성과 정확성을 보장한다.

(6) Task Adapter module : 다양한 다운스트림 작업(downstream tasks)에 맞게 RAG 시스템을 조정해, zero-shot 입력에 대한 프롬프트 검색을 자동화하고, few-shot 질의 생성을 통해 태스크별 검색기를 만든다.

이러한 종합적 접근 방식은 정보 검색 프로세스를 간소화하고, 검색된 정보의 품질과 관련성을 크게 향상시키며, 다양한 작업과 쿼리에 대해 더 정밀하고 유연한 대응이 가능하게 한다.

(b) New Patterns : Modular RAG는 특정 문제를 해결하기 위해 모듈을 교체하거나 재구성할 수 있도록 하여 놀라운 적응성(adaptability)을 제공하는데, 이는 단순한 검색(Retrieve) 및 읽기(Read) 매커니즘으로 구성된 fixed-structures의 Naive RAG 및 Adanced RAG의 성능을 뛰어넘는다.
게다가, 새로운 모듈을 통합하거나 기존 모듈 간의 상호작용 흐름을 조정해 모듈을 교체하거나 재구성하는 등의 유연성을 확장하는 특징은 다양한 작업에 대한 적용 가능성을 높여준다.

(1) Rewrite-Retrieve-Read 모델 : LLM의 기능을 활용하여 재작성 모듈을 사용한 검색 쿼리 정제, LM-feedback 매커니즘을 통한 재작성 모델 업데이트 등을 진행해 작업 성능을 향상시킨다.

(2) Generate-Read 접근법 : 기존의 검색 과정을 LLM이 생성한 콘텐츠로 대체하는 방식

(3) Recite-Read 접근법 : 모델의 가중치에서 직접 정보를 검색하는 방식을 강조하여 지식 집약적인 작업을 처리하는 모델의 능력을 향상시키는 방식

(4) Hybrid retrieval 전략 : 키워드 검색, 의미적 검색, 벡터 검색을 통합해 다양한 쿼리에 대응하고, 서브쿼리와 가상의 문서 임베딩(HyDE)을 활용해 생성된 답변과 실제 문서 간의 임베딩 유사성을 기반으로 검색 관련성을 향상시키는 전략

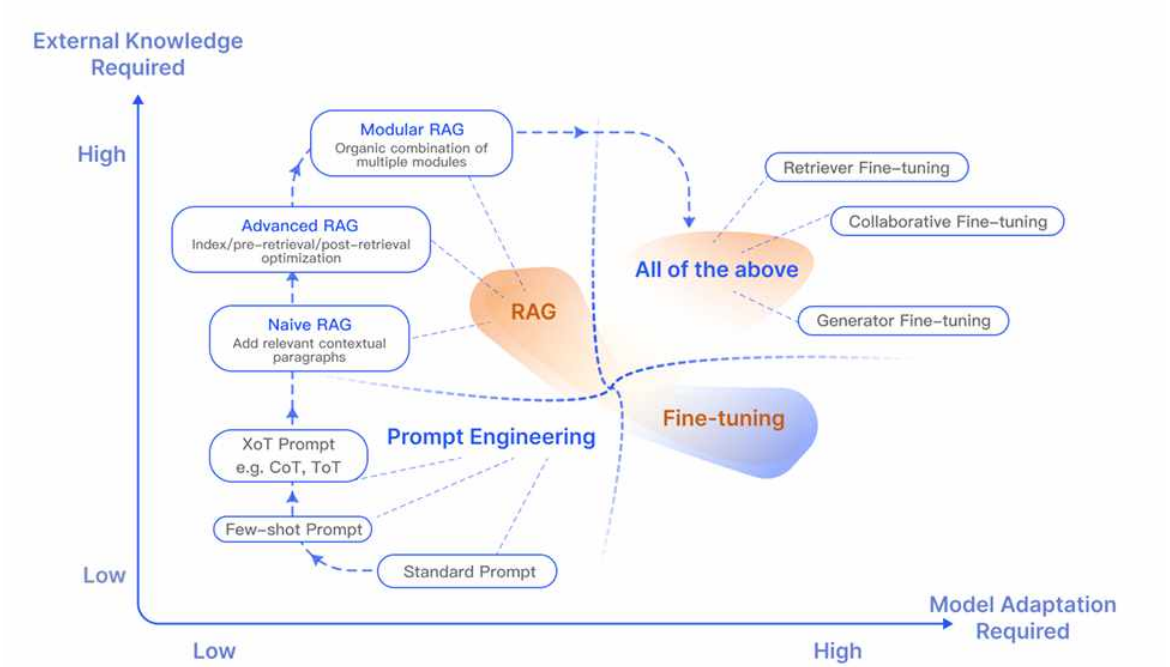
(5) Demonstrate-Search-Predict(DSP) 프레임워크 및 ITERRETGEN (반복적인 Retrieve-Read_Retrieve-Read 흐름 사용) : 한 모듈의 출력을 다른 모듈의 기능을 강화하는데 활용하는 동적 사용 방식으로, 모듈 간 시너지를 향상시키는 정교한 접근 방식을 설명한다.

이러한 Flexible orchestration(조율) of Modular RAG Flow는 FLARE 및 self-RAG와 같은 기법을 통해 적응형 검색의 이점을 보여주는데, 이는 다양한 시나리오를 기반으로 검색의 필요성을 평가함으로써 고정된 RAG 검색 프로세스의 성능을 뛰어넘는다. 더해서, RAG 시스템이 fine-tuning 또는 강화학습과 같은 다른 기술들과 더욱 쉽게 통합할 수 있도록 만든다.

예로, 검색기의 검색 성능을 향상시키기 위한 파인튜닝, 생성기의 개인화된 출력을 위한 파인튜닝, 협력적 파인튜닝(collaborative fine-tuning)과 같은 과정들을 들 수 있다.

D. RAG vs Fine-tuning

LLM의 증강은 LLM의 확산에 따라 많은 관심을 받고 있는데, LLM을 최적화하는 방법에는 RAG, Fine-tuning(FT), 그리고 prompt engineering이 있다.



각 방법에는 그림에서 볼 수 있듯이 독특한 특징을 갖고 있는데, 위 그림은 외부 지식 요구사항(y축)과 모델 적응 요구사항(x축)이라는 두 가지 차원으로 세 가지 방법의 차이를 설명한다.

(a) RAG : 모델에게 정보 검색을 위한 맞춤형 교과서를 제공하는 것에 비유할 수 있는데, 이는 정확한 정보 검색 작업에 이상적이다.
-> 실시간 지식을 업데이트를 제공하고 외부 지식 차원을 효과적으로 활용해 해석 가능성이 높은 동적 환경에서 우수한 성능을 보인다.
-> 데이터 검색과 관련된 더 높은 지연(latency)과 윤리적 고려사항이 문제점으로 꼽을 수 있다.

(b) prompt engineering : 외부 지식과 모델 적응에 대한 최소한의 필요성으로 모델의 고유한 능력을 활용한다.

(c) FT(Fine-Tuning) : 시간이 지남에 따라 지식을 내면화하는 학생과 비슷하게 볼 수 있으며, 특정 구조, 스타일 또는 형식을 복제해야 하는 상황에 적합하다.
-> 보다 더 정적인 방식으로, 업데이트를 위해 재학습이 필요하지만 모델의 행동과 스타일을 깊이 있게 맞춤화할 수 있다.
-> 데이터셋 준비와 훈련을 위해 상당한 계산 자원을 요구하며, 환각(hallucinations) 문제를 줄일 수 있지만 익숙하지 않은 데이터에 대해서는 어려움을 겪을 수 있다.

성능적 측면에서 이들을 비교해 보자면, 여러 지식 집약적인 작업에서의 성능평가 결과는 비지도 학습 방식의 FT가 약간 더 나은 성능을 보였지만, RAG가 기존 훈련 중에 접한 지식과 완전히 새로운 지식 모두에서 지속적으로 더 뛰어난 성능을 보였고, LLM은 비지도 FT를 통해 새로운 사실적 정보를 학습하는 데 어려움을 겪는 것으로 나타났다.

결국, RAG와 FT 사이의 선택은 응용 컨텍스트에서 데이터 동적성, 맞춤화 및 계산 능력에 대한 특정 요구에 따라 달라지는데, RAG와 FT는 상호 배타적이지 않으며 서로 보완할 수 있고, 이를 통해 모델의 능력을 다양한 수준에서 향상시킬 수 있다.

즉, 경우에 따라서 이들의 결합된 사용은 최적의 성능을 이끌어 낼 수 있는데, RAG와 FT를 포함하는 최적화 과정은 만족스러운 결과를 얻기 위해서는 여러 번의 반복이 필요할 수 있다.

3. Retrieval

RAG를 기준으로, 데이터 소스에서 관련 문서를 효율적으로 검색하는 것이 무엇보다도 중요하다. 다음에는 관련된 몇 가지 핵심문제들이 있는데, 차례로 검색 소스, 검색 세부 수준, 검색의 전처리, 그리고 임베딩 모델의 선택을 포함한다.

A. Retrieval Source(검색 소스)

RAG는 LLM을 향상시키기 위해 외부 지식에 의존하며, 검색 소스의 유형과 검색 단위의 세부 수준이 최종 생성 결과에 영향을 미친다.

(1) Data Structure

초기에는 텍스트가 RAG의 주된 검색 소스였다면, 점진적으로 검색 소스 향상을 위해 반구조화된 데이터(PDF)와 구조화된 데이터(KG; Knowledge Graph)를 포함하도록 확장됐다.
추가로, 외부 소스에서 검색하는 것 외에도 LLM이 자체적으로 생성한 콘텐츠를 검색 및 향상 목적으로 활용하는 경향이 증가하고 있다.

(a) 텍스트와 같은 비정형 데이터(Unstructured Data) :
가장 널리 사용되는 검색 소스로, 주로 corpus에서 수집된다. 예로 위키피디아 덤프를 주요 검색 소스로 사용하는 개방형 질의응답(ODQA) 작업이 있는데, 이러한 백과사전형 데이터 외에도 일반적인 비정형 데이터에는 다국어 텍스트 자료와 의료 및 법률 분야와 같은 도메인 특화 데이터가 포함된다.

(b) 반구조화된 데이터(Semi-structured Data) :
일반적으로 텍스트와 테이블 정보가 혼합된 데이터를 의미하며, 대표적인 예로 PDF가 있다. 반구조화된 데이터를 처리하는 것은 기존 RAG 시스템의 두 가지 주요 이유로 인해 어려움이 있는데,

- 첫째, 텍스트 분할 과정에서 테이블이 의도치 않게 분리될 수 있으며, 이로 인해 검색 중 데이터 손상이 발생할 수 있고,
- 둘째, 테이블을 데이터에 포함시키면 의미적 유사도 검색을 복잡하게 만들 수 있기 때문이다.

이에, 반구조화된 데이터를 처리할 때의 접근 방식 중 하나는 바로 TableGPT와 같은 모델을 활용하여 LLM의 코드 실행 능력을 이용해 데이터베이스 내 테이블에서 Text-2-SQL 쿼리를 수행하는 것이다.

또는, 테이블을 텍스트 형식으로 변환하여 텍스트 기반 방법을 통해 추가 분석할 수도 있지만, 이러한 방법들은 최적의 해결책이 아니기 때문에 이 분야에서의 상당한 연구 과정이 필요하다.

(c) 지식 그래프와 같은 구조화된 데이터(Structured Data):
일반적으로 검증된 데이터이며, 보다 정확한 정보를 제공할 수 있다. 예로 KnowledGPT는 지식 베이스(KB) 검색 쿼리를 생성하고, 이를 개인화된 데이터베이스에 저장해 RAG 모델의 지식 풍부함을 향상시킨다.

텍스트 기반 그래프를 이해하고 질문에 답하는 LLM의 한계를 극복하기 위해, G-Retriever는 그래프 신경망(GNN), LLM, RAG를 통합해 LLM의 소프트 프롬프팅을 통해 그래프 이해 및 질의응답 기능을 향상시킨다.

또한, 특정 그래프 검색을 최적화하기 위해 'Prize-Collecting Steiner Tree(PCST)' 최적화 문제를 활용하는데, 구조화된 데이터베이스를 구축, 검증 및 유지하는 데는 추가적인 노력이 필요하다.

(d) LLM이 생성한 콘텐츠:
RAG에서 외부 보조 정보의 한계를 해결하기 위해, 즉 LLM의 내부 지식을 활용하기 위해 사용하는 컨텐츠로, 이를 구현하기 위해 다음과 같은 방법론들이 사용된다.

- SKR은 질문을 '알려진 질문'과 '알려지지 않은 질문'으로 분류한 후, 검색 강화 기능을 선택적으로 적용한다.
- GenRead는 검색기(retrieval)를 LLM 생성기(generator)로 대체하며, LLM이 생성한 문맥이 인과적 언어 모델링(causal language modeling)의 사전 학습 목표와 더 잘 맞아 떨어지기 때문에 보다 정확한 답변을 포함하는 경우가 많다는 사실을 발견했다.
- Selfmem은 검색이 강화된 생성기를 활용해 무한한 메모리 풀을 반복적으로 생성하며, 메모리 선택기를 통해 원래 질문과 상호작용하는 이중 문제9dual problems)로 작용하는 출력을 선택함으로써 생성 모델을 스스로 강화한다.

이러한 방법론들은 RAG에서 혁신적인 데이터 소스 활용의 폭을 넓히고 있으며, 모델 성능과 작업 효율성을 향상시키는 데 조력하고 있음을 알 수 있다.

(2) Retrieval Granularity (검색 단위의 세분화)
검색 소스의 데이터 형식 외에도, 검색 데이터의 세분화(검색 단위의 크기)는 매우 중요한 요소인데, 단위의 종류에는 다음과 같은 것들이 있다.

(a) 거친 단위(Coarse-grained) retrieval units
: 이론적으로 문제 해결에 더 관련성이 높은 정보를 제공할 수 있지만, 불필요한 내용이 포함될 가능성이 있어 검색기(retrieval)와 언어 모델이 후속 작업을 수행하는 데 방해가 될 수 있다.

(b) 세밀한 단위(fine-grained) retrieval units
: 검색 과정의 부담을 증가시키며, 의미적 일관성이나 필요한 정보를 충족시킨다는 보장이 없다.

추론과정에서 적절한 검색 단위를 선택하는 것은 밀집 검색기(dense retriever)의 검색 성능과 후속 작업 성능을 향상시키는 간단하면서도 효과적인 전략이 될 수 있다.

- 텍스트에서의 검색 단위에는 작은 단위에서 큰 단위까지 다양한 범위를 가지며, 토큰(Token), 구(Phrase), 문장(Sentence), 명제(Proposition), 청크(Chunks), 문서(Document) 등이 있다.

이 중에서 DenseX는 명제(Proposition)를 검색 단위로 사용하는 개념을 제안했는데, 명제는 텍스트 내의 가장 작은 의미 단위(atomic expression)으로 정의되며, 각각이 고유한 사실적 내용을 포함하고 독립적인 자연어 형식으로 표현된다.

이러한 접근 방식은 검색의 정확성과 관련성을 향상시키는 것을 목표로 한다.

- 지식 그래프(KG)에서 검색 단위는 엔터티(Entity), 삼중항(Triplet), 서브 그래프(sub-graph)로 나뉘는데, 이러한 검색 단위는 후속 작업에 맞게 조정될 수 있으며, 예로 추천 시스템에서 항목 ID 검색 또는 문장 쌍 검색과 같은 방식으로 활용될 수 있다.

B. Indexing Optimization

색인(Indexing) 단계에서는 문서를 처리, 분할, 임베딩 변환하여 벡터 데이터베이스에 저장하는 작업이 이루어지는데, 이 색인 품질이 검색 단계에서 올바른 문맥(Context)을 가져올 수 있는지를 결정한다.

(1) Chunking Strategy

가장 일반적인 방법은 문서를 고정된 개수의 토큰 단위(100, 256, ...)로 분할하는 것으로, 큰 청크의 경우 더 많은 문맥을 포함할 수 있지만 노이즈가 증가하고 처리 시간이 길어지며 비용이 증가한다. 반대로, 작은 청크의 경우 노이즈가 적고 처리 속도가 빠르며 비용이 낮지만, 필요한 문맥 정보를 충분히 전달하지 못할 수 있다.

청크를 최적화 하는 방법에는 다음과 같은 방법들이 있다.

- 재귀적 분할(Recursive Splits) 및 슬라이딩 윈도우(Sliding Window) 기법 : 문장을 잘리지 않도록 최적화하고, 여러 번 검색하여 전역적인 관련 정보를 계층적으로 합치는 방식을 뜻한다. 단점으로는 문맥 길이와 의미적 완전성(Semantic Completeness)의 균형을 맞추기 힘들다는 점이 있다.

- Small2Big 방법 : 문장을 작은 단위(small)로 검색하고, 앞뒤 문장을 추가(Big context)하여 LLM에 제공하는 방식을 말한다.

색인 단계에서 문서를 효과적으로 분할하는 것은 검색 성능에 직접적인 영향을 미치기 때문에 토근 기반 분할의 한계를 뛰어넘고자 슬라이딩 윈도우 기법, 재귀적 분할 등의 기법들이 등장하였지만, 모든 방법이 완벽하지는 않으며 문맥 길이와 의미 보존 간의 균형을 맞추는 것은 여전히 연구 과제로 남아있다.

(2) Metadata Attachments

청크(Chunks)에 메타데이터를 추가하면 검색 성능을 향상시킬 수 있는데, 메타 데이터의 종류에는 페이지 번호, 파일명, 저자, 카테고리, 타임스탬프 등이 있다.

메타데이터 기반 검색 최적화 방법에는 특정 메타데이터 조건으로 검색 범위를 제한하거나 (ex-특정 저자가 작성한 문서만으로 검색), 최신 정보를 반영하고 오래된 정보를 배제할 수 있도록 문서의 타임스탬프에 가중치를 부여하는 타임스탬프 가중치 조정(Time-Aware RAG) 방법이 있다.

원본 문서를 추출하는 것 외에도, 인위적으로 메타 데이터를 생성할 수 있는데, 문단 요약(Summary)를 추가하거나 가상 질문을 생성하는 Reverse HyDE 기법이 그 예이다.

특히 Reverse HyDE 기법은 LLM을 활용하여 문서에서 답변할 수 있는 질문을 생성하는 방식으로, 검색할 때 실제 질문과 가상 질문의 유사도를 계산해 의미적 간격을 줄이는 방법을 택한다. 이는 질문과 답변 사이의 의미적 차이를 최소화하고, 보다 정확한 문서 검색을 가능하게 한다.

(3) Structural Index (구조적 인덱스)

RAG 시스템에서 검색 효율을 높이는 효과적인 방법 중 하나는 문서를 계층적으로 정리하는 것인데, 구조적 인덱스를 활용하면 더 빠르고 정확한 검색이 가능하다.

(a) 계층적 인덱스 구조 (Hierarchical Index Structure)

파일을 부모-자식 관계로 정리하는 구조로, 문서를 계층적으로 정리하여 데이터 탐색 속도를 향상시킨다. 부모 노드에는 문서 요약(Summary)을 저장해 검색을 가속화하고, 자식 노드에는 실제 검색할 청크(Chunks) 데이터를 저장한다.

장점으로는 검색 속도의 향상과 청크 단위 검색 시 발생하는 문맥 손실(Block Extraction Issus) 완화, 그리고 불필요한 데이터 로딩을 줄여 RAG 성능을 개선한다는 점이 있다.

(b) 지식 그래프 기반 인덱스 (KG Index)

지식 그래프(KG)를 활용하면 문서 간 개념 및 관계를 체계적으로 정리할 수 있는데, 이는 문서 간 개념/앤터티 간의 관계를 정리하여 일관성을 유지하고, 검색 시 LLM이 문서 구조를 더 잘 이해하고 의미론적으로 정합성 있는 답변을 생성하게 만든다. 또, 환각 문제를 줄여주기도 한다.

예로 KGP(Knowledge Graph-based Preprocessing) 는 문서의 문단(Paragraph)과 구조(페이지, 표 등)를 노드로 변환하는 과정으로, 문서 내 문단 간 유사성과 관계를 엣지로 표현한다. 이는 다중 문서 환경에서의 검색 및 추론 성능을 향상시켜 준다.

C. Query Optimization

Naive RAG의 주요 문제점은 바로 사용자의 원래 질문(Query)에 직접 의존하는 방식에 의한 한계점인데, 자세하게 다음과 같은 것들이 있다.

(a) 부정확한 질문의 검색 성능 저하

사용자가 명확하고 정제된 질문을 작성하는 것은 어려운데, 이렇게 질문이 애매하거나 불명확하면 관련성이 낮은 결과가 검색되고 복잡한 질문일수록 검색 효율성이 떨어지게 된다.

ex) ✗ *"AI 모델은 어떻게 학습하나요?"*→ 범위가 너무 넓음
✓ *"Transformer 기반 LLM의 사전학습과 미세조정(Fine-tuning) 과정은?"*→ 검색 최적화 가능

(b) 언어의 복잡성과 모호성 (Ambiguity)

전문 용어 또는 다의어 해석의 문제점으로, 동일한 약어가 여러 의미를 가질 수 있다. 이는 문맥을 반영한 질문 리팩토링(Query Refinement), 다의어 처리(Disambiguation)를 위한 전처리 과정 도입, 질문을 구조화해 RAG 시스템의 검색 성능을 개선하는 등 다양한 해결책이 반영되야 한다.

(1) Query Expansion

하나의 질문을 여러 개의 세부 질문으로 확장하여, 더 풍부한 내용과 맥락을 제공하고, 이를 통해 최적의 관련성을 확보하는 것을 뜻한다. 이는 세부 질문을 통해 원본 질문의 구체적인 뉘앙스를 보강하고, 검색 결과의 정확도를 향상시킨다.

- Multi-Query : LLM을 활용한 질문 확장 방법으로, 쿼리를 확장해 여러 개의 쿼리를 병렬로 실행한다. 이때 확장은 임의로 이루어지지 않고 정교하게 설계되고, 여러 질문을 한 번에 처리하여 다양한 관점에서 검색이 가능하게 한다.

- Sub-Query : 원본 질문을 더 간단한 하위 질문으로 분해하여, 문제 해결을 위한 추가 맥락을 제공하는 것으로, Least-to-Most prompting 방법을 사용해 복잡한 질문을 단계적으로 풀어나간다. 즉, 복잡한 질문을 나누어 세부적인 질문을 생성하고, 이를 결합하여 최종적으로 완전한 답을 도출한다.

- Chain-of-Verification(CoVe) : 확장된 쿼리를 검증하는 과정으로, 생성된 쿼리들이 신뢰성 높은 답변을 도출하도록 보장하는 역할을 맡는다. 검증된 확장 쿼리는 환각을 줄이고, 정확한 결과를 제공한다.

(2) Query Transformation

사용자의 원본 쿼리 대신에 변형된 쿼리를 사용해 관련된 청크를 검색하는 방법으로, 쿼리 자체를 변형하거나 재구성하여 보다 정확한 검색 결과를 얻는다. 관련 기법에는 다음과 같은 것들이 있다.

- Query Rewrite : 원본 쿼리가 LLM 검색에 최적화되지 않을 수 있기 때문에, 이를 변형하여 더 적합한 쿼리로 만드는 기법으로, LLM을 활용하여 사용자의 질문(쿼리)을 최적화된 형태로 재구성하거나 특화된 작은 언어 모델(RRR; Rewrite-retrieve-read)을 사용한다. 대표적인 적용 사례로는 Taobao의 BEQUE 시스템이 있는데, 이는 긴 쿼리에서 검색 효과성을 크게 향상시켰으며, GMV(Gross Merchandise Volume)가 증가하는 성과를 냈다.

- HyDE (Hypothetical Document Embedding) : 원본 쿼리로 가상의 문서(hypothetical documents)를 구성해 답변 간 임베딩 유사도를 기반으로 검색하는 방법으로, 문제나 쿼리 자체가 아닌 답변 간 유사도를 중시하여 더 관련성 있는 검색 결과를 유도한다. 이는 정확한 답변을 추출하는 데 집중하며, 단순한 검색 질의보다 더 효과적인 정보 검색을 가능하게 한다.

- Step-back Prompting : 원본 쿼리를 고차원 개념 질문(step-back question)으로 변형하는 방법으로, 원본 질문을 추상화해 더 넓은 범위의 질문을 생성하고, 두 가지 질문 모두를 이용해 검색한다. (두 쿼리의 결과를 결합해 더 정확한 답변 생성에 활용한다.)

(3) Query Routing

다양한 쿼리에 대해 적합한 RAG 파이프라인으로 라우팅하는 기법으로, 다채로운 상황을 처리할 수 있도록 유연하고 효과적인 RAG 시스템을 설계할 수 있게 만든다. 관련 기법에는 다음과 같은 것들이 있다.

- Metadata Router / Filter : 쿼리에서 핵심 키워드(엔티티)를 추출한 후, 메타데이터를 활용해 검색 범위를 좁히는 방법으로. 다음과 같은 절차를 따른다.

  (a) 쿼리에서 엔티티 추출 : 검색 쿼리에서 중요한 키워드나 엔티티 추출
  (b) 메타데이터 기반 필터링 : 추출된 엔티티와 관련된 메타데이터를 사용해 검색 범위를 축소

  이는 특정 주네나 도메인에 대한 검색을 효율적으로 제한할 수 있어 효율성을 향상시킨다.

- Semantic Router : 쿼리의 의미적 정보를 기반으로 라우팅하는 방법으로, 쿼리의 의미적 요소를 분석해 관련성이 높은 정보를 제공하는 RAG 파이프라인으로 라우팅한다. 예로, 쿼리의 의미적 유사도를 측정해 관련된 문서나 데이터를 정확하게 찾도록 라우팅을 하는 것인데, 이는 정확도가 높은 검색 결과를 제공하는 데 도움을 준다.

- Hybrid Routing Approach : 메타데이터와 의미적 정보를 결합하여 라우팅하는 혼합 방식으로, 두 가지 접근 방식을 모두 결합해 정확성과 효율성을 모두 향상시키는 방법이다. 즉, 메타데이터 기반의 필터링과 의미적 정보 기반의 라우팅을 조화롭게 결합해 최적의 검색 결과를 도출해 내는데 도움을 준다.

D. Embedding

Retrieval Mechanism은 질문과 문서 청크의 임베딩 간 유사도를 계산하여 검색을 수행하는 체제로, 일반적으로 코사인 유사도와 같은 기법을 사용해 임베딩의 문맥적 유사성을 평가한다. 바로 여기서, 임베딩 모델의 의미적 표현 능력이 검색의 성능에 큰 영향을 미친다는 사실을 알 수 있다. 주요 임베딩 모델로는 다음과 같은 것들이 있다.

- Sparse Encoder (BM25) : 전통적인 방식으로, 단어 기반의 불일치 검색 모델
- Dense Retriever (BERT 기반 모델) : BERT와 같은 사전 학습된 언어 모델을 사용해 문서의 문맥적 의미를 효과적으로 포착하는 모델
- 최근 주요 연구 모델 : AnglE, Voyage, BGE 등 멀티태스크 튜닝을 통해 다양한 작업에서 성능을 향상시킨 임베딩 모델들

이러한 임베딩 모델들을 평가하기 위해서는 MTEB(Multilingual Text Embedding Benchmark; Hugging Face의 리더보드)를 사용하는데, 8개의 태스크와 58개의 데이터셋을 사용해 모델을 평가한다.

물론 어떤 임베딩 모델을 사용할 것인지에 대한 정답은 없으며, 각 모델의 특정 용도에 적합한 임베딩 모델을 선정하는 것이 필요하다. 예로, 전문적인 도메인 지식이 필요한 경우 해당 도메인에 맞게 튜닝된 모델이 더 효과적이다.

(1) Mix/hybrid Retrieval

희소(Sparse) Retrieval과 밀집(Dense) Retrieval을 결합해 상호 보완적인 정보를 활용하는 Retrieval로, 각 Retrieval의 장점을 모두 갖는다는 특징이 있다.

- Sparse Retrieval (BM25 등) : 빠르고 계산 비용이 적어 효율적이고, 희귀한 단어나 특수한 엔티티를 다룰 때 유리하다. 또 초기 검색 결과를 제공하기 때문에 밀집 검색 모델을 훈련하는 데 사용한다.
- Dense Retrieval (BERT 기반 모델 등) : 문맥적 의미를 잘 포착해 더 정확한 검색 결과를 제공하고, 희귀한 엔티티나 복잡한 질문에 대한 제로샷 검색을 수행할 수 있다.

즉 희소 검색 모델의 초기 검색을 통해 밀집 검색 모델에 제공되는 검색 결과를 향상시키고, 밀집 검색 모델은 희소 검색 모델의 한계를 보완해 더 정교한 검색 결과를 도출할 수 있도록 도와준다.

특히, 사전 학습된 언어 모델(PLMs)를 활용해 Sparse Retrieval을 향상시킬 수 있는데, 단어 가중치를 학습해 희소 검색 모델을 강화하고, 이는 결국 밀집 모델이 제공하는 의미적 깊이와 단어의 중요도를 이해하는 데 도움을 준다. 구체적으로, 희소 검색 모델이 제공하는 초기 검색 결과가 밀집 검색 모델이 더 나은 결과를 도출할 수 있도록 돕는(희귀한 엔티티나 특정 단어에 대해서) Zero-Shot Capability가 바로 그 특징이다.

(2) Fine-tuning Embedding Model

특정 분야(ex-헬스케어, 법률 등)에서는 기존 훈련 데이터와 상당한 차이가 있을 수 있는데, 특히 전문 용어가 많은 분야에서 모델의 성능을 높이기 위해서는 도메인 데이터셋으로 임베딩 모델을 미세 조정(fine-tuning) 하는 것이 중요하다. 즉, 모델이 해당 분야의 특수한 용어와 문맥을 이해하도록 돕고, 검색기(retriever)와 생성기(generator)를 함께 훈련시켜(정렬) 두 모델 간의 상호 연관성을 최적화해야 한다.

- LM-Supervised Retriever (LSR) : LLM을 검색기의 지도(supervised) 신호로 사용하여 검색기의 성능을 더욱 향상시키는 모델로, LLM 결과를 감독 신호로 사용해 검색기를 미세 조정한다. 예로 PROMPTAGATOR는 LLM을 소수 샷 쿼리 생성기로 활용해 작은 데이터셋에서도 효과적인 훈련이 가능하도록 돕는다.

- Dual-Signal Fine-Tuning : 데이터셋에서 제공되는 정확한 라벨(하드라벨)을 사용하고, LLM에서 제공되는 상호작용적 보상 신호(소프트 보상)를 사용하는 방법으로, 하드라벨을 사용해 정확도를 높이고 소프트 보상을 통해 문맥적 이해를 강화하는 것이다. (ex-LLM-Embedder)

- REPLUG 방법 : 검색기와 LLM을 결합하여 문서의 확률 분포를 계산하는 방법으로, KL Divergence(Kullback-Leibler Divergence)을 사용해 감독 학습을 수행, 이를 통해 검색기의 성능을 높이는 것이다. 장점으로는 크로스 어텐션 매커니즘 없이도 효과적인 훈련이 가능하다는 점이다.

- RLHF(Reinforcement Learning from Human Feedback) : 인간 피드백을 통해 검색기를 강화하는 방법으로, 여기서는 LLM을 강화학습의 보상 신호로 사용해 검색기를 훈련시킬 수 있다.

E. Adapter

LLM을 직접 API 기반으로 활용하려면 기능을 통합하는 과정이 복잡할 수 있고, 로컬 자원 한계로 인해 LLM을 직접 미세 조정하기 어려운 경우가 많다. 특히 대규모 모델의 경우 GPU 메모리 부족 등의 문제가 발생할 수 있다. 이에, 외부의 adapter를 활용해 이러한 문제를 해결하는 방법을 몇 가지 소개하도록 하겠다.

(1) UPRISE : Zero-Shot Prompt Retriever

가벼운 프롬프트 검색기를 훈련하여 사전 구축된 프롬프트 풀에서 적절한 프롬프트를 선택하는 방법으로, 다중 작업 환경에서 최적의 프롬프트를 자동으로 검색해 Zero-Shot 성능을 극대화한다.

(2) AAR(Augmentation-Adapted Retriever) : Universal Adapter

다양한 다운스트림 작업을 지원하는 ARR는 범용 어댑터를 사용하여 여러 개의 검색기와 생성 모델(generator)을 연결하는 역할을 수행한다. 이는 다양한 데이터셋과 태스크를 쉽게 적용할 수 있도록 설계돼 있다.

(3) PRCA : Reward-Driven Contextual Adapter

PRCA는 보상 신호를 기반으로 한 컨텍스트 어댑터를 플러그인 형태로 추가한 것으로, 특정 태스크의 성능을 향상시키도록 설계돼 정확한 문맥 정보를 선택하는 기능을 강화해준다.

(4) BGM : Bridge Seq2Seq Model

기존 검색기(retriever)와 LLM을 변경하지 않고, 중간에 Seq2Seq 모델을 추가해 검색된 정보를 LLM이 활용하기 쉽게 변환해준다. 주요 기능으로는 검색된 문서의 재랭킹(re-Ranking), 질문별 적절한 문서 선택의 동적 조정, 반복적인 검색 및 생성 전략 적용 가능성 등이 있다.

(5) PKG : Directive Fine-Tuning for Knowledge Integration

지시형(Directive) fine-tuning을 통한 화이트박스 모델(White-box LLM)로의 지식 통합 방법으로, PKG는 검색기 모듈을 직접 대체하여 질문에 맞는 문서를 생성하도록 개선한다.

4. GENERATION

검색 후에는 검색된 모든 정보를 그대로 LLM에 입력하여 질문에 답하도록 하는 것은 좋은 방식이 아니다. 이에 검색된 컨텐츠를 조정하는 방법과 LLM을 조정하는 방법에 대해 알아보도록 하겠다.

A. Context Curation(문맥 정제)

불필요한 정보는 LLM의 최종 응답 생성에 방해가 될 수 있으며, 지나치게 긴 문맥은 LLM이 “Lost in th middle” 문제를 일으킬 가능성이 있는데, 이는 인간과 유사하게, LLM이 긴 텍스트의 시작과 끝 부분에 집중하고 중간 부분을 잊어버리는 경향이 있기 때문이다. 따라서 RAG 시스템에서는 검색된 콘텐츠를 그대로 사용하지 않고 추가적인 정제 과정을 거쳐야 한다.

(1) Reranking (재정렬)

재정렬은 문서 조각 순서를 다시 정리하여 가장 관련성이 높은 결과를 우선적으로 배치하는 방식으로, 전체 문서 풀(pool)을 줄이고 정보를 더욱 정제하는 역할을 한다. 이는 정보 검색에서 향상(enhancer)과 필터링(filter)이라는 두 가지 기능을 수행하며, 언어 모델이 보다 정밀하게 처리할 수 있도록 정제된 입력을 제공한다.

재정렬은 사전 정의된 다양성(Diversity), 관련성(Relevance), MRR과 같은 지표를 기반으로 하는 규칙 기반 방법(rule-based methods) 또는 BERT 계열의 인코더-디코더 모델, Cohere rerank, bge-raranker-large와 같은 특화된 재정렬 모델, GPT와 같은 대형 언어 모델을 활용한 모델 기반 접근법을 통해 수행될 수 있다.

(2) Context Selection / Compression (문맥 선택/압축)

RAG 프로세스에서 흔히 발생하는 오해 중 하나는 바로 최대한 많은 관련 문서를 검색해 이를 연결하여 긴 검색 프롬프트를 만드는 것이 유리하다는 생각이다. 그러나 과도한 문맥(context)은 오히려 불필요한 노이즈를 증가시키고, LLM이 핵심 정보를 인식하는 능력을 저하시킬 수 있다.

LLMLingua는 GPT-2 Small 또는 LLaMA-7B와 같은 소형 언어(SLMs)을 활용해 중요하지 않은 토큰을 감지하고 제거한다. 이를 통해 인간에게는 이해하기 어려우나 LLM이 잘 이해할 수 있는 형태로 변환한다. 이 접근 방식은 추가적인 LLM 훈련 없이 언어의 완전성과 압축률을 균형 있게 유지하면서 프롬프트를 압축하는 직접적이고 실용적인 방법을 제공한다.

물론 문맥을 압축하는 것 외에도, 문서 수를 줄이는 것은 모델의 답변 정확도를 향상시키는 데 도움이 된다. Ma 등은 LLM과 SLM의 장점을 결합한 “Filter-Reranker” 패러다임을 제안했는데, 이 패러다임에서 SLM은 필터 역할을 하고, LLM은 문서를 재정렬하는 역할을 한다. 연구에 따르면, SLM이 식별한 어려운 샘플을 LLM이 재배치하도록 지시하면 다양한 정보 추출(IE) 작업에서 상당한 성능 향상이 이루어진다.

또 다른 직관적이고 효과적인 접근 방식은 바로 LLM이 최종 답변을 생성하기 전에 검색된 내용을 평가하도록 하는 것인데, 이를 통해 LLM이 관련성이 낮은 문서를 자체적으로 걸러낼 수 있다. 예를 들어, Chatlaw에서는 LLM이 참조된 법률 조항의 관련성을 평가하도록 자체 검토(self-suggestion)를 수행하도록 프롬프트를 설정한다.

B. LLM Fine-tuning

시나리오와 데이터 특성에 맞춘 맞춤형 fine-tuning은 LLM의 성능을 더욱 향상시킬 수 있다. 이는 온프레미스(on-premise; IT 시스템과 소프트웨어를 자체적인 물리 공간에서 직접 설치 및 운영하는 것) LLM을 사용할 때의 가장 큰 장점 중의 하나이다. 특정 도메인에 대한 데이터가 부족한 경우, 추가적인 지식을 미세 조정을 통해 LLM에 제공할 수 있다. 물론 Huggingface의 미세 조정 데이터도 초기 단계에서 활용될 수 있다.

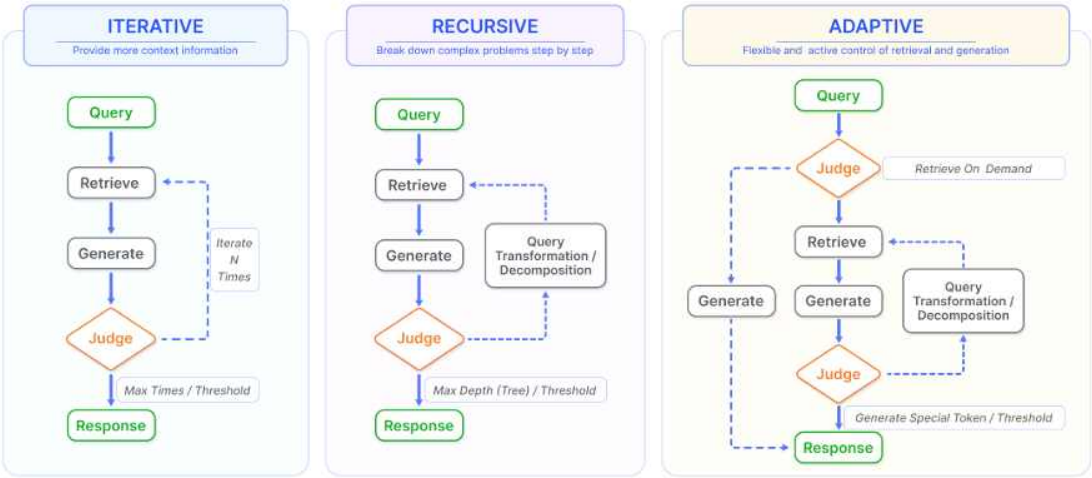
미세조정의 또 다른 장점은 모델의 입력과 출력을 조정할 수 있다는 점이다. 예를 들어, 특정 데이터의 형식에 맞게 LLM을 조정하고, 지시된 스타일에 따라 응답을 생성하도록 만들 수 있다. 하나의 예로 구조화된 데이터와 연관된 검색(retrieval) 작업의 경우, SANTA 프레임워크는 구조적, 의미적 특징을 효과적으로 반영하기 위해 3단계 학습 체계를 구현한다. 초기 단계에서는 검색기(retriever)에 집중하여, 대비 학습(contrastive learning)을 활용해 질의(query)와 문서 임베딩을 정제한다.

강화학습을 통해 LLM의 출력을 인간 또는 검색기의 선호도에 맞추는 것도 하나의 방법이다. 예를 들어, 최종 생성된 답변을 수동적으로 주석(annotation)하고, 이를 기반으로 강화학습을 통해 피드백을 제공할 수 있다. 인간의 선호도뿐만 아니라, 미세 조정된 모델과 검색기의 선호도의 맞추는 것도 가능하다. 강력한 독점 모델이나 대규모 파라미터를 가진 오픈소스 모델에 접근할 수 없는 경우, 보다 강력한 모델을 증류(distillation; 큰 모델에서 작은 모델로 지식을 전달하는 것)하는 것이 간단하면서 효과적인 방법이 될 수 있다.

또한 LMM의 미세 조정을 검색기의 미세 조정과 조정하여 선호도를 일치시킬 수도 있다. 예로, RA-DIT 같은 방법은 KL divergence을 활용해 검색기와 생성기 간의 점수 함수(scoring function)를 정렬한다.

5. Augmentation process in RAG

RAG 분야에서 일반적으로 적용되는 표준 방식은 단일 검색 후 생성하는 step이다. 그러나 이러한 방식은 비효율적일 수 있으며, 다단계 추론이 필요한 복잡한 문제에 대해서는 정보의 범위가 제한되기 때문에 충분하지 않을 때가 많다. 이러한 문제를 해결하기 위해 검색 프로세스를 최적화하려는 다양한 연구가 진행되었으며, 종류로는 다음과 같은 것들이 있다.



A. Iterative Retrieval

반복 검색(Iterative Retrieval)은 초기 질의와 지금까지 생성된 텍스트를 기반으로 지식 기반을 반복적으로 검색하여, LLM이 보다 포괄적인 지식 기반을 활용할 수 있도록 하는 과정이다. 이 접근 방식은 여러 번의 검색을 통해 추가적인 문맥 정보를 제공함으로써, 이후 생성되는 답변의 견고성을 향상시키는 것으로 입증되었다. 그러나 이 과정은 의미적 불연속성과 불필요한 정보의 누적 문제에 영향을 받을 수 있다.

ITERRETGEN은 검색 강화 생성(retrieval-enhanced generation)과 생성 강화 검색(generation-enhanced retrieval)을 결합한 시너지 접근 방식을 활용하여, 특정 정보를 재현해야 하는 작업에 적용된다. 이 모델은 입력된 작업을 해결하는 데 필요한 콘텐츠를 문맥적 기반으로 활용해 적절한 지식을 검색하며, 이를 통해 이후 반복에서 더욱 향상된 응답을 생성할 수 있도록 돕는다.

B. Recursive Retrieval

재귀 검색(Recursive Retrieval)은 정보 검색 및 자연어 처리(NLP)에서 검색 결과의 깊이와 관련성을 향상시키기 위해 자주 사용된다. 이 과정은 이전 검색에서 얻은 결과를 기반으로 검색 쿼리를 반복적으로 개선하는 방식으로 진행된다. 재귀 검색은 피드백 루프를 통해 점진적으로 가장 적절한 정보에 수렴함으로써 검색 경험을 향상시키는 것을 목표로 한다.

예로, IRCoT은 Chain-of-Thought 이하 Cot를 활용해 검색 프로세스를 안내하고, 검색 결과를 반영하여 CoT를 개선한다. ToC는 질의(Query) 내의 모호한 부분을 체계적으로 최적화하는 명확화 트리(clarification tree)를 생성한다. 이 방법은 사용자의 요구가 처음부터 명확하지 않거나, 찾고자 하는 정보가 매우 전문적이거나 미묘한 경우에 특히 유용하다. 이는 재귀적 특성을 통해 사용자의 요구에 지속해서 학습하고 적응할 수 있으며, 나아가 검색 결과에 대한 만족도를 높이는 데 기여한다.

특정 데이터 시나리오를 해결하기 위해서 재귀 검색과 다중 단계 검색(Multi-hop retrieval) 기법이 함께 사용되기도 한다. 재귀 검색은 계층적 방식으로 데이터를 처리하고 검색하기 위해 구조화된 인덱스를 활용하며, 이는 긴 문서나 PDF의 일부를 요약한 후 이 요약을 기반으로 검색을 수행하는 과정을 포함할 수 있다. 이후 문서 내에서 2차 검색을 수행하여 검색을 정제하며, 이러한 방식이 재귀적 검색 과정의 본질을 반영한다. 반면, 다중 단계 검색은 그래프 구조화된 데이터 소스를 더 깊이 탐색하여 상호 연결된 정보를 추출하는 데 초점을 맞춘다.

C. Adaptive Retrieval

Flare와 Self-RAG와 같은 적응형 검색(Adaptive Retrieval) 방법은 LLM이 검색을 수행할 최적의 시점과 검색할 내용을 능동적으로 결정할 수 있도록 하여 RAG 프레임워크를 개선한다. 이를 통해 검색된 정보의 효율성과 관련성을 높일 수 있다.

(1) WebGPT는 강화학습 프레임워크를 통합하여 GPT-3 모델이 텍스트 생성 중에 자율적으로 검색 엔진을 활용할 수 있도록 훈련한다. 이 모델은 검색 엔진 쿼리, 결과 탐색, 참고 문헌 인용 등의 작업을 수행하는 특수 토큰을 활용해 검색 프로세스를 진행하며, 외부 검색 엔진을 활용함으로써 GPT-3의 기능을 확장한다.

(2) Flare는 생성 과정의 신뢰도를 모니터링하여 검색 타이밍을 자동화하며, 이는 생성된 용어의 확률을 기준으로 판단된다. 특정 임계값 이하로 확률이 떨어지면 검색 시스템이 활성화되어 관련 정보를 수집함으로써 검색 주기를 최적화한다.

(3) Self-RAG는 “reflection-tokens”을 도입해 모델이 자신의 출력을 반성적으로 평가할 수 있도록 한다. 이 토큰은 retrieve와 critic 두 가지 유형으로 제공되며, 모델은 검색을 활성화할 시점을 자율적으로 결정하거나, 미리 정의된 임계값을 기준으로 검색을 실행할 수도 있다. 검색이 수행될 때 생성기는 여러 단락에 걸쳐 단편 수준의 빔 서치(Beam search; 제한된 집합에서 가장 유망한 노드를 확장해 그래프를 탐색하는 경험적 검색 알고리즘; greedy)를 수행하여 가장 일관된 시퀀스를 도출한다.

비평 점수(critic scores)는 세부 점수를 업데이트하는 데 활용되며, 추론 과정에서 가중치를 조정할 수 있는 유연성을 제공하며 모델의 동작을 최적화한다. Self-RAG는 추가적인 분류기나 자연어 추론(NLI) 모델에 의존할 필요 없이 검색 메커니즘을 활용할 시점을 스스로 결정할 수 있도록 설계되었으며, 이를 통해 모델의 자율적 판단 능력을 향상시키고 보다 정확한 응답을 생성할 수 있도록 한다.

6. TASK AND EVALUATION

자연어 처리(NLP) 분야에서 RAG의 급속한 발전과 확산은 LLM 커뮤니티에서 RAG 모델 평가를 연구의 핵심 과제로 부각했는데, 이 평가의 주요 목적은 다양한 응용 시나리오에서 RAG 모델의 성능을 이해하고 최적화하는 것이다.

A. Downstream Task

RAG의 핵심 작업은 여전히 질문 응답(Question Answering; QA)이며, 전통적인 단일 단계(single-hop) 및 다단계(multi-hop) QA, 객관식 문제, 도메인 특화 QA 뿐만 아니라 RAG에 적합한 장문 시나리오도 포함된다.

QA 외에도, RAG는 정보 추출(Information Extraction; IE), 대화 생성(dialogue generation), 코드 검색(code search) 등 다양한 다운스트림 작업으로 지속해서 확장되고 있다.

B. Evaluation Target

과거 RAG 모델의 평가는 특정 다운스트림 작업에서의 성능에 중점을 두었는데, 이러한 평가에는 해당 작업에 적합한 기존의 평가 지표를 사용한다. 예를 들어, QA 평가에서는 정확한 일치(Exact Match; EM) 및 F1 점수를 활용하며, 사실 확인(Fact-checking) 작업에서는 정확도(Accuray)를 주요 지표로 사용한다.

또한, BLEU 및 ROUGE 지표는 생성된 응답의 품질을 평가하는 데 널리 사용된다.

RAG 애플리케이션의 자동 평가를 위해 설계된 RALLE과 같은 도구들도 이러한 작업별 평가 지표를 기반으로 평가를 수행한다.

그럼에도 불구하고 RAG 모델의 고유한 특성을 평가하는 연구는 아직 부족한데, RAG 모델의 주요 평가 목표는 다음과 같다.

(1) Retrieval Quality (검색 품질)

검색 품질을 평가하는 것은 검색기(retrieval) 구성 요소가 제공하는 문맥의 효과성을 결정하는 데 필수적이다. 검색 엔진, 추천 시스템, 정보 검색 시스템 분야에서 사용되는 표준 지표들이 RAG 검색 모듈의 성능을 측정하는 데 활용된다.

이를 위해 일반적으로 적중률(Hit Rate), 평균 역순위(MRR), 정규화된 DCG(NDCG) 등의 지표가 사용된다.

(2) Generation Quality (생성 품질)

생성 품질 평가는 검색된 문맥을 바탕으로 일관되고 관련성 높은 응답을 생성하는 생성기

(generator)의 능력에 중점을 둔다. 이 평가는 콘텐츠의 목표에 따라 unlabeld 및 labeld 데이터로 구분될 수 있다.

- unlabeld content : 생성된 응답의 신뢰성(faithfulness), 관련성(relevance), 비유해성(non-harmfulness)을 평가한다.
- labeled content : 모델이 생성한 정보의 정확성(accuracy)에 초점을 맞춘다.

또한, 검색 및 생성 품질 평가는 수동(manual) 또는 자동(automatic) 평가 방법을 통해 수행될 수 있다.

C. Evaluation Aspects

최신 RAG 모델 평가 방식은 세 가지 주요 품질 점수와 네 가지 필수 능력을 강조하며, 이는 RAG 모델의 두 가지 핵심 목표인 검색(Retrieval)과 생성(Generation) 평가에 기여한다.

(1) Quality Scores (품질 점수)

품질 점수는 문맥 적절성(Context Relevance), 응답 신뢰성(Answer Faithfulness), 응답 적절성(Answer Relevance)으로 구성되며, 정보 검색 및 생성 과정에서 RAG 모델의 효율성을 다양한 관점에서 평가하는 데 사용된다.

- Context Relevance (문맥 적절성) : 검색된 문맥의 정확성과 구체성을 평가하며, 관련성을 보장하고 불필요한 콘텐츠로 인한 처리 비용을 최소화하는 데 기여한다.
- Answer Faithfulness (응답 신뢰성) : 생성된 응답이 검색된 문맥과 일치하는지 평가하며, 일관성을 유지하고 모순을 방지하는 역할을 한다.
- Answer Relevance (응답 적절성) : 생성된 응답이 질문과 직접적으로 관련이 있는지를 평가하며, 핵심 질문을 효과적으로 해결하는지를 측정한다.

(2) Required Abilities (필수 능력)

RAG 평가에서는 적응성과 효율성을 나타내는 네 가지 능력(노이즈 견고성, 부정 응답 거부, 정보 통합, 반사실 견고성)을 포함한다. 이러한 능력은 다양한 도전 과제와 복잡한 상황에서 모델의 성능을 결정하는 중요한 요소이며, 품질 점수에도 영향을 미친다.

- Noise Robustness (노이즈 견고성) : 질문과 관련이 있지만 실질적인 정보를 포함하지 않은 문서를 처리하는 모델의 능력을 평가한다.
- Negative Rejection (부정 응답 거부) : 검색된 문서에 질문을 해결할 필수 정보가 포함되지 않았을 때, 모델이 응답을 회피할 수 있는 능력을 평가한다.
- Information Integration (정보 통합) : 여러 문서에서 정보를 통합해 복잡한 질문에 응답하는 모델의 능력을 평가한다.
- Counterfactual Robustness (반사실 견고성) : 잘못된 정보가 포함된 문서를 인식하고 이를 무시할 수 있는 모델의 능력을 평가한다.

문맥 적절성과 노이즈 견고성은 검색 품질 평가에 중요하며, 응답 신뢰성, 응답 적절성, 부정 응답 거부, 정보 통합, 반사실 견고성은 생성 품질 평가에 중요한 요소이다.

D. Evaluation Benchmarks and Tools

RAG 평가를 용이하게 하기 위해 여러 벤치마크 테스트와 도구들이 제안되었는데, 이 도구들은 RAG 모델의 성능을 측정하는 정량적인 지표를 제공할 뿐만 아니라 다양한 평가 측면에서 모델의 능력을 이해하는 데 도움을 준다.

RGB, RECALL, CRUD와 같은 주요 벤치마크는 RAG 모델의 필수 능력을 평가하는 데 중점을 둔다. 동시에, RAGAS, ARES, TruLens8과 같은 최신 자동화 도구들은 LLM을 사용하여 품질 점수를 판별한다.

7. DISCUSSION AND FUTURE PROSPECTS

RAG 기술이 상당한 발전을 이뤘음에도 불구하고, 여전히 심층적인 연구가 필요한 여러 과제들이 존재하는데, 이번 장에서는 RAG가 직면한 현재의 주요 과제와 향후 연구 방향에 대해 소개한다.

A. RAG vs Long Context

현재 LLM은 컨텍스트 처리 능력이 지속적으로 확장되고 있음에 따라 200,000 개 이상의 토큰을 포함하는 컨텍스트도 원활하게 처리할 수 있다. 이러한 능력의 발전으로 인해, 기존 RAG에 의존해야 했던 장문 문서 기반의 질문응답(Long-Document QA)이 이제는 전체 문서를 직접 프롬프트에 포함할 수 있게 됐다.

이에 LLM이 컨텍스트 제약을 받지 않는다면 과연 RAG가 여전히 필요할 것인지에 대한 논의가 활발해지고 있는데, 실제로 RAG는 여전히 대체할 수 없는 중요한 역할을 수행한다.

1) LLM에 대량의 컨텍스트를 한꺼번에 제공하면 추론 속도가 크게 저하될 수 있다. 반면, RAG를 활용한 청크 기반 검색(chunked retrieval) 및 필요 시점(on-demand) 입력 방식은 운영 효율성을 크게 향상시킬 수 있다.
2) RAG 기반 생성 방식은 LLM이 원본 참고 자료를 빠르게 찾아 사용자가 생성된 응답을 검증할 수 있도록 지원한다.
3) RAG는 검색 및 추론 과정을 투명하게 관찰할 수 있도록 하지만, 장문 컨텍스트만을 활용한 생성 방식은 여전히 블랙박스로 남아있다.

반대로, 컨텍스트 확장은 RAG 발전에 새로운 기회를 제공하여, 보다 복잡한 문제를 해결하거나 방대한 자료를 읽고 통합 또는 요약해야 하는 질문에 대응할 수 있도록 하기 때문에, 초장문 컨텍스트 환경에서 새로운 RAG 방법론을 개발하는 것은 향후 중요한 연구 방향 중 하나가 될 것이다.

B. RAG Robustness (강건성)

검색 과정에서 노이즈 또는 상반된 정보가 존재할 경우, RAG의 출력 품질에 부정적인 영향을 미칠 수 있는데, 이러한 상황은 흔히 “잘못된 정보는 정보 부족보다 더 나쁠 수 있다”라는 표현으로 설명된다.

이와 같은 적대적(adversarial) 또는 반사실적(counterfactual) 입력에 대한 RAG 내성을 향상시키는 것이 중요한 연구 주제로 부상하고 있으며, 이는 주요 성능 평가 지표 중 하나로 자리 잡고 있다.

‘Cuconasu et al’는 어떤 유형의 문서를 검색해야 하는지 분석하고, 해당 문서의 프롬프트 관련성, 위치, 컨텍스트 내 포함 개수등을 평가했는데, 연구 결과에 따르면, 관련성이 낮은 문서를 포함하는 것이 오히려 정확도를 30% 이상 증가시킬 수 있는 것으로 나타났으며, 이는 품질 저하를 초래할 것이라는 기존의 가정을 뒤집는 결과였다. 이러한 결과는 검색과 언어 생성 모델을 효과적으로 통합하는 특화된 전략의 필요성을 강조하며, RAG의 강건성에 대한 추가 연구와 탐색이 필수적임을 시사한다.

C. Hybrid Approcaches

RAG와 파인튜닝을 결합하는 것이 주요 전략 중 하나로 부상하고 있는데, 그 방법에는 어떤 것들이 있는지(순차적, 교차적 또는 end-to-end 공동 학습), 그리고 파라미터 기반(parameterized) 및 비파라미터 기반(non-parameterized) 접근법의 장점을 어떻게 활용할 것인지에 대한 연구가 활발히 진행되고 있다.

또 다른 연구 흐름은 특정 기능을 갖춘 소형 언어 모델(SLMs)을 RAG에 도입하고, RAG 시스템의 결과를 기반으로 파인튜닝하는 것이다.

예를 들어, CRAG는 경량화된 검색 평가 모델을 학습시켜 쿼리에 대한 검색 문서의 전반적인 품질을 평가하고, 신뢰도(confidence level)에 따라 다양한 지식 검색 작업을 트리거하는 방식을 사용한다.

D. Scaling Laws of RAG

end-to-end RAG 모델과 RAG 기반 사전 학습 모델은 여전히 연구자들의 주요 관심사 중 하나이다. 이러한 모델의 성능을 결정짓는 핵심 요소 중 하나는 바로 파라미터 수인데, LLM에 대한 스케일링 법칙은 정립돼 있지만, 이를 RAG에 그대로 적용할 수 있을지는 아직 불확실하다.

‘RETRO++’와 같은 초기 연구들이 이 문제를 다루기 시작했지만, RAG 모델의 파라미터 수는 여전히 LLM에 비해 적은 편이고, 오히려 작은 모델이 더 큰 모델보다 성능이 뛰어난 역스케일링 법칙(Inverse Scaling Law)의 가능성이 제기되고 있다.

E. Production-Ready RAG

RAG의 실용성과 엔지니어링 요구사항의 정렬(alignment)이 필수로 다가오는 현 동향에서, 검색 효율성 향상, 대규모 지식 베이스에서의 문서 리콜 개선, 그리고 LLM이 문서 출처나 메타데이터를 실수로 노출하는 것을 방지하는 등의 데이터 보안 확보는 여전히 해결해야 할 중요한 엔지니어링 과제이다.

RAG 생태계의 발전은 그 기술 스택의 진보로 크게 영향을 받는다. LangChain과 LLamaIndex와 같은 핵심 도구들은 ChatGPT의 등장과 함께 빠르게 인기를 얻었으며, 광범위한 RAG 관련 API를 제공해 LLM 분야에서 필수적인 요소가 되었다.

새롭게 부상하는 기술 스택(technology stack)은 LangChain이나 LLamaIndex만큼 기능이 풍부하지는 않지만, 특화된 제품을 통해 차별성을 갖는다. 예로 Flowise AI는 로우 코드(low-code) 방식을 우선시하여 사용자들이 드래그 앤 드롭 인터페이스를 활용해 RAG를 포함한 AI 애플리케이션을 쉽게 배포할 수 있도록 한다. 또한, HayStack, Meltano, Cohere Coral과 같은 기술들도 각각의 독창적인 기여로 인해 주목받고 있다.

AI 중심의 벤더뿐만 아니라, 기존 소프트웨어 및 클라우드 서비스 제공업체들도 RAG 중심의 서비스를 확대하고 있다. 예로, Weaviate의 Verba는 개인 비서 애플리케이션을 위해 설계되었으며, Amazon의 Kendra는 지능형 엔터프라이즈 검색 서비스를 제공하여 사용자가 내장된 커넥터를 활용해 다양한 콘텐츠 저장소를 탐색할 수 있도록 지원한다.

RAG 기술 개발에서는 다음과 같은 여러 특화 방향으로의 명확한 발전 경향이 나타나고 있다.

1) Customization : 특정 요구사항에 맞춰 RAG를 조정하는 방식
2) Simplification : RAG의 사용을 더 쉽게 만들어 초기 학습 곡선을 줄이는 방식
3) Specialization : RAG를 최적화하여 실제 프로덕션 환경에서 더 효과적으로 활용할 수 있도록 하는 방식

RAG 모델과 그 기술 스택의 상호 발전은 명확하게 드러나고 있는데, 기술 발전은 기존 인프라에 대한 새로운 표준을 지속해서 수립하고 있으며, 이에 따라 기술 스택의 형상이 RAG 기능의 발전을 견인하고 있다.

현재 RAG 도구들은 점점 하나의 기본 기술 스택으로 통합되어 가고 있으며, 이는 향후 고급 엔터프라이즈 애플리케이션의 기반을 마련하고 있다. 그러나 완전히 통합된 종합적인 플랫폼 개념은 아직 미래의 과제로 남아있으며, 이를 실현하기 위해서는 추가적인 혁신과 개발이 필요하다.

F. Multi-modal RAG

RAG는 초기의 텍스트 기반 질의응답(QA) 모델에서 벗어나 다양한 모달 데이터를 통합하는 방향으로 발전하고 있다. 이러한 확장은 RAG 개념을 다양한 도메인에서 활용하는 혁신적인 멀티모달 모델을 탄생시켰다.

(1) 이미지 기반 RAG

a) RA-CM3 : 텍스트 및 이미지의 검색과 생성을 모두 수행하는 최초의 멀티모달 모델

b) BLIP-2 : 고정된 이미지 인코더와 LLM을 결합하여 효율적인 비주얼-언어 사전 학습을 수행하며, 이를 통해 제로샷 이미지-텍스트 변환을 가능하게 함.

c) 'Visualize Before You Write' : 이미지 생성을 통해 LM의 텍스트 생성을 유도하는 방법으로, 개방형 텍스트 생성 작업에 유망한 결과를 보임

(2) 오디오 및 비디오 기반 RAG

a) GSS 방법 : 오디오 클립을 검색하고 연결하여 기계 번역 데이터를 음성 번역 데이터로 변환

b) UEOP : 외부 오프라인 전략을 활용해 음성-텍스트 변환을 수행하는 end-to-end 자동 음성 인식(ASR) 모델

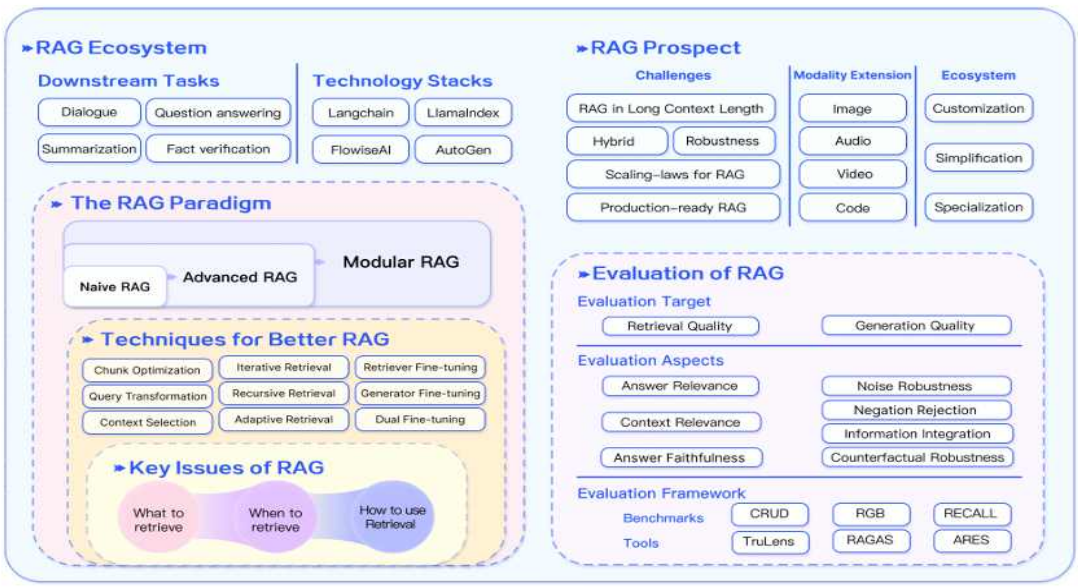
c) KNN 기반 Attention Fusion : 오디오 임베딩과 의미적으로 관련된 텍스트 임베딩을 결합하여 ASR 성능을 향상하고, 도메인 적응을 가속화함.

d) Vid2Seq : 언어 모델에 특화된 시간 마커를 추가하여 사건 경계 및 텍스트 설명을 통합된 출력 시퀀스로 예측하는 모델

(3) 코드 기반 RAG

a) RBPS : 코드 예제를 검색하여 개발자의 목표에 맞는 사례를 제공하며, 주로 테스트 어설션 생성 및 프로그램 수정을 지원하는 소규모 학습(task)에서 효과적임

b) CoK 방법 : 지식 그래프에서 입력 쿼리와 관련된 사실을 먼저 추출한 후, 이를 힌트로 활용하여 지식 그래프 QA 성능을 향상시킴



8. CONCLUSION

RAG가 외부 지식 베이스의 비모수적(non-parameterized) 데이터와 언어 모델(LLM)의 모수적(parameterized) 지식을 통합해 LLM의 성능을 획기적으로 향상시켰는데, 본 연구는 RAG 기술의 발전과 다양한 과제에서의 적용 사례를 다루며, RAG 프레임워크 내에서 Naive, Advanced, Modular RAG의 세 가지 발전 패러다임을 분석했다.

RAG는 파인튜닝 및 강화학습과 같은 AI 기법들과 결합하면서 그 기능이 더욱 확장되었는데, 장기 문맥(long context) 처리 능력과 강건성(robustness)을 개선해야 할 연구 과제가 여전히 남아있다.

RAG의 적용 범위는 multi-modal 도메인까지 확대되고 있으며, 이를 통해 이미지, 비디오, 코드와 같은 다양한 데이터 유형을 해석하고 처리할 수 있도록 진화하고 있다. 이러한 확장은 AI 기술의 실용성을 더욱 강화하며, 학계 및 산업계에서 RAG에 대한 관심이 증가하는 배경이 되고 있다.

RAG의 생태계는 점점 더 성장하고 있으며, RAG 기반 AI 애플리케이션의 증가 및 지속적인 RAG 관련 도구 개발이 이를 뒷받침하고 있다. RAG의 응용 분야가 넓어짐에 따라, 그 성능을 더 정확하게 평가할 수 있는 방법론을 개선할 필요가 있다. 이는 RAG가 AI 연구 및 개발 커뮤니티에서 기여하는 바를 더 명확히 포착하기 위해 필수적이다.