[summary]검색 증강 생성(Retrieval-Augmented Generation, RAG)

LLM의 한계

기존 LLM은 사용자의 입력을 받아 학습된 데이터를 기반으로 답변을 생성한다. 사용자가 학습되지 않은 질문을 하면, 보유한 데이터 중에 확률이 가장 높은 정보를 조합하여 답변을 생성한다. 이 과정에서 허위 정보 혹은 오래된 정보를 사실인 것처럼 제공하는 환각 현생(할루시네이션)이 발생 할 수 있다.

- 최근 지식, 구체적인 분야 전문지식이 없다.
- 거짓말을 잘한다(할루시네이션)

⇒ 답변이 그럴듯해 보여도 실제로는 도움이 되지 않거나 허위 혹은 잘못된 정보를 제공

LLM의 한계 보완 방법

- fine tuning: 사전 학습 모델(pre-trained model)에 특정 도메인(예: 의료, 법률, 금융)의 데이터를 추가 학습시켜 모델을 최적화하는 방식

- RAG: RAG는 Retrieval-Augmented Generation의 약자로, 정보 검색과 생성 모델을 결합한 자연어 처리(NLP) 기술로, 외부 지식 소스와 연계하여 모델의 범용성과 적응력을 유지하면서도 정확하고 신뢰할 수 있는 답변을 생성

항목	RAG	파인튜닝
사용 데이터	외부 데이터 검색	사전 학습 데이터
시간·비용 소모	적음	많음
베이스 모델 개선	불가	가능
환각 현상 발생 가능성	낮음	사전 학습되지 않은 데이터에서 발생 가능성 있음
데이터 변동성	역동적 데이터	정적 데이터
의사 결정 과정	검색된 문서 확인 가능	알 수 없음

[표 1] RAG와 파인튜닝 비교 (출처: Skelter Labs, 재구성: 이글루코퍼레이션)

Retrieval-Augmented Generation

R:

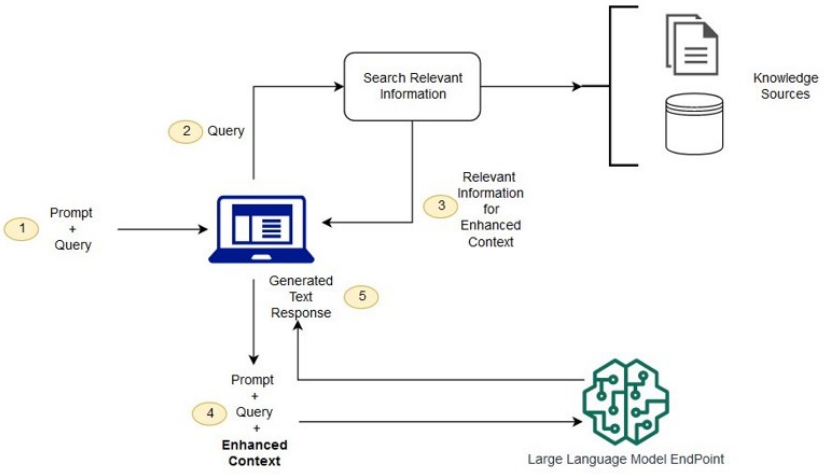
A

G

RAG 장점

1. **Fine tuning에 비해 시간과 비용이 적게 소요**
   - 외부 데이터베이스를 활용하기 때문에 별도의 학습 데이터를 준비할 필요가 없음
2. **모델의 일반성을 유지**
   - 특정 도메인에 국한되지 않고 다양한 분야에 대한 질문에 답변할 수 있다.
3. **답변의 근거를 제시할 수 있다.**
   - 답변과 함께 정보 출처를 제공하여 답변의 신뢰도를 높일 수 있다.
4. **환각 현상 가능성을 줄어듦.**
   - 외부 데이터를 기반으로 답변을 생성하기 때문에 모델 자체의 편향이나 오류를 줄일 수 있다.

RAG의 작동 방식



1. **질문 입력**: 사용자가 질문을 입력하면, 이 질문은 먼저 RAG 시스템으로 전달된다.
2. **검색**: RAG의 검색 부분은 질문에 관련된 정보를 외부 데이터베이스에서 실시간으로 검색한다. 이는 최신 정보와 특정 도메인 지식을 포함한다.
3. **결합 및 생성**: 검색된 정보를 바탕으로 LLM이 답변을 생성한다. 이 과정에서 LLM은 검색된 문서의 내용을 이해하고 이를 바탕으로 자연스럽고 정확한 답변을 제공한다.
4. **출력**: 최종 답변은 사용자가 이해하기 쉽게 후처리 과정을 거쳐 출력된다.

RAG 구조

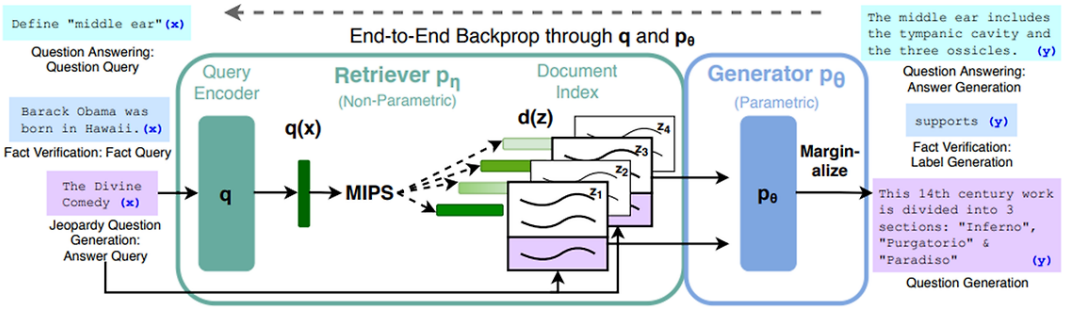


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Retriver와 Generator로 구성

- **Retriver**: 데이터셋으로 부터 질문과 관련된 최근 문서를 찾는 모듈
- **Generator:** 쿼리와 Retriever가 구한 문서들을 같이 받아 output을 생성

쿼리 임베딩 모델q, 문서 임베딩 모델d를 이용해 쿼리와 문서를 임베딩하고, 쿼리와 내적한 값이 높은 문서를 찾음.
내적한 값이 가장 높은 문서 n개 찾아서(설정가능) 개별 문서 뒤에 쿼리를 붙여 Generator에 n번 입력함.

Generator output도 n개가 나올 것이고 이 중 소프트맥스 확률 값이 가장 높은 결과를 사용

RAG에서 중요한 것: Retrieval

RAG는 비용이나 시간적인 문제로 LLM을 Fine tuning하기 어렵기 때문에 선택하는 경우가 많다. 따라서 같은 LLM을 사용하더라도 inference 성능을 잘 끌어내야함.

따라서 RAG에서 통제할 수 있는 것은 Retrieval로 가져오는 문서 뿐

1) 어떻게 문서를 잘 가져올 것이며

2) 가져온 문서들을 쿼리와 조합하여 LLM에 어떻게 넘길 것인지가 중요

문서를 잘 가져오는 방법

- **Data structure**: PDF와 같은 semi-structured data나 Knowledge graph 같은 structured data도 data source로 사용할 수 있는지

- **Retrieval Granularity**: retrieval 단위를 토큰, Phrase, Sentence, Chunk, Document 등 어떻게 잡을지
- **Indexing Optimization**
- **Query Optimization**
- **Embedding**

참고 블로그

https://devbasket.tistory.com/77

https://medium.com/rate-labs/rag의-짧은-역사-훑어보기-첫-논문부터-최근-동향까지-53c07b9b3bee