Scientific Knowledge Question Answering

Information Retrieval 8조

목차

1.Team Introduction

김민준

Personal information

- 컴퓨터공학과 학사
- 3년차 ML Engineer

Project Role

- Langchain기반 베이스라인 제작
- Embedding Model 선정 및 parameter search를 위한 정량평가 파이프라인 구현
- Ensemble Retriever
- Query Ensemble
- Contextual Retrieval

조수한

Personal information

- 컴퓨터공학 학사

Project Role

- 영어 번역 후 Ensemble Embedding 비교 실험
- Retrival인 Faiss, Elastic Search, colbert, Milvus 비교실험
- HyDE 실험
- CrossEncoder, Gpt reranker 비교 실험

안수민

Personal information

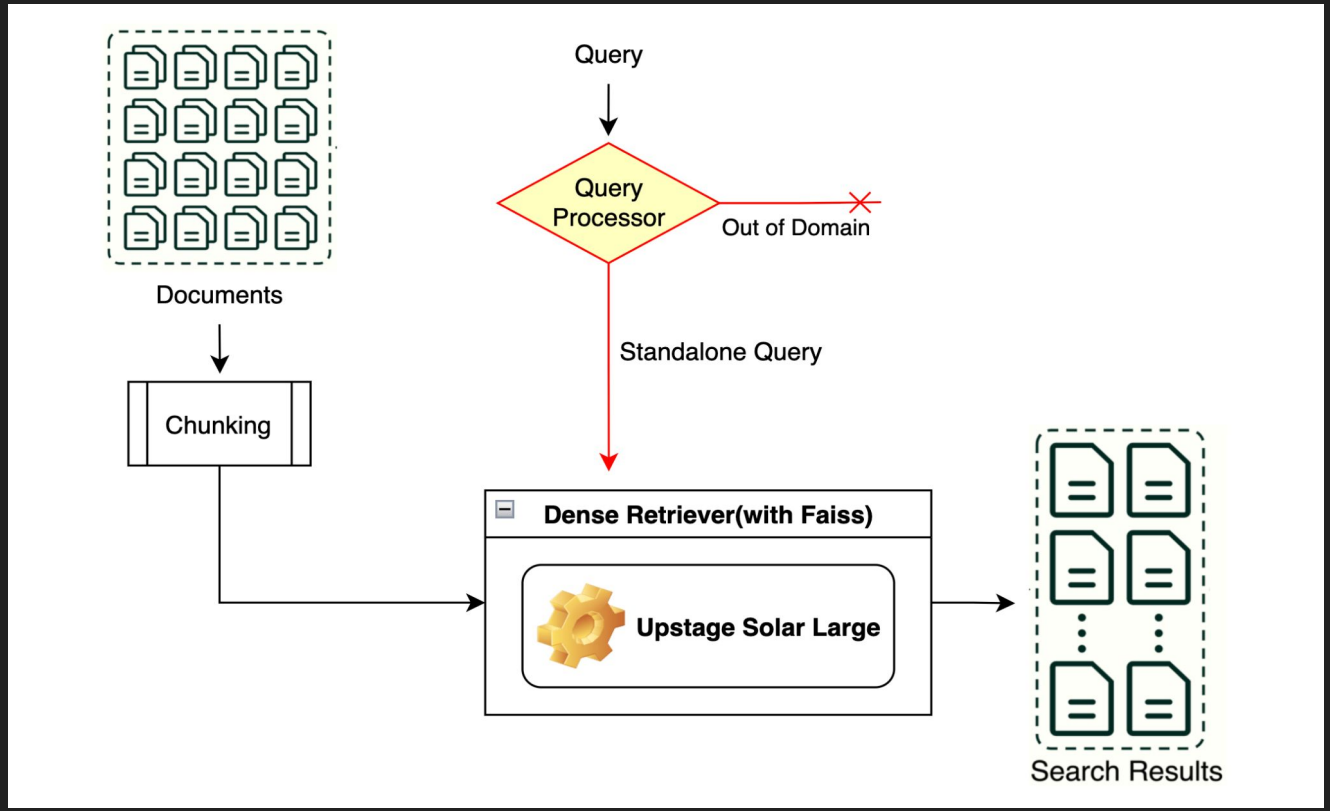
- 디지털미디어학
- 2년차 웹 개발

Project Role

- LLM Query Expansion, sparse 비교

2.Baseline

Baseline Flow chart



2-1.Query process

- 쿼리를 LLM으로 전처리 후 검색엔진에 입력.
  a. Multi-turn query인 경우 전체 대화를 기반으로 하나의 질의 문장을 생성
     - `[{"role": "user", "content": "기억 상실증 걸리면 너무 무섭겠다."},`
       `{"role": "assistant", "content": "네 맞습니다."},`
       `{"role": "user", "content": "어떤 원인 때문에 발생하는지 궁금해."}]}`
     - `{"eval_id": 107, "query": "기억 상실증의 원인은 무엇인가요?"}`
  b. 생성된 질의가 과학적인 상식에 관한 질문인지 판단.
     - `{"eval_id": 276, "query": "요새 너무 힘들다."}`
     - 이처럼 상식에 대한 질의가 아니라면 “답변을 할 수 없습니다"를 반환하도록 구현.

2-2.Chunking

- 문서별 글자단위로 길이 분석.
  - **최소 길이**: 44
  - **최대 길이**: 1230
  - **평균 길이**: 315
  - **중앙값 길이**: 299

- 평균 및 중앙값이 약 300. ➡️ chunking이 필요한지 실험.
  - Chunking Parameter Search : Chunk_size, Overlap_size
  - Embedding model 선정 단계에서 parameter search 수행.
    - chunk_size:100
    - overlap_size: 50

2-3.Embedding Model 선정

- 어떤 모델이 (한국어)임베딩을 잘할까?
  - 데이터셋에는 label이 없다.
  - 즉, 모델의 성능을 측정하려면 리더보드 점수 밖에 없다.
- 대략적인 성능이라도 파악해보자.
  - 문서마다 10개의 질문을 LLM(gpt-4o)이 생성.
  - 문서의 내용과 적절한 관련성을 갖는 질의임을 최소한으로 보장 받기 위해 GEval 3점 이상인 질의를 최대 3개 사용.
  - 생성된 질문은 문서와 동일한 docid를 포함.
  - 대회 평가 지표와 동일한 방식(MAP)으로 검색 결과를 평가.

2-3.Embedding Model 선정

평가결과

Model	mAP
upstage/solar-embedding-1-large	0.9152
dragonkue/bge-m3-ko	0.9093
nlpai-lab/KoE5	0.8946
BAAI/bge-m3	0.8933
intfloat/multilingual-e5-large-instruct	0.8417
OpenAI/text-embedding-3-large	0.8288

2-4. Sparse vs Dense

- Sparse 방식과 Dense 방식 중 어떤 방식이 더 좋은 성적을 내는지 평가.
- 대회에 적합한 방식을 찾기 위한 것이므로 리더보드 점수를 평가 기준으로 채택.

Model	mAP(리더보드)
upstage/solar-embedding-1-large	0.9197
intfloat/multilingual-e5-large-instruct	0.8417
Okt-BM25	0.7882
Kiwi-BM25	0.7651

2-5.Query Expansion

- 검색에 부적합하다고 판단되는 쿼리들
  - `{"eval_id": 280, "query": "Dmitri Ivanovsky가 누구야?"}`
    “Dimitri Ivanovsky”가 아닌 “드미트리 이바노프스키"인 문서가 존재.
  - `{"eval_id": 217, "query": "오세아니아 섬나라에 광견병이 있는지 여부"}`
    오세아니아는 대륙이라는데 “오세아니아 섬나라" 는 무엇을 말하는 것일까?
  - `{"eval_id": 38, "query": "목성 trojan의 특징에 대해 알려줘."}`
    “목성 trojan”??
  - `{"eval_id": 46, "query": "B-형 간염에 대해 알려줘."}`
    “B-(마이너스) 간염”일까 “B형 간염”일까

2-5.Query Expansion

- LLM(gpt-4o)을 이용한 query expansion

  1.질문 의도를 파악한 후, 이를 더욱 명확히 하도록 개선.

  - Before : **나무의 분류에 대해 조사해 보기 위한 방법은?**
  - After : **식물학 및 생물학에서 나무의 분류 체계와 분류 방법을 조사하는 방법은 무엇인가요?**
  - mAP 0.9000로 감소. 쿼리가 지나치게 구체화?

  2.핵심 단어들만 추출해서 더 간단명료하게 개선.

  - Before : **나무의 분류에 대해 조사해 보기 위한 방법은?**
  - After : **나무를 분류하는 방법과 조사 방법**
  - mAP 0.9030로 감소. 원인 해석 불가.

2-5.Query Expansion

- Human query expansion
  - Clear

```
{"eval_id": 269, "query": "식물이 높이 자랄 수 있게 하는 메커니즘이 궁금해."}
{"eval_id": 269, "query": "식물이 높이 성장하기 위한 환경적인 조건은 무엇인가요?"}

{"eval_id": 68, "query": "python 공부중인데... 숫자 계산을 위한 operator 우선순위에 대해 알려줘."}
{"eval_id": 68, "query": "파이썬 프로그래밍에서 숫자 계산을 위한 연산자 우선순위를 알려주세요."}

{"eval_id": 309, "query": "특정 농도의 황산 sample을 만드는 방법?"}
{"eval_id": 309, "query": "특정 농도의 황산(Sulfuric Acid, H2SO4) 샘플을 만드는 방법을 알려주세요."}
```

  - Unclear

```
{"eval_id": 284, "query": "개인의 생물학적, 사회적인 특성 형성에 영향을 미치는 요소에 대해 설명해줘."}
{"eval_id": 307, "query": "강아지가 사회화되는 행동의 사례는 뭐가 있을까?"}
```

  - mAP 0.9197로 expansion을 적용하지 않았을 때와 동일.
  - 원본 쿼리의 의도를 유지하면서 불필요한 정보를 담지 않고자 했지만 생각보다 많이 까다로운 작업.

2-6.Document Expansion

- 문서 내용을 기반으로 여러가지 정보들을 LLM으로 생성.
- 추가된 정보들이 noise로 적용되어서인지 성능은 오히려 감소.

"<제목>"
"에너지 균형을 유지해 건강 관리하기"

"<본문>"
"건강한 사람이 에너지 균형을 평형 상태로 유지하는 것은 중요합니다 …"

"<요약>"
"건강한 사람이 에너지 균형을 1-2주 동안 유지하기 위해 영양가 있는 …"

Document	mAP
원본	0.9197
제목 + 원본	0.8848
요약 + 원본	0.8955
질문(3개) + 원본	0.8553
제목 + 원본 + 요약 + 질문	0.9000

3.Problems

3-1.실험 실패 원인?

- 실험별로 “왜 성능이 향상되지 않는지” 객관적인 해석이 어렵다. ➡️ 모든 해석이 주관적.
  - Query, Document Expansion
    - 정보가 너무 많이 손실 되었다.
    - 불필요한 정보가 너무 많다.
  - Reranker
    - 한국어 성능이 낮다.

3-2.결과 정성평가

쿼리와 의미적으로 유사하지만 핵심 단어가 다른 문서들이 검색되고 있다.

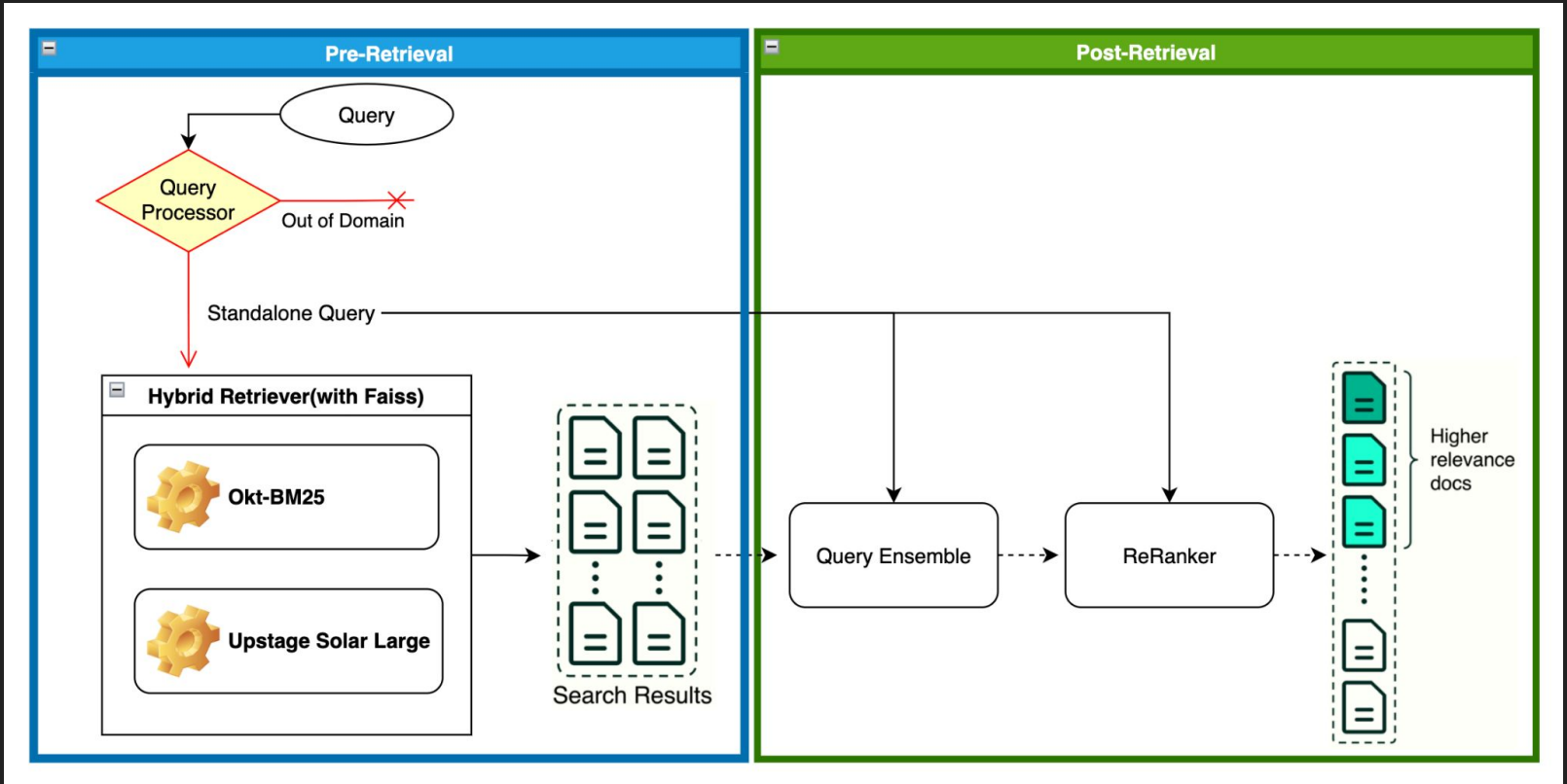
{"eval_id": 213, "query": "각 나라에서의 공교육 지출 현황에 대해 알려줘."}

```
  검색결과
  Rank1
   ------------------------------------------------------------------------
    - DocID: 79c93deb-fe60-4c81-8d51-cb7400a0a156
    - Score: 1.0000
    - Content: 2017년 현재, 전세계의 공공 교육 지출은 세계 GDP의 약 4%를 차지하고 있습니다.
하고, 인구의 교육 수준을 높여 경제적인 경쟁력을 강화하고자 합니다. 따라서 많은 국가들이 교육
니다. 공공 교육 지출은 국가의 교육 정책과 우선순위를 반영하며, 국가의 교육 수준과 경제 성장

  Rank2
   ------------------------------------------------------------------------
    - DocID: 4f0ea39c-f16c-4d6a-9709-d106c66afcc9
    - Score: 0.9711
    - Content: 세계 의료비 총지출은 2017년 현재 세계 GDP의 약 10%를 차지하고 있습니다. 이는
다. 세계 의료비 총지출이 GDP의 10%에 이르는 것은 의료 분야에 대한 긍정적인 투자와 의료 기술
```

4.개선시도

Flow Chart


Pre-Retrieval
Query
Query Processor
Out of Domain
Standalone Query
Hybrid Retriever(with Faiss)
Okt-BM25
Upstage Solar Large
Search Results
Post-Retrieval
Query Ensemble
ReRanker
Higher relevance docs

4-1.Ensemble Retriever

- “복숭아 키우는 노하우 좀" 과 같이 특정 대상이 명시되는 경우 BM25가 효과적일 것.
- BM25를 벡터 임베딩과 하이브리드로 검색하는 Ensemble Retriever를 적용.
  - Langchain은 RRF(Reciprocal Rank Fusion) 방식.
    - 검색된 문서들의 유사도로 정렬 후, 낮은 순위일수록 큰 가중치를 부여하고 합산.
  - CC(Convex Combination) 방식은 구현되어 있지 않으므로 별도로 구현.
    - a * BM25_score + (1 - a) * cos_sim
  - Sparse, Dense 각각에 적용될 가중치는 parameter search로 탐색, [0.4, 0.6]로 설정.
  - mAP가 약 0.1~0.2 증가

Method	mAP	MRR
RRF	0.9212	0.9288
CC	0.9379	0.9288

4-2.Contextual Retrieval

- 문서를 청크로 나누었을 때 쿼리와 관련성이 높은 청크는 소수. ➡️ 대부분의 청크는 질의와 관련성이 적다.
- Contextual Retrieval은 원본 문서와 청크를 LLM에 입력해서 청크에 문서의 문맥 정보를 간략히 채워주는 방식.

Before : “회사의 수익은 지난 분기 대비 3% 증가했습니다.”

‘회사'는 어떤 회사인지, ‘지난 분기'가 정확히 언제인지 알 수 없다.

After : ""“회사의 수익은 지난 분기 대비 3% 증가했습니다.

이 청크는 2023년 2분기 ACME corp의 실적에 대한 SEC 제출 자료에서 가져온 것입니다. 전 분기 매출은 3억 1,400만 달러였습니다. 회사의 매출은 전 분기 대비 3% 증가했습니다.”

- chunking으로 만들어진 24799개의 chunk별로 contextual retrieval을 적용.

4-2.Contextual Retrieval

- 프롬프트를 고도화할수록 성능이 좋아진다.
  - version1 : normal
  - version2 : 청크에 부족한 점을 먼저 찾은 후, 정보 추가하기
  - version3 : few-shot 적용(ex.현재 청크에는 “회사명", “정확한 날짜" 정보가 없습니다.)
  - version4 : 제목, 요약 추가 ➡️ 과도한 정보 추가는 역효과

	mAP	MRR
UP-ER-QEN-CRV1	0.9424	0.9455
UP-ER-QEN-CRV2	0.9470	0.9500
UP-ER-QEN-CRV3	0.9515	0.9545
UP-ER-QEN-CRV4	0.9045	0.9091

4-3.Query Ensemble

- 쿼리와 Pre-Retrieval 단계에서 검색된 문서들을 여러 모델로 임베딩.
- 다양한 값의 벡터로 임베딩되고 이를 종합하여 성능을 증가시키는 아이디어.
- 모델별로 서로 다른 가중치 적용 가능. ➡️ Parameter Search

$$\text{combined\_similarity}(d_j) = \sum_{i=1}^{n} w_i \cdot \text{cosine}(q_i, d_j)$$

Model	Ensemble Weight
BAAI/bge-m3	0.1
dragonkue/bge-m3-ko	0.3
solar-embedding-1-large	0.6

4-4.Reranker

- 검색된 문서들의 순위를 재조정.
- pre-retrieval 단계에서 10개의 문서들을 추출하고, reranker에 전달.
- reranker는 top3를 선택해 반환.
- 몇가지 Reranker 모델들을 활용해봤지만 점수는 향상되지 않았다.

Model	Ensemble Weight
Cohere	0.9152
Voyage	0.9144
Dongjin-kr/ko-reranker	0.9076

고려사항

- 성능개선은 분명했으나 단점도 존재한다.
  - Parameter Search로 적절한 weight, 모델들의 조합을 탐색하는 시간 필요.
  - Query Ensemble
    - GPU 사용량 증가
    - 앙상블 모델별로 전체 문서에 대한 임베딩을 미리 구해야하므로 실행시간 증가.
  - Contextual Retriever는 문서와 파생된 각각의 청크를 LLM에 입력.
    - 성능이 좋은 LLM일수록 퀄리티가 좋지만 그만큼 비싸다.
    - Anthropic의 Claude는 prompt cachin 기능으로 문서를 캐싱하여 청크에 대한 비용만 지불.
    - 그러나 RPM(Requests Per Minutes) 제한에 의해 강제로 대기 시간을 부여하므로 시간 소모가 더 크다.
    - 처리해야할 데이터양 자체가 많기 때문에 시간, 비용 소모가 크다.

5.Review

리뷰

- 김민준
  - 모르는 것을 편하게 알아보기 위해 LLM을 사용하다가 서비스 관점에서 활용할 수 있다는 점이 굉장히 재밌었다.
  - 성능개선으로 연결되지 못한 방법들이 어떤 이유 때문인지 분석하고 해결하지 못해 아쉽다.
- 조수한
  - LLM의 성능을 올리기 위한 다양한 방법과 요즘 트랜드인 RAG을 경험할 수 있어 좋은 경험이었습니다.
  - Prompt을 잘 만드는 것이 중요하다는 것을 알았습니다.
- 안수민
  - 한 것 대비 제일 많이 배운 기간이었습니다.

6.Q&A

Reference

- Langchain https://python.langchain.com/docs/versions/v0_3/
- Contextual Retrieval, Ensemble Retriever https://www.anthropic.com/news/contextual-retrieval
- Query Ensemble https://www.kaggle.com/competitions/kaggle-llm-science-exam/discussion/446358
- Convex Combination
  - paper https://arxiv.org/pdf/2210.11934
  - Code https://github.com/Marker-Inc-Korea/AutoRAG/blob/main/autorag/nodes/retrieval/hybrid_cc.py