
Scientific Knowledge Question Answering |

과학 지식 질의 응답 시스템 구축

Information Retrieval Team 2. 전지재능

목 차

01

팀 소개

02

대회 소개 & 데이터 탐색

EDA
전처리

03

IR 파이프라인 구축

Embedding
Standalone_query
Elasticsearch indexing
Retrieval
Reranking
LLM

04

인사이트

05

프로젝트 결과

06

회고

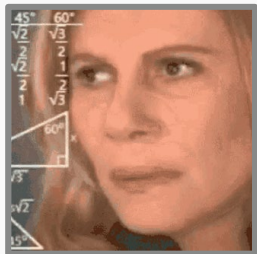
07

Q&A

01

팀원 소개

팀원소개



이윤재

Elasticsearch
Reranker training



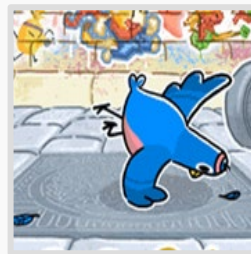
이재명

ColBERT
Embedding



장은지

ColBERT
CrossEncoder



전백찬

Chunking
Faiss



최지미

Preprocessing
Prompt Engineering

협업방식



GitHub

코드 공유



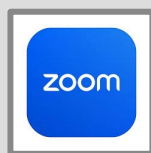
Google
스프레드시트

실험 기록



Google
프레젠테이션

발표 자료 작성



zoom

미팅



Slack

실시간 소통

- 미팅 일정 : 매일 오전 11시
- 기록 : 협업 툴을 이용하여 기록 및 공유

현황 공유

현황 기록 후 미팅 시 활용하여 아이디어 발굴

	<p>onlypositivepair로 roberta학습시킨 경우 성능이 차이가 없었음</p> <p>spacy로 ner방식을 도입해봤으나 결과 좋지 않음</p> <p>정답을 찾아봤는데 틀린 문제들은 인과관계를 추론할 수 있는 능력이 있어야 했음. ex) eval_id 305, 102, 37 등 -> 이런 경우 dense방식으로 찾는 것이 가능하다면 좋겠지만 그 방법을 못찾음</p> <p>일단 답지부터 만들고 실험 시작하는 것이 좋을 듯</p> <p>@재명 - ColBERT 설치 문서 작성 완료.</p> <p>@지미 퀴리 시 명사구 문장과 키워드 합치도록 처리</p> <p>@지미 gemma2로 k 3~10개 리랭킹까지 완료 - 아직 섭미 못했지만 로컬 점수는 안 좋음..ㅠ</p> <p>@은지 - 재명님이 문서 공유해주셔서 드디어 colbert 설치->결과 60점정도</p> <p>- 메트릭 만들어서 서브밋 전에 점수확인해서 거를 수 있게 만들-테스트 제대로 못해봤습니다.</p> <p>- 책임랑 강의 다시 보면서 다시 공부</p>
오늘 할거	<p>@재명 - 지난 번에 만들어 놔던 질문들로 ColBERT 학습해보고 서브미션 해보기</p> <p>- Validation 데이터 만들기 (멘토의 제안)</p> <p>@지미 - gemma2로 테스트했던 것 유료 API로 전환해서 테스트</p> <p>@은지 (cross encoder-대회데이터로 모델 학습해서)rerank, 프롬프트 변경, 모델변경</p> <p>@윤재 cross-encoder로 reranking해보면?</p>
tip	
아이디어	<p>여러개 submission에서 topk가 많이 등장한 항목을 높은 순위로 다시 정렬해보기?</p> <p>colbert로 topk 넉넉하게 뽑은 후 crossencoer로 reranking하기</p>
기타	<p>다음 멘토링 목요일 점심 11시 45분 - 12시 45분</p>
10/15 어제까지 한 거	<p>@윤재 cross-encoder 결과 안좋은 매우 명칭함</p> <p>sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 384차원인데도 동일한 결과가 나옴, 저차원으로도 충분히 해결가능한 상태임, 임베딩모델이 문제가 아니라 학습의 문제인 듯</p> <p>@재명 - 실철014 ~ 017 을 실험 시트에 정리해 봤음.</p> <p>- Validation data 를 수동으로 7개 만들. 어떻게 만들어야 짧은 시간에 대량으로 만들 수 있을까??</p> <p>@지미 퀴리를 명사구와 키워드를 모두 사용했을때보다 명사구만 사용한 것이 레퍼 문서 3개일때는 점수가 더 좋았음(0.7629)</p> <p>@지미 llm 리랭킹 유효함</p> <p>@은지 재명님 colbert 점수보고 충격받아서 코드 비교해보고, 옵션 다 없애고 기본으로 하니 69~72정도 나옴(84점 submission 기준)</p> <p>colbert로 topk10개 뽑아서 sentence transformers crossencoder로 topk3개 뽑기->모델 파인튜닝하면서 비교하는 중(LLM은 안해봤어요)</p>

실험 기록

Google sheet를 이용해 실험 기록 공유

날짜	시간	환경	MAP	MRR	MODEL	내용	메모
10/8(화)		업스테이지	0.7091	0.7091	- 임베딩 : text-embedding-ada-002 - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Elasticsearch 8.8.0	- 임베딩 모델을 text-embedding-ada-002 로 -- Dense retrieval 만. -- 결론 : 베이스라인보다 점수가 조금 더 좋아짐.	실험 005
10/8(화)		업스테이지	0.6038	0.6061	- 임베딩 : text-embedding-3-small - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Elasticsearch 8.8.0	- 임베딩 모델을 text-embedding-3-small 로 -- Dense retrieval 만. -- 결론 : text-embedding-ada-002 보다 점수가 훨씬 안좋네!	실험 006
10/8(화)		업스테이지	0.7742	0.7758	- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Elasticsearch 8.8.0	- 임베딩 모델을 text-embedding-3-large 로 -- Dense retrieval 만. -- 결론 : Elasticsearch 8.8.0 은 2048 차원까지만 지원해서 차원을 줄였음, text-embedding-ada-002 보다 0.0651 좋음! 베이스라인 점수보다 0.07점 상승!	실험 007
10/8(화)	15:50:00	업스테이지	0.797	0.7985	- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Faiss	- Elasticsearch 는 임베딩 차원 2048 까지 지원함. 그 이상을 지원하는 Faiss 를 사용해서 text-embedding-3-large 의 원래 차원인 3072 를 그대로 사용해보자. -- Dense retrieval 만 함. -- 결론 : 2048 차원보다 0.02 점 상승!	실험 008
10/9(수)	9:47	업스테이지	0.7008	0.7015	- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Faiss	- 문서와 관련된 질문들을 각각 10개 생성하여, 이 질문들을 인덱싱 해놓고 이 질문을 검색해보자. - 결론 : 문서를 임베딩해서 그걸 검색하는것보다 점수가 0.09점 정도 하락!	실험 009
10/9(수)	10:21	업스테이지	0.5848	0.5894	- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-4o-2024-08-06 - 검색 엔진 : Faiss	- 실험 009 에서 llm 모델만 gpt-4o-2024-08-06 로 바꿔보자 - 결론 : 점수 왕창 떨어짐. 이유는 과학 상식이 아닌 질문을 보다 잘 인식해버려서 topk 를 비우고 자체적으로 답을 해버려서인듯!	실험 010
10/9(수)	14:38	업스테이지	0.7152	0.7167	- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Faiss	- llm 모델을 gpt-3.5-turbo-1106 로 바꾸고, 문서와 문서로부터 생성한 질문들, 각각에서 dense retrieval 한 것을 리랭킹해보자. - 결론 : score 가 문서에서 뽑은 것보다, 문서로부터 생성한 질문에서 뽑은 것들이 점수가 대체적으로 높아서 실험 009 와 비슷한 점수가 나옴.	실험 011
10/9(수)		업스테이지			- 임베딩 : OpenAI text-embedding-3-large - LLM : gpt-3.5-turbo-1106 - 검색 엔진 : Faiss	- 과학 상식 질문이 아닌 것들에 대해선 standalone_query 와 topk 를 수동으로 지워서 서버미션 해보자. 기존 파일은 실험 008 이고, 총 10 문제를 지웠음. eval_id : 213, 81, 279, 42, 21, 210, 233, 300, 212, 66 - 결론 : x / 220 = 0.797, y / 220 = 0.7061, 두 식을 계산하면, x = 175.34, y = 155.342, x - y = 20, 근데, 10문제를 지웠는데, 10문제가 다 틀렸다고해도 한문제당 1점인데 20점이 하락하는건 이상함..	

02

대회 소개 & 데이터 탐색

02 대회 소개 & 데이터 탐색

- 목표 : 질문과 이전 대화 히스토리를 기반으로 참고할 문서를 검색엔진에서 추출한 후, 이를 활용하여 질문에 적합한 답변을 생성하는 대회
 - 대회 기간 : 2024년 10월 2일 (10시) ~ 2024년 10월 24일 (19시)
 - 색인 대상 문서
 - 4272개의 교육 및 과학 상식 문서
 - Open Ko LLM Leaderboard에 들어가는 Ko-H4 데이터 중 MMLU, ARC 데이터 기반
 - 평가 데이터
 - 220개의 대화형 자연어 메시지
-

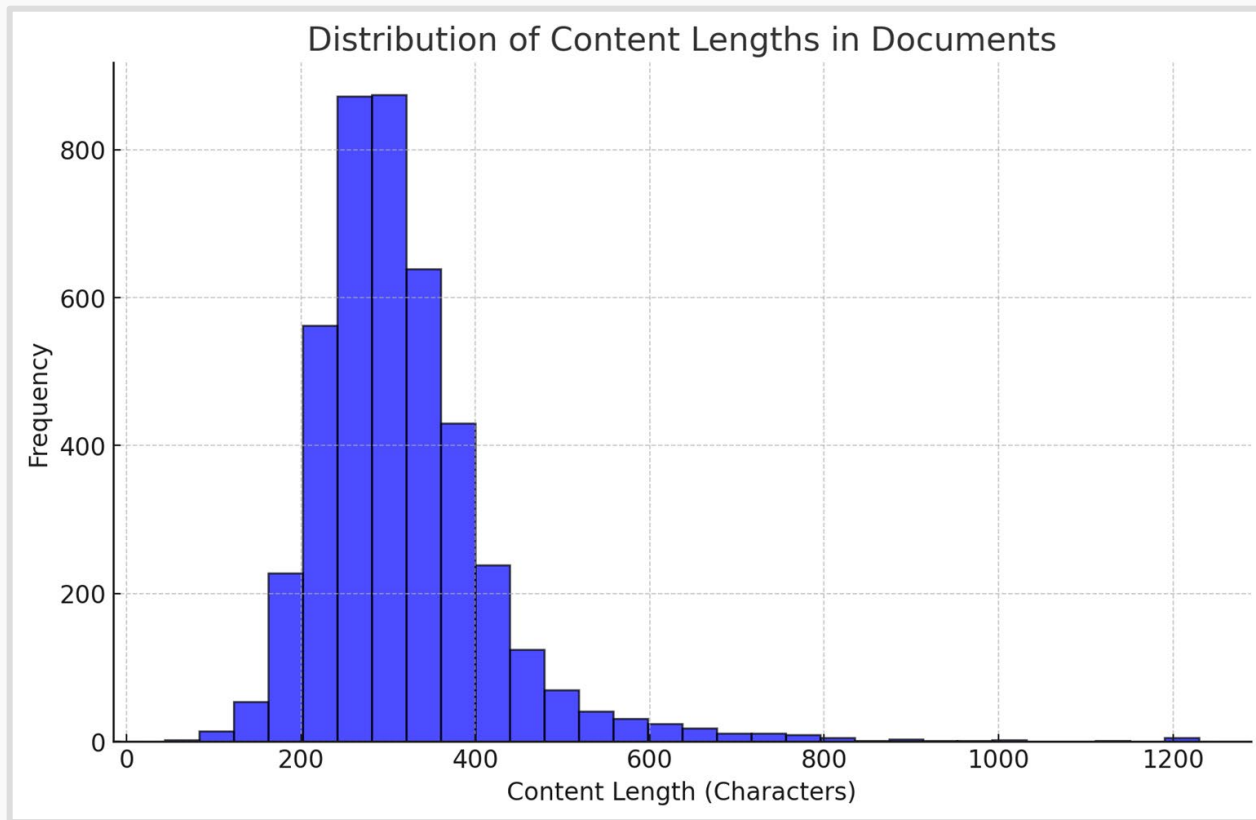
02 대회 소개 & 데이터 탐색

Evaluation Metric : MAP(Mean Average Precision)

```
def calc_map(gt, pred):
    sum_average_precision = 0
    for j in pred:
        if gt[j["eval_id"]]:
            hit_count = 0
            sum_precision = 0
            for i, docid in enumerate(j["topk"][:3]):
                if docid in gt[j["eval_id"]]:
                    hit_count += 1
                    sum_precision += hit_count / (i + 1)
            average_precision = sum_precision / hit_count if hit_count > 0 else 0
        else:
            average_precision = 0 if j["topk"] else 1
        sum_average_precision += average_precision

    return sum_average_precision / len(pred)
```

02 대회 소개 & 데이터 탐색

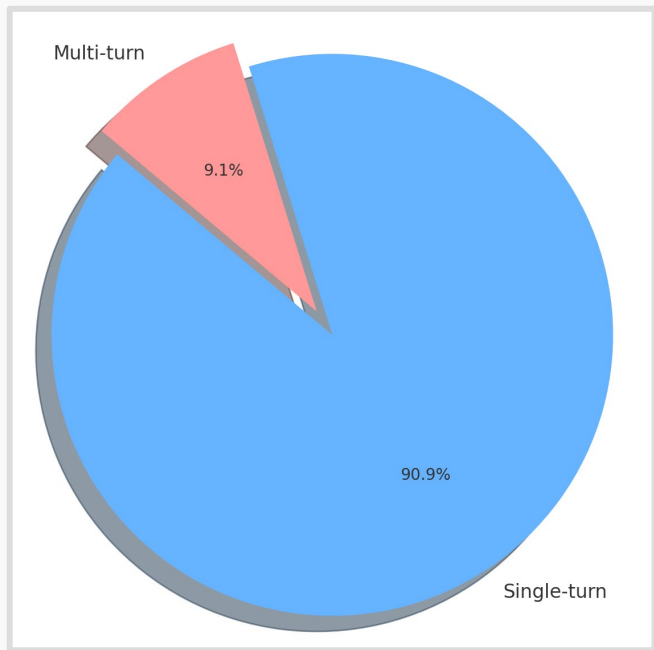


02 대회 소개 & 데이터 탐색

최소 글자 수	44자
최대글자수	1,230자
평균 글자 수	315자
중앙 값	299자
표준편차	104자
총 문서 수	4,272개

- 각 문서당 글자 수는 44자에서 1230자까지 문서마다 넓은 분포를 가지며 편차가 104로 상당히 큼을 알 수 있음
 - 평균 글자수와 표준편차를 고려하여 문서마다 chunk를 끊으면 문서에 대한 의미파악에 용이할 것으로 생각하였음
-

02 대회 소개 & 데이터 탐색 - EDA



총 평가 항목 수: 220개
단일 질문 데이터: 200개 (전체의 90.91%)
멀티턴(3 turn only) 대화 데이터: 20개 (전체의 9.09%)

Single Turn

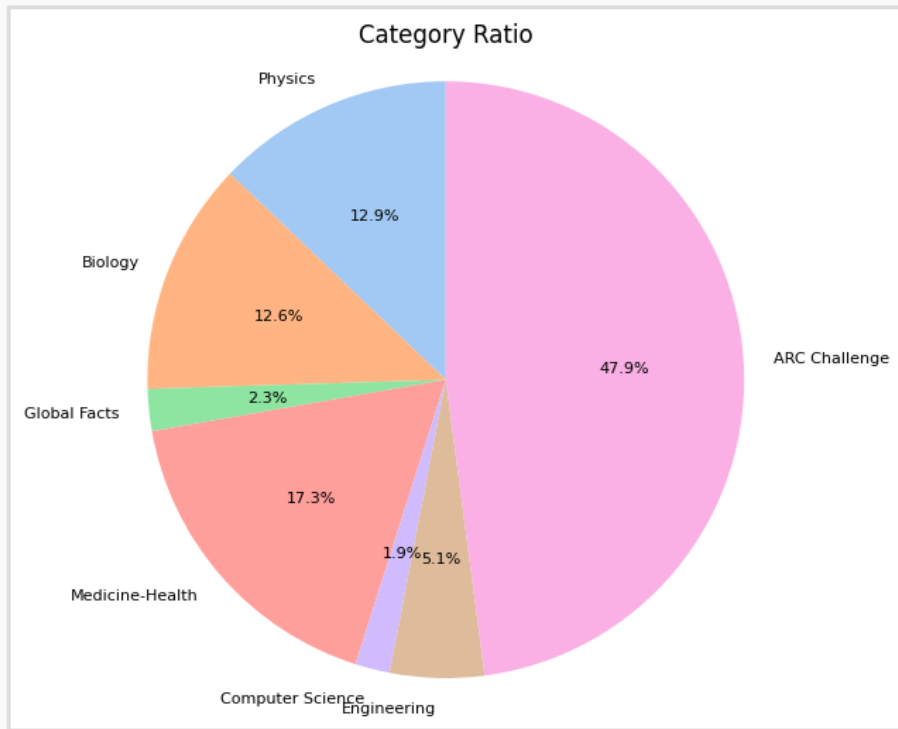
```
{"eval_id": 244,  
 "msg": [{"role": "user", "content": "정상파가 뭐야?"]}]}
```

Multi Turn

```
{"eval_id": 278,  
 "msg":  
 [{"role": "user", "content": "노인들이 종종 후각 능력을 잃게 되는 경  
우가 있더라구?}],  
 [{"role": "assistant", "content": "네 맞습니다."},  
 [{"role": "user", "content": "이렇게 냄새를 못맡게 되면 어떤 결과로  
이어지지?"]}]]}
```

- 문서 검색 과정에서 대화의 맥락 이해가 요구됨

02 대회 소개 & 데이터 탐색 - EDA



[ARC]

- 이번 대회에서는 비공개. 원본은 5개 카테고리로 구성
- 물리학, 화학, 생물학, 지구과학, 과학적 방법론

[MMLU]

대분류	소분류	src
공학	전기공학	electrical_engineering
공학	컴퓨터 보안	computer_security
컴퓨터 과학	고등학교 컴퓨터 과학	high_school_computer_science
컴퓨터 과학	대학 컴퓨터 과학	college_computer_science
물리학	고등학교 물리학	high_school_physics
물리학	개념 물리학	conceptual_physics
물리학	대학 물리학	college_physics
물리학	천문학	astronomy
생물학	고등학교 생물학	high_school_biology
생물학	대학 생물학	college_biology
생물학	바이러스학	virology
화학	고등학교 화학	high_school_chemistry
화학	대학 화학	college_chemistry
의료 건강	의학 유전학	medical_genetics
의료 건강	대학 의학	college_medicine
의료 건강	인간 성	human_sexuality
의료 건강	노화학	human_aging
의료 건강	영양학	nutrition
의료 건강	해부학	anatomy
일반 상식	일반 상식	global_facts

02 대회 소개 & 데이터 탐색 - Meta 정보 생성

- LLM(rt2r-ko-gemma-2-9b-it)으로 Meta 정보 생성

- 원본

"content": "트로이군 소행성은 목성의 궤도를 따라 위치해 있습니다. 이 소행성은 목성 앞뒤로 60°에 위치하고 있습니다. 목성은 태양계에서 가장 큰 행성으로 알려져 있으며, 그 궤도 주위에는 다양한 소행성들이 존재합니다. 트로이군 소행성은 목성과의 중력 상호작용으로 인해 안정한 궤도를 유지하고 있습니다. 이러한 특징 때문에 트로이군 소행성은 천문학적인 연구에 많은 관심을 받고 있습니다.",

"src": "ko_mmlu__astronomy__test"

- Meta 정보 : 제목, 키워드, 요약, 연관 카테고리

"title": "트로이군 소행성: 목성의 중력 궤도 안에 위치한 소행성"

"keywords": ["트로이군 소행성", "목성", "소행성", "중력", "궤도", "천문학"]

"summary": "트로이군 소행성은 목성의 궤도를 따라 60° 앞뒤로 위치하며, 목성과의 중력 상호작용으로 안정적인 궤도를 유지합니다. 목성은 태양계에서 가장 큰 행성으로, 다양한 소행성들이 그 주위를 도는 것으로 알려져 있습니다. 이러한 특징 때문에 트로이군 소행성은 천문학적 연구의 주요 대상입니다."

"categories": ["physics", "earth_science"]

02 대화 소개 & 데이터 탐색 - Valid Set 생성

- LLM(rt2r-ko-gemma-2-9b-it)으로 전체 문서의 대화 생성
- 해당 시점 리더보드 1위 모델 기준으로 규칙 적용
 - Valid Set 추론해서 topk 생성 후, 질문 생성 기준인 doc_id를 topk-1으로 re-ranking
 - Test Set과 중복 줄이기 위해 추론 결과의 topk-1 문서는 제외
- 최종적으로 3세트의 Valid Set 생성 - 230개, 460개, 1000개
 - 1 Turn 일상 대화 : 10%
`{"valid_id": 81, "docid": "", "msg": [{"role": "user", "content": "이 시간 즐거웠어. 다음에 또 이야기 나눌까?"}]}`
 - 1 Turn 질의 : 30%
`{"valid_id": 110, "msg": [{"role": "user", "content": "마이애미 파랑나비의 개체수 감소의 주요 원인은 무엇이며, 그 이유는 무엇인가요?"}], "docid": "910107a6-2a42-41a2-b337-fbf22d6440fe"}`
 - 1 Turn 유의어 질의 : 50%
`{"valid_id": 69, "msg": [{"role": "user", "content": "문자열 비교에서 대문자와 소문자의 우선순위가 어떻게 작동하는지 설명해주세요."}], "docid": "01b3d119-c0e1-4dcc-b5f4-9f57cd681f43"}`
 - 3 Turn 질의 : 10%
`{"valid_id": 0, "msg": [{"role": "user", "content": "화학 변화가 일어나는 예시를 좀 더 자세히 설명해줄 수 있니?"}, {"role": "assistant", "content": "물론입니다. 예를 들어, 빵을 구울 때 글루텐이라는 단백질이 열에 의해 변형되어 빵의 쫄깃한 식감을 만들고, 냄새와 맛을 변화시키는 화학 반응이 일어납니다."}, {"role": "user", "content": "글루텐이 어떤 변화를 겪는지 더 자세히 설명해줄 수 있니?"}], "docid": "ef6ad09c-9502-452f-85a9-796b978ca3b8"}`

03

IR 파이프라인 구축

03 IR 파이프라인 구축: Chunking

Naive Chunking

글자수 기준 Chunking

- Fixed size chunking
- Recursive Chunking
- Document Specific Chunking

Semantic Chunking

의미적 유사성 기반의 Chunking

- Semantic Chunking
- Agentic Chunking

03 IR 파이프라인 구축: Naive Chunking

Fixed Size Chunking

- 고정된 크기(200자)로 텍스트를 나누는 방법
- 문장의 논리적 구조와 관계없이 일정한 글자 수 단위로 텍스트를 잘라냅니다.

입력 텍스트 ----> [청크 1] ----> [청크 2] ----> [청크 3] ----> ... ----> [청크 N]
고정 크기 200자씩 나누어지는 구조

Recursive Chunking

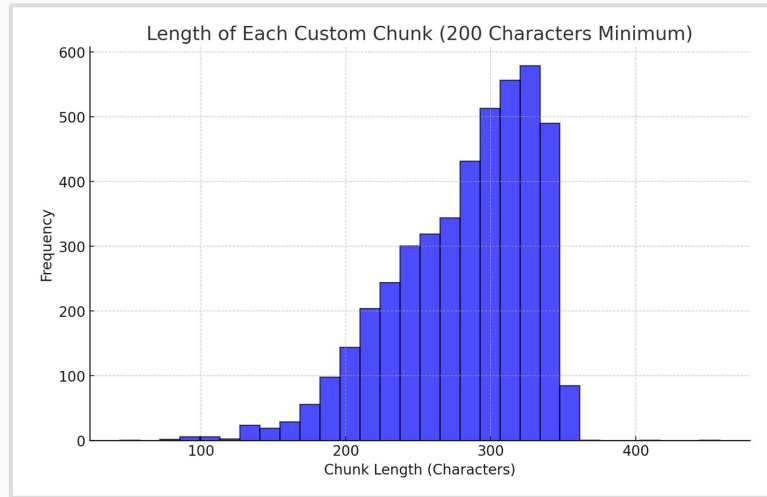
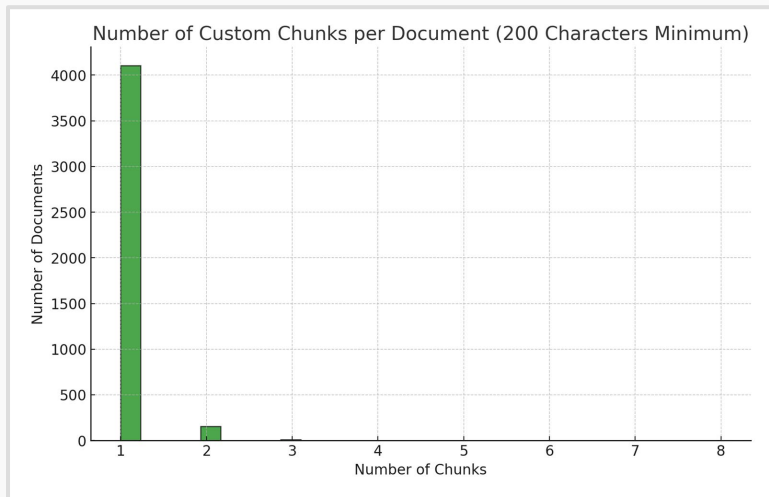
- 문장을 기준으로 텍스트를 나누되, 각 청크의 크기가 200자를 넘지 않도록 재귀적으로 나누는 방법
- 문장 단위의 논리적 흐름을 유지하면서 청크를 나눕니다.

입력 텍스트 ----> [문장 1 + 문장 2] ----> [문장 3] ----> [문장 4 + 문장 5] ----> ...
각 청크는 문장 단위로 나뉘며, 청크 크기가 200자를 넘지 않도록 제어

03 IR 파이프라인 구축: Naive Chunking

Fixed Size Chunking	Recursive Chunking
<p>동물 해부 시에는 다양한 장비와 도구가 사용됩니다. 그러나 안전 샤워기는 동물 해부 시에 필요하지 않은 장비입니다. 동물 해부는 주로 해부학 연구나 교육 목적으로 수행되며, 동물의 내부 구조를 탐구하는 과정입니다. 이를 위해 해부 칼, 해부용 장갑, 해부용 고글 등의 도구가 사용됩니다. 이러한 도구들은 해부 과정에서 안전하고 정확한 작업을 할 수 있도록</p>	<p>동물 해부 시에는 다양한 장비와 도구가 사용됩니다. 그러나 안전 샤워기는 동물 해부 시에 필요하지 않은 장비입니다. 동물 해부는 주로 해부학 연구나 교육 목적으로 수행되며, 동물의 내부 구조를 탐구하는 과정입니다. 이를 위해 해부 칼, 해부용 장갑, 해부용 고글 등의 도구가 사용됩니다.</p>
<p>도움을 줍니다. 그러나 안전 샤워기는 동물 해부에는 필요하지 않은 장비입니다. 안전 샤워기는 주로 동물들을 샤워하거나 청소하는데 사용되는 장비로, 동물 해부와는 관련이 없습니다. 따라서 동물 해부 시에는 안전 샤워기를 사용할 필요가 없습니다.</p>	<p>이러한 도구들은 해부 과정에서 안전하고 정확한 작업을 할 수 있도록 도움을 줍니다. 그러나 안전 샤워기는 동물 해부에는 필요하지 않은 장비입니다. 안전 샤워기는 주로 동물들을 샤워하거나 청소하는데 사용되는 장비로, 동물 해부와는 관련이 없습니다. 따라서 동물 해부 시에는 안전 샤워기를 사용할 필요가 없습니다.</p>
<p>글자수 단위로 끊음에 따라 문장이 부자연스럽게 끊기면서 문장구조가 파괴되는 문제가 발생</p>	<p>문장을 보존하며 Chunk를 나누지만 각 chunk별로 의미적으로 뚜렷하게 나누어지지 않음</p>

03 IR 파이프라인 구축: Naive Chunking



문서별 청크 개수:

- 총 문서 수: 4,271개
- 평균 청크 수: 1.04개
- 최소 청크 수: 1개
- 최대 청크 수: 8개
- 문서의 50%: 청크 1개

200~350글자 기준 청크의 글자 수:

- 총 청크 수: 4,459개 평균 글자 수: 282자
- 최소 글자 수: 44자 최대 글자 수: 458자
- 청크의 50%: 292자 청크의 75%: 320자

03 IR 파이프라인 구축: Semantic Chunking

Semantic Chunking

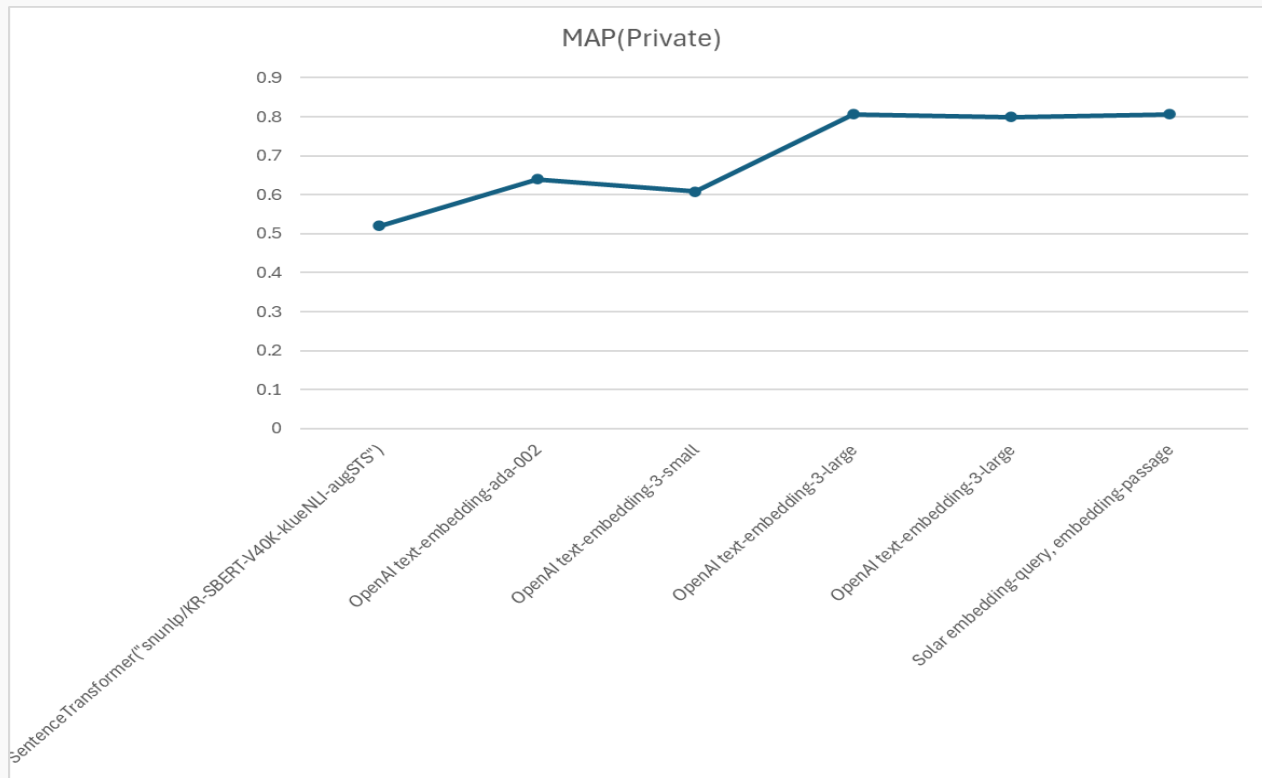
- 텍스트를 의미적으로 유사한 단위로 나누어 청크로 분리
 - 각 청크는 문맥적 유사성을 기반으로 형성되며, 단순한 글자수나 문장 경계를 따르는 것이 아니라 문장 간의 의미적 관계에 따라 분리
1. 문서 로드
 - → 문서 텍스트 추출
 - → 문장 분할 (KSS 사용)
 2. 임베딩 및 유사도 계산
 - → Sentence Transformer로 문장 임베딩 (snunlp/KR-SBERT-Medium-extended-patent2024-hn)
 - → 문장 간 유사도 계산 (코사인 유사도)
 3. 청크 생성
 - → 유사도 임계값에 따라 청크 분리 (Threshold = 0.7)
 - → 청크 길이 조정 (최소 150자)
 4. K-Means 클러스터링 (최대 청크 수 조정)
 - → 최대 청크 수 초과 시, K-Means를 사용해 청크 그룹화
 5. 결과 저장
 - → 각 청크 결과를 JSON 파일로 저장

03 IR 파이프라인 구축: Embedding

- Baseline 에서 아래 임베딩을 적용하여 Dense retrieval 만 수행시 결과.
- OpenAI text-embedding-3-large 와 Solar embedding 이 성능이 제일 좋음.

임베딩 모델	MAP(Private)	설명
SentenceTransformer("snunlp/KR-SBERT-V40K-klueNLI-augSTS")	0.5205	
OpenAI text-embedding-ada-002	0.6394	
OpenAI text-embedding-3-small	0.6076	
OpenAI text-embedding-3-large	0.8061	<ul style="list-style-type: none">• 임베딩 차원을 3072 에서 2048 로 축소• Elasticsearch 8.8.0
OpenAI text-embedding-3-large	0.8000	<ul style="list-style-type: none">• 임베딩 차원 3072• Faiss• 2048 로 차원 축소한것보다 점수 낮음!
Solar embedding-query, embedding-passage	0.8061	<ul style="list-style-type: none">• 임베딩 차원 4096• Elasticsearch 8.15.2• OpenAI text-embedding-3-large 와 동일!

03 IR 파이프라인 구축: Embedding



03 IR 파이프라인 구축: Embedding

- 사용한 모델 성능 순 나열
`SentenceTransformer("snunlp/KLUE-SRoBERTa-Large-SNUExtended-klueNLI-klueSTS")`
`SentenceTransformer("jhgan/ko-sroberta-multitask")`
`SentenceTransformer("snunlp/KR-SBERT-V40K-klueNLI-augSTS")`
`SentenceTransformer("intfloat/multilingual-e5-large")`
⇒ 한국어 훈련 모델이 다국어 모델보다 MAP기준 0.1점 정도 상승
BERT보다 SRoBERTa 성능이 더 좋음
- `Embeddings_text-embedding-3-large_dim2048`
`SentenceTransformer("sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2")`
⇒ `embeddings.shape[1]`이 2048, 384인 경우 MAP 결과 차이가 없음
과학지식문서를 뽑는 현 대회 상황에서는 `sparse_retrieve`방식이 `dense`보다 적합한 문서를 더 잘 뽑기 때문
- 언제 큰 embedding 모델을 쓰는게 좋을까?
⇒ 다양한 주제와 문서에서 범용적인 문장 표현을 생성해야 하는 경우
풍부한 임베딩이 필요한 복잡한 의미 분석 작업을 수행할 때
대규모 자원과 시간이 허용되는 환경에서 더 높은 성능을 기대할 때

03 IR 파이프라인 구축: Embedding

- 임베딩 모델을 적용하여 Dense Retrieve만 수행했을 때

임베딩 모델	언어	Fine-tuning	MAP / MRR (submission점수 0.84점 기준)
colbert-ir/colbertv2.0	영어	x	0.2091 / 0.2091
hunkim/sentence-transformer-klue	한국어	x	0.6893 / 0.6992
hunkim/sentence-transformer-klue	한국어	(문서를 기준으로 질의 3개 생성) 1 epoch	0.7113 / 0.7219
klue/roberta-large	한국어	x	0 / 0
klue/roberta-large	한국어	(문서를 기준으로 질의 3개 생성) 1 epoch	0.675 / 0.6856

- > 한국어로 학습된 모델이 영어 기반 모델 보다 더 높은 성능을 보임
- > 모델을 그대로 사용하는 것보다 학습하여 적용한 경우 성능이 더욱 향상됨

- 학습은 documents.jsonl 데이터의 content 기준으로 LLM모델을 이용해 문서 당 질의 3개씩 추출하여 강의에 나왔던 쿼리, 긍정 예시, 부정 예시로 구성된 트리플렛을 만들어서 학습함.
- 초반 점수가 낮아 submission 기회를 save하기 위해 팀 내 최고점 데이터를 이용하여 점수 확인하여 실제 submission점수와는 차이가 있을 수 있음

03 IR 파이프라인 구축: Standalone Query 프롬프트

• LLM 모델 별 프롬프트

Anthropic : claude-3-5-sonnet-20240620	Gemma2 : rtzr-ko-gemma-2-9b-it	OpenAI : gpt-4o	OpenAI : o1-mini(Beta)
<p>persona_answer = "당신은 상식 전문 AI입니다. 사용자의 메시지와 Reference 문서를 활용하여 짧게 답변합니다."</p> <p>persona_function_calling = "당신은 상식 전문 AI입니다. 사용자 메시지가 일상대화가 아니라면, tools를 호출하세요."</p> <pre>tools=[{ "name": "search", "description": "Search for documents related to common sense", "input_schema": { "type": "object", "properties": { "standalone_query": { "type": "string", "description": "The final query suitable for searching common sense in user message history." } } }, "required": ["standalone_query"], }]</pre>	<p>당신은 과학 기술에 특화된 상식 RAG 시스템입니다. 사용자 대화를 읽고, 답변 참조 문서 검색용(명사구 질문, 키워드)를 생성해주세요.</p> <p>응답의 상세한 조건은 다음을 따르세요:</p> <ol style="list-style-type: none"> **입력 정보**: 사용자 대화 이력 **출력 형식**: JSON 포맷으로 다음 정보를 포함해 주세요: <ul style="list-style-type: none"> - "query": 문맥이 정확히 드러나는 명사구 질문. 사용자 대화 이력이 일상 대화라면 필드 삭제. - "keywords": 검색 의도가 명확하게 드러나는 핵심 키워드 목록. 사용자 대화 이력이 일상 대화라면 필드 삭제. - "answer": 사용자 대화 이력이 일상 대화라면 스레드 종료 응답, 아니라면 필드 삭제. **예시**: <ul style="list-style-type: none"> - 과학이나 기술 상식 답변 예시: { "query": "카를로 로벨리의 루프 양자중력 이론과 최근 연구 동향", "keywords": ["카를로 로벨리", "Carlo Rovelli", "물리학자", "루프 양자중력 이론", "loop quantum gravity", "LQG"], } - 일상대화 답변 예시: { "answer": "죄송합니다. 저는 과학 지식이나 일반 상식에 대한 답변만 가능합니다." } **주의사항** <ul style="list-style-type: none"> - 응답은 반드시 예시처럼 JSON Body 만 출력하세요. - 당신이 바로 답변할 수 있는 내용도 답변이 아니라, 쿼리와 키워드를 생성해야 합니다. - 당신에 관한 대화는 쿼리 생성할 필요가 없는 일상 대화입니다. 	<p>대화의 질문을 파악해서, 관련 상식을 검색하는 데 적합한 최종 쿼리를 생성하세요.</p> <p>**output**: 다음 필드를 사용하는 json body only</p> <p>-query: 지식 검색이라면 쿼리, 잡담이라면 공백</p> <p>-answer: 지식 검색이라면 공백, 잡담이라면 짧은 메시지</p> <p>**dialogue**:</p>	<p>사용자와 대화한 내용을 바탕으로 질문에 대한 답변이 될 문서를 검색하기 위한 구체적인 질문 query를 만드세요.</p> <p>최대한 사용자가 작성한 단어를 그대로 사용함으로써 핵심 내용을 잘 간직하며 다음과 같은 원칙을 지켜주세요.</p> <ol style="list-style-type: none"> 1. 명확한 정보 요청 추출 2. 대화 흐름에 따라 추가 정보를 반영 3. 감정적 표현은 배제하고 정보 요청에 집중 4. 정보가 부족한 대화는 제외하거나 재구성 5. 대명사나 생략된 정보를 보완 6. 불필요한 맥락 제거 7. 중복 및 반복 제거 8. 질문에 대한 대답으로 예상되는 핵심 키워드가 있다면 3개까지 추가 <p>사용자와의 대화는 다음과 같습니다:</p>

03 IR 파이프라인 구축: Standalone Query 결과

LLM 모델	Standalone Query	MAP MAP (Final)	MRR MRR (Final)
Anthropic : claude-3-5-sonnet	바이러스와 원핵 세포 상호작용 메커니즘	0.8689 0.8045	0.8712 0.8045
Gemma2 : rtzr-ko-gemma-2-9b-it	바이러스가 원핵 세포에 감염할 때 사용하는 기전	0.7674 0.8053	0.7682 0.8045
OpenAI : gpt-4o	바이러스가 원핵 세포와 상호 작용하는 방법	0.7848 0.8182	0.7848 0.8182
OpenAI : o1-mini(Beta)	바이러스가 원핵 세포와 상호작용할 때 사용하는 것	0.8015 0.7591	0.8091 0.7591

03 IR 파이프라인 구축: Elasticsearch Indexing

특정 부분에 국한한 내용보다

전반적인 경우에 가중치를 두어 점수 향상

```
settings = {  
  "index": {  
    "similarity": {  
      "lm_jelinek_mercer": {  
        "type": "LMJelinekMercer",  
        "lambda": 0.7  
      }  
    }  
  },  
}
```

similarity: LMJelinekMercer

- 랭킹 및 검색의 점수 계산 방식에서 Jelinek-Mercer 확률적 언어 모델(LM)을 사용하여 문서와 쿼리 간의 유사도를 측정하는 방식
- **lambda**는 Jelinek-Mercer 스무딩에서 문서 내 정보와 코퍼스 전체 정보의 가중치를 조절하는 혼합 계수
- **lambda** 값이 0.7이라는 것은, 문서 자체의 단어 빈도를 30% 반영하고, 코퍼스 전체에서의 단어 빈도를 70% 반영한다는 의미.

디엔에이와 단백질의 관계와 역할

lambda=0.3

징크 핑거 단백질과 헬릭스-턴-헬릭스 단백질은 디엔에이 결합 단백질의 유형입니다. 디엔에이 결합 단백질은 DNA와 상호작용하여 유전자 발현을 조절하는 역할을 합니다. 징크 핑거 단백질은 징크 이온을 이용하여 DNA와 결합하여 특정 유전자의 발현을 조절합니다. 헬릭스-턴-헬릭스 단백질은 특정 DNA 서열과 결합하여 DNA를 휘감는 구조를 형성하며, 이를 통해 유전자 발현을 조절합니다. 디엔에이 결합 단백질은 생물체의 발달, 세포 분열, 면역 반응 등 다양한 생물학적 과정에서 중요한 역할을 수행합니다.

lambda=0.7

DNA와 단백질은 생물체 내에서 중요한 역할을 담당하는 분자들입니다. DNA는 유전 정보를 저장하고 전달하는 역할을 담당하며, 단백질은 다양한 생화학적 기능을 수행합니다. 이 두 분자는 서로 긴밀하게 상호작용하여 생물체의 생존과 발달에 기여합니다... 뉴클레오타이드는 DNA와 단백질이 상호작용하여 생물체의 유전 정보를 제어하고 조절하는 역할을 수행합니다. 이러한 뉴클레오타이드의 구조와 기능은 생물체의 특성과 발달에 중요한 영향을 미칩니다. 따라서, 뉴클레오타이드는 생물체의 생존과 진화에 핵심적인 역할을 담당하고 있습니다."

Topk
5위
→1
위로
변동

03 IR 파이프라인 구축: Elasticsearch Indexing

동의어, 신조어 추가를 통한 인덱싱 검색 성능 확장

▼ elasticsearch-8.8.0

> bin

▼ config

> certs

> jvm.options.d

▼ synonyms

≡ synonyms1.txt

≡ synonyms2.txt

≡ synonyms3.txt

≡ synonyms4.txt

≡ synonyms5.txt

≡ synonyms6.txt

≡ synonyms7.txt

≡ user_dictionary.txt

user_dictionary.txt

```
1  오토마톤, NNG
2  NMR, NNP
3  Elodea, NNP
4  AIDS, NNP
5  마이애미파랑나비, NNP
6  헤르페스, NNG
7  광자, NNG
8  아메리카알리게이터, NNP
```

synonyms.txt

예외 처리, 예외처리

Dmitri Ivanovsky, 드미트리 이바노프스키

traffic, 트래픽

sniffing, 스니핑

농장 유출 => 농장

비료 => 농장

dense similarity를 L2_norm 으로 선택

```
mappings = {
  "properties": {
    "content": {
      "type": "text",
      "analyzer": "nori_index_1",
      "search_analyzer": "nori_1"
    }
  }
}
```

점 분석기

```
},
"embeddings": {
  "type": "dense_vector",
  "#dims": 384,
  "dims": len(embeddings),
  "index": True,
  "similarity": "l2_norm"
}
}
```

cosine: 벡터를 정규화한 상태에서 유사성 측정, 벡터의 길이가 결과에 영향을 주지 않도록 함. 벡터의 방향성에 초점.

l2_norm: 유클리드방식, 벡터의 크기와 방향 모두를 고려하여 계산. 데이터가 정규화되어 있지 않을 경우 절대적 거리를 계산하여 더 세밀한 차이 포착.

03 IR 파이프라인 구축: Retrieval

제목과 본문을 결합하여 문서 벡터 생성 (텍스트 임베딩)

```
vector = encoder.encode(  
    doc['title'] + " " + doc['content'])
```

문서의 메타 정보와 벡터 데이터를 Elasticsearch에 인덱싱하기 위한 payload 생성

```
index_document = {  
    "_index": "science_documents",  
    "_id": doc_id,  
    "_source": {  
        "docid": doc_id,  
        "categories": doc['categories'],  
        "title": doc['title'],  
        "summary": doc['summary'],  
        "keywords": doc.get('keywords', []),  
        "content": doc['content'],  
        "vector": vector.tolist()}}
```

content에 title, keywords, summary 추가 :
Query와의 간극을 줄이려는 시도

```
body = {# MAP 0.8045 MRR 0.8061  
    "query": {  
        "bool": {  
            "should": [  
                {  
                    "multi_match": {  
                        "query": query_str,  
                        "fields": ["content^3"],  
                        "boost": 0.0025,  
                        "fuzziness": "AUTO"}  
                    }  
                ],  
            "minimum_should_match": 1}  
        }  
    },  
    "knn": {  
        "field": "embeddings",  
        "query_vector": query_embedding.tolist(),  
        "k": 7,  
        "num_candidates": 70,  
        "boost": 3.5  
    },  
    "size": size  
}
```

Hybrid retrieval : 최적의 조합 pool 생성 가능

03 IR 파이프라인 구축: Retrieval

ColBERT v2



- ColBERT 추가시 성능이 좋아짐
 - (대회 초반에는 **0.037** 정도 상승 했지만, 후반에 여러 기법으로 성능을 올린 후에는 약 **0.009** 정도로 큰 성능 향상은 없기는 함)
- ColBERT 학습 데이터 만들때, 관련 없는 문서는 랜덤으로 선정하는 것이 제일 성능이 좋았음
- ColBERT 학습 시, nbits 를 2 에서 8 로 수정하면 성능이 좋아짐
 - Public 에서는 성능 하락(**-0.0212**), Private 에서 성능 상승(**+0.0083**)
 - Public 점수가 하락하여 nbits 를 2 로 유지
- ColBERT 학습시, 문서당 질문 10개와 문서당 질문 20개는 성능에 큰 차이가 없음.

03 IR 파이프라인 구축: Reranking

상호 순위 조합 (Reciprocal Rank Fusion, RRF)

- 대회 초반에 사용 (Hybrid Retrieval 적용 이전)
- 연산이 가벼움
- 크로스 인코더 baseline보다 높은 성능

```
from collections import defaultdict
from typing import List

def reciprocal_rank_fusion(common_results_list: List[List[dict]], k=5):
    rrf = {}

    for common_results in common_results_list:
        for i, common_result in enumerate(common_results, 1):
            key = common_result['docid']

            if key not in rrf:
                dic = defaultdict(float)
                rrf[key] = dic
            else:
                dic = rrf[key]

            dic['score'] += 1.0 / (k + i)
            dic['content'] = common_result['content']

    return sorted(rrf.items(), key=lambda x: x[1]['score'], reverse=True)
```

03 IR 파이프라인 구축: Reranking

Colbert로 10개 문서 추출 후 SentenceTransformer의 Cross-Encoder를 활용하여 결과를 리랭크

Model	Fine-tuning	MAP / MRR (submission점수 0.84점 기준)
Colbert 단일모델 사용 시(기준 점수)		0.7113 / 0.7219
klue/roberta-small	X	0.3166 / 0.3151
klue/roberta-small	(문서를 기준으로 질의 3개 생성하여 학습) 1 epoch	0.2560 / 0.2583
bongsoo/klue-cross-encoder-v1	X	0.7215 / 0.7340
bongsoo/klue-cross-encoder-v1	(문서를 기준으로 질의 3개 생성하여 학습) 1 epoch	0.7215 / 0.7340
bongsoo/klue-cross-encoder-v1	(문서를 기준으로 질의 3개 생성하여 학습) 2 epoch	0.7405 / 0.7492 (submission mid 기준 0.7871 / 0.7879)
klue/roberta-large	X	0.3265 / 0.3280
klue/roberta-large	(문서를 기준으로 질의 3개 생성하여 학습) 1 epoch	0.7056 / 0.7196
klue/roberta-large	(문서를 기준으로 질의 3개 생성하여 학습) 2 epoch	0.7037 / 0.7181

bongsoo/klue-cross-encoder-v1 모델

→ KLUE/bert-base 모델을 훈련시켜 Cross Encoder로 파인튜닝한 모델이기 때문에 성능이 가장 우수했음

- 학습은 documents.jsonl 데이터의 content 기준으로 LLM모델을 이용해 문서 당 질의 3개씩 추출하여 문서-질의 쌍을 만들어 문서에 대한 긍정질문일 경우 label=1, 부정질문일 경우 label=0부여함
- 초반 점수가 낮아 submission 기회를 save하기 위해 팀 내 최고점 데이터를 이용하여 점수 확인하여 실제 submission점수와는 차이가 있을 수 있음

03 IR 파이프라인 구축: Reranking, LLM query 작성시 주의할 점

standalone_query <<

새로운 항생제의 효과를 입증하기 위한 과학적 방법

combined_dialogue + standalone_query <<

새로운 항생제의 효과를 입증하기 위한 과학적 방법 새로 만든 항생제가 드디어 나왔어. 네 무엇을 도와 드릴까요? 그 효과를 확실히 얘기하기 위해서 해야 할 일은?

<< combined_dialogue

새로 만든 항생제가 드디어 나왔어. 네 무엇을 도와 드릴까요? 그 효과를 확실히 얘기하기 위해서 해야 할 일은?

standalone_query 일반적이고 학술적인 설명을 요구하는 지문이 더 적합할 수 있음. 이 경우에는 항생제의 효과를 입증하는 방법론에 대한 명확하고 체계적인 설명이 포함된 지문이 가장 적합.

combined_dialogue 실제 상황에서 새로운 항생제의 효과를 입증하기 위해 어떤 구체적인 단계를 밟아야 하는지 설명하는 지문이 더 적합할 수 있음.

⇒ 짧아진 summary, title, standalone_query는 대화의 미묘한 문맥의 차이를 발생시킴

⇒ combined_dialogue 인 경우가 점수가 가장 높음

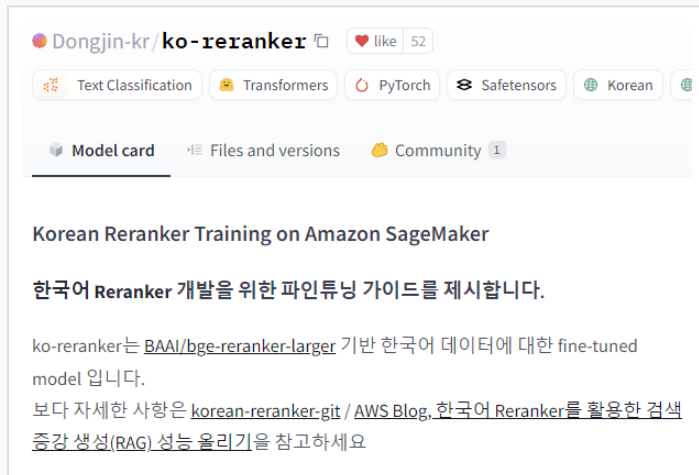
03 IR 파이프라인 구축: Reranking

1차로, Reranker 모델로 문서 순위 정렬

- Dongjin-kr/ko-reranker 모델 사용
- 질문과 문서를 입력으로 사용
- 성능 매우 좋음 (약 **+0.0425**)

2차로, LLM 으로 최대 3개의 문서 ID 선별

- 최대 상위 20개 문서 전달
- 성능이 더 좋아짐 (약 **+0.0273**)
- GPT-4o 사용



```
topk_content = ""
for i, rst in enumerate(common_results):
    topk_content += f"관련문서_{i}_ID: {rst['docid']}\n"
    topk_content += f"관련문서_{i}_내용: {rst['content']}\n" # content 는 체크가 아닌 원래 문서의 content!

system_message = {
    "role": "system",
    "content": """
당신은 문서 내용의 적합성을 평가하는 전문가입니다.
질문과 가장 관련도가 높은 3개 문서만 주려서, 해당 문서의 ID를 알려주세요.
가장 관련도가 높을수록 ID를 먼저 적어주고, 다음과 같은 형태로 알려주세요.
b0864744-6dfd-4240-8a93-0320aac2429f, b0864744-6dfd-4240-8a93-0320aac2428f, b0864744-6dfd-4240-8a93-0320aac2427f
관련된 문서들이 없는 경우 아무것도 생성하지 마세요.
"""
}

query_message = {"role": "user", "content": f"질문: {standalone_query}"}
document_message = {"role": "user", "content": f"문서 {max_len}개의 ID와 내용: {topk_content}"}

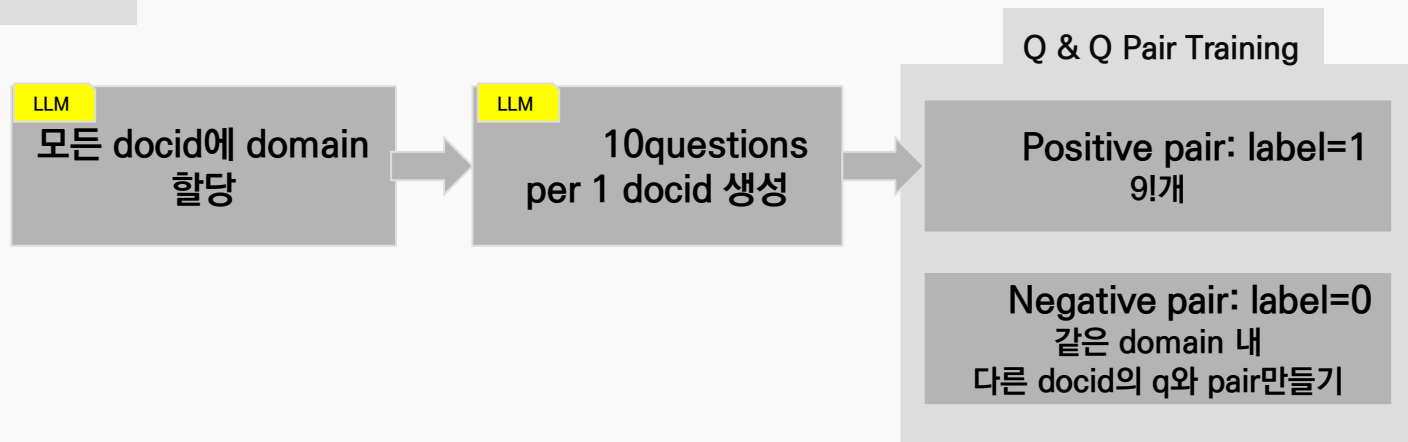
full_message = [system_message, query_message, document_message]
```

03 IR 파이프라인 구축: Retrieval

Chunk

- 문서를 여러 개의 Chunk로 나눠 사용하는 것이 성능이 더 좋음
- 문서의 글자수 분포를 고려하여 300자 기준으로 나눔
- 리랭커 모델로 리랭킹시, 문서 보다는 Chunk 를 사용하는 것이 성능이 더 좋음

03 IR 파이프라인 구축: CrossEncoder: reranker 모델 훈련



*기하급수적으로 늘어나는 pair수
⇒reranker로 topk5에 해당하는 docid에 대해서만 진행

*Hard Negative pair:
similarity가 높은 타 문서의 question으로 positive와 구별이 어려운 고난이도의 pair를 만들어 정교한 학습을 유도할 수 있음

Lesson Learned

*Q & Q 방식은 Q & C 방식에 비해 질문 자체가 더 짧고 간결하기 때문에, 유사도를 판단할 때 의미적으로 완전한 맥락이 부족할 수 있음

*잘못된 pos-neg pair 학습→ Map점수 하락

*의미없는 only positive pair 학습→Map점수변동없음

⇒ 정교한 훈련데이터셋 생성의 어려움

03 IR 파이프라인 구축: LLM prompt engineering

당신은 과학 기술에 특화된 상식 RAG 시스템입니다.

사용자 대화 이력에 대해 답변을 작성하기 위해 참고할 content의 id를 관련도 높은 순으로 3개 반환해주세요.

응답의 상세한 조건은 다음을 따르세요:

1. 입력 정보: dialogue와 참고 문서

- dialogue: 사용자 대화 이력

- references : 답변 작성에 사용할 문서(id, content)

- 입력 예시: {

"dialogue":[{"role": "user", "content": "몽계구름이 발생했을 때 비가 얼마나 와?"},

{"role": "assistant", "content": "몽계구름 발생 시 비의 양은 기압, 습도, 온도 등 여러 요소에 따라 달라집니다."},

{"role": "user", "content": "어떤 데이터를 분석하면 정확히 예측할 수 있어?"}],

"references":{

{

"id":0,

"content":"건강한 아침 식사를 하는 사람은..."

},

{

"id":1,

"content":"적운은 하층 대기에서 주로 맑은 날씨에 나타나는..."

},

{

"id":2,

"content":"지진은 지각판의 움직임으로 인해 발생하는..."

},

{

"id":3,

"content":"적운은 보통 비와는 관련이 없지만, 특정 조건에서는..."

}

}}

2. 출력 예시: [3,1]

3. **주의사항**

- 응답은 반드시 예시처럼 문서 id 배열만 출력하세요.

● 전체 포맷

- 요구사항과 출력 형식 명확하게 전달
- OPENAI의 Structured output 기능을 PROMPT ENGINEERING을 통해 구현
- 로컬 LLM 에서도 최대한 OUTPUT의 일관성 보존

● Dialogue

- Function Call 기능이 없는 LLM 에서도 맥락 전달

03 IR 파이프라인 구축: LLM의 한계: 매번 달라지는 아웃풋

“해구가 생기는 이유”에 알맞는 지문을 선택해주세요

1번. 섭입대는 대륙 지각과 해양 지각이 충돌하는 지점을 나타내는 지질 현상입니다. 이러한 충돌 지점에서는 다양한 지질 특성이 관찰될 수 있습니다. 그 중 하나는 해구입니다. 해구는 해양 지각이 대륙 지각을 침식하여 형성된 해안 지형입니다. 해구는 흐르는 물의 힘에 의해 형성되며, 대개 반달 모양을 가지고 있습니다. 이러한 해구는 섭입대의 끝 부분에 위치할 수 있으며, 섭입대의 지질 특성 중 하나로 포함될 수 있습니다. 섭입대의 다른 지질 특성으로는 퇴적층, 암석의 굽은 형태, 균열 등이 있을 수 있습니다. 이러한 지질 특성들은 학생이 대륙 지각과 해양 지각이 충돌하는 섭입대를 보여주는 도표를 만들 때 고려해야 할 중요한 요소입니다.

2번. 지구의 지각은 주로 현무암과 화강암으로 구성되어 있습니다. 현무암은 주로 해양 지각을 형성하고, 화강암은 대륙 지각을 형성합니다. 판이 충돌할 때, 해양 지각은 대체로 대륙 지각 아래로 잠기는 현상이 발생합니다. 이는 해양 지각이 대륙 지각보다 밀도가 높기 때문입니다. 해양 지각은 더 무거우므로 대륙 지각 아래로 잠기게 됩니다. 이러한 현상을 통해 지구의 지각은 지속적으로 변화하고 있습니다.

gpt-4o-mini-2024-07-18 2번 선택

gpt4o 1번 선택

gpt o1-preview 2번 선택

주어진 질문 "해구가 생기는 이유"에 가장 알맞은 지문은 1번입니다.

1번 지문에서는 해구가 형성되는 이유를 설명하고 있습니다. 구체적으로, 대륙 지각과 해양 지각의 충돌이 섭입대를 형성하며, 이로 인해 해양 지각이 대륙 지각 아래로 침식되어 해구가 만들어진다고 언급하고 있습니다. 이는 해구의 형성과 관련된 지질학적 원인을 잘 설명하고 있습니다. 반면, 2번 지문은 해양 지각과 대륙 지각의 구성과 충돌 시의 현상에 대해 설명하고 있지만, 직접적으로 해구의 형성 이유에 대한 설명보다는 지각의 밀도 차이에 따른 섭입 현상에 집중하고 있습니다.

03 IR 파이프라인 구축: LLM의 한계: 매번 달라지는 아웃풋

“해구가 생기는 이유”에 알맞는 지문을 선택해주세요

1번. 섭입대는 대륙 지각과 해양 지각이 충돌하는 지점을 나타내는 지질 현상입니다. 이러한 충돌 지점에서는 다양한 지질 특성이 관찰될 수 있습니다. 그 중 하나는 해구입니다. 해구는 해양 지각이 대륙 지각을 침식하여 형성된 해안 지형입니다. 해구는 흐르는 물의 힘에 의해 형성되며, 대개 반달 모양을 가지고 있습니다. 이러한 해구는 섭입대의 끝 부분에 위치할 수 있으며, 섭입대의 지질 특성 중 하나로 포함될 수 있습니다. 섭입대의 다른 지질 특성으로는 퇴적층, 암석의 굽은 형태, 균열 등이 있을 수 있습니다. 이러한 지질 특성들은 학생이 대륙 지각과 해양 지각이 충돌하는 섭입대를 보여주는 도표를 만들 때 고려해야 할 중요한 요소입니다.

2번. 지구의 지각은 주로 현무암과 화강암으로 구성되어 있습니다. 현무암은 주로 해양 지각을 형성하고, 화강암은 대륙 지각을 형성합니다. 판이 충돌할 때, 해양 지각은 대체로 대륙 지각 아래로 잠기는 현상이 발생합니다. 이는 해양 지각이 대륙 지각보다 밀도가 높기 때문입니다. 해양 지각은 더 무거우므로 대륙 지각 아래로 잠기게 됩니다. 이러한 현상을 통해 지구의 지각은 지속적으로 변화하고 있습니다.

gpt-4o-mini-2024-07-18 2번 선택

gpt4o 1번 선택

gpt o1-preview 2번 선택

질문 **“해구가 생기는 이유”**에 가장 알맞는 지문은 2번입니다.

2번 지문은 해양 지각과 대륙 지각이 충돌할 때 밀도가 더 높은 해양 지각이 대륙 지각 아래로 잠기는 섭입 현상을 설명하고 있습니다. 이 과정에서 해양 지각이 아래로 내려가면서 해저에 깊은 골이 형성되는데, 이것이 바로 해구입니다. 해구는 이러한 판의 움직임과 섭입 현상 때문에 생기는 지형입니다.

반면에 1번 지문은 해구를 해양 지각이 대륙 지각을 침식하여 형성된 해안 지형으로 설명하고 있습니다. 또한 해구가 흐르는 물의 힘에 의해 형성되고 반달 모양을 가진다고 하는데, 이는 해구의 실제 형성 원리와 맞지 않습니다.

따라서 2번 지문이 해구가 생기는 이유를 정확하게 설명하고 있으므로, 질문에 가장 적합한 답변입니다.

03 IR 파이프라인 구축: but, LLM의 역할

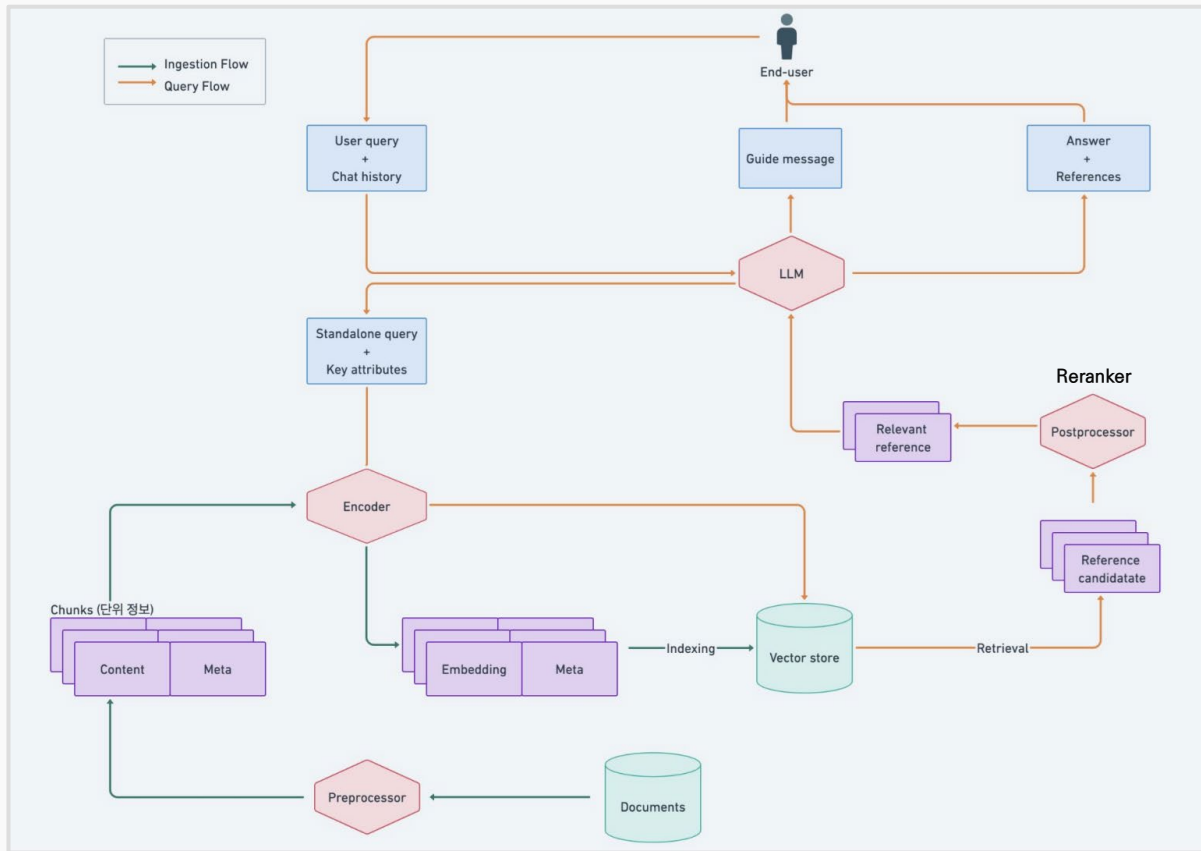
- 정답없는 문제: {"eval_id": 56, "standalone_query": "파리가 생태계에 미치는 영향에 대해 알려줘.", "topk": []}

```
{"docid":"fdf2729f-f2f6-4aef-bb57-152a4215e5df","src":"ko_ai2_arc_ARC_Challenge_validation","content":"파리는 한 동물로서 여섯 개의 다리를 가지고 있습니다. 파리는 곤충 중 하나로서 작은 크기와 날개를 가지고 있습니다. 파리는 주로 썩은 음식이나 동물 배설물에 서식하며, 사람들에게는 귀찮은 해충으로 알려져 있습니다. 파리는 빠른 속도로 날아다니며, 사람들이나 동물들에게 짜증을 주기도 합니다. 그러나 파리는 생태계에서 중요한 역할을 하고 있으며, 다른 동물들의 먹이로서도 기여합니다. 따라서 파리는 여섯 개의 다리를 가진 동물 중 하나로서 가장 가능성이 높은 후보입니다.", "new_domains":"high_school_biology"}
```

⇒ LLM은 핵심답안이 단 한 문장만 있는 경우, 해당 문서를 오답으로 판단할 수 있다.
구체적인 정보 부족 여부, 불필요한 세부 정보 판단 능력으로 주제의 부적합성을 판단한다.

⇒ 문서를 작은 chunk단위로 잘라서 reranker가 판단할 경우
LLM 사용없이도 높은 성능을 보장할 수 있으나,
전체 문맥을 종합하여 적합하지 않은 문서를 판단해야하는 능력이 부족하다.

03 IR 파이프라인 구축: workflow



04

인사이트

04 인사이트 _ 문서자체 특성 분석의 중요성

문서 길이의 다양성

문서당 글자 수가 최소 44자에서 최대 1,230자까지 분포하며, 표준편차가 104자로 편차가 큼을 확인했습니다. 이는 문서 길이가 균일하지 않아 이후의 처리 과정에서 하나의 접근 방식으로는 한계가 있음을 시사합니다.

체크 분할 필요성

평균 글자 수와 표준편차를 고려하여 문서를 의미적으로 분할하면 문서의 의미 파악과 검색 성능 향상에 도움이 될 것으로 판단했습니다

멀티턴 대화 데이터의 비중

평가 데이터 중 멀티턴 대화 데이터가 9.09%를 차지하며, 이는 대화의 맥락 이해가 요구되는 시나리오를 포함하고 있음을 의미합니다. 따라서 멀티턴 대화에 대한 별도의 처리 전략이 필요했습니다.

04 인사이트 _ 검색 쿼리 확장

메타 정보의 활용

LLM을 활용하여 문서의 제목, 키워드, 요약, 연관 카테고리 등의 메타 정보를 생성하였습니다. 이는 검색 과정에서 추가적인 피처로 활용되었으나, 세팅에 따라 결과가 많이 달라져서 세밀한 조정이 필요합니다.

카테고리 분류의 어려움

50%에 달하는 ARC 데이터는 도메인 정보가 없고, MMLU 정보도 일부 불명확하여, LLM을 활용하여 연관 카테고리로 재분류하였습니다. 때문에 검색에 직접 활용하기에는 한계가 있었습니다.

04 인사이트 _ Chunking Strategy

나이프 청크의 한계

고정된 글자 수나 문장 단위로 청크를 분할하는 방식은 문장의 논리적 구조를 파괴하거나, 의미적으로 뚜렷하지 않은 청크를 생성하여 검색 성능을 저하시킬 수 있었습니다.

의미 기반 청크(Semantic Chunking)의 도입

문장 간의 의미적 유사도를 계산하여 의미적으로 일관된 청크를 생성하였습니다. Sentence Transformer로 문장 임베딩을 생성하고, 코사인 유사도를 활용하여 유사도를 계산하였습니다.

클러스터링 적용

최대 청크 수를 제한하기 위해 K-Means 클러스터링을 적용하여 청크를 그룹화하였습니다. 이를 통해 청크의 개수를 조절하면서도 의미적 일관성을 유지할 수 있었습니다.

04 인사이트 _ Standalone Query & Prompt Engineering

효과적인 쿼리 생성의 중요성

사용자 대화로부터 검색에 적합한 Standalone Query를 생성하는 것은 정확한 문서 검색에 핵심적인 역할을 합니다.

LLM 모델 비교

Anthropic의 Claude, Gemma2, OpenAI의 GPT-4 등 다양한 LLM을 활용하여 Standalone Query를 생성하였으며, 모델별로 생성된 쿼리의 품질과 검색 성능에 차이가 있었습니다.

프롬프트 엔지니어링의 역할

원하는 출력 형식을 명확하게 지시하고, 예시를 포함한 프롬프트를 설계하여 LLM이 일관된 형식의 출력을 생성하도록 유도하였습니다. 이를 통해 쿼리의 품질을 높이고 검색 성능을 향상시킬 수 있었습니다.

04 인사이트 _ IR 파이프라인 구축 _ Embedding Selection

모델 성능 비교

다양한 임베딩 모델을 테스트한 결과, 한국어로 학습된 모델이 다국어 모델보다 우수한 성능을 보였습니다. 특히 **snunlp/KR-SBERT-V40K-klueNLI-augSTS**와 같은 한국어 특화 모델이 높은 MAP 점수를 기록하였습니다.

임베딩 차원의 영향

임베딩 차원이 높다고 해서 항상 성능이 향상되는 것은 아니었습니다. 실험 결과, 임베딩 차원을 축소하였을 때도 유사한 성능을 보였으며, 이는 과학 지식 문서의 특성상 스파스 벡터가 밀집 벡터보다 효과적일 수 있음을 시사합니다.

파인튜닝의 효과

문서를 기반으로 LLM을 활용하여 생성한 질의로 임베딩 모델을 파인튜닝하였을 때 성능이 향상되었습니다. 이는 도메인 특화된 임베딩이 검색 성능에 긍정적인 영향을 미침을 보여줍니다.

04 인사이트 _ Elasticsearch

유사도 알고리즘의 선택

Elasticsearch의 유사도 계산 방식 중 **LMJelinekMercer**를 적용하고, 람다(lambda) 값을 조절하여 검색 성능을 최적화하였습니다.
. 람다 값은 코퍼스 전체 빈도와 문서 내 빈도의 가중치를 조절하는 역할을 합니다.

Analyzer 및 Tokenizer 설정

한국어 처리를 위해 **nori** analyzer를 사용하고, 사용자 정의 사전과 동의어 사전을 추가하여 인덱싱과 검색 시의 언어적 특성을 반영 하였습니다.

Dense Vector 설정

임베딩 벡터를 인덱싱할 때 **dense_vector** 필드를 활용하고, 유사도 측정 방식을 **cosine**에서 **l2_norm**으로 변경하여 벡터의 크기와 방향 모두를 고려한 유사도 계산을 수행하였습니다.

04 인사이트 _ 복합 검색(Retrieval) 기법의 적용

하이브리드 검색의 효과

스파스 검색(BM25)과 밀집 벡터 검색(Dense Retrieval)을 결합한 하이브리드 검색 방식을 적용하여 단일 검색 방식보다 우수한 성능을 얻었습니다.

ColBERT의 활용

ColBERT를 추가하여 문서의 세부적인 토큰 수준의 매칭을 강화하였으며, 특히 nbits 설정을 통해 인덱싱 효율성과 검색 성능 간의 균형을 맞추었습니다.

문서 청크 기반 검색

전체 문서보다 의미적으로 분할된 청크 단위를 검색하고 랭킹하는 것이 더 높은 성능을 보였습니다. 이는 청크 단위로 문서의 핵심 정보를 보다 정확하게 매칭할 수 있음을 나타냅니다.

04 인사이트 _ 재순위화(Reranking) 및 크로스 인코더의 적용

재순위화의 필요성

초기 검색 단계에서의 문서 순위는 완벽하지 않으므로, 상위 문서에 대한 재순위화를 통해 검색 성능을 향상시켰습니다.

크로스 인코더 모델의 활용

`bongsoo/kluce-cross-encoder-v1` 모델을 사용하여 질의와 문서 간의 관련성을 정밀하게 평가하였으며, 파인튜닝을 통해 성능을 더욱 향상시켰습니다.

학습 데이터의 중요성

긍정 및 부정 예시를 포함한 고품질의 학습 데이터를 생성하는 것이 크로스 인코더의 성능 향상에 핵심적이었습니다. LLM을 활용하여 문서 당 다수의 질의를 생성하고, 이를 기반으로 학습 데이터를 구축하였습니다.

04 인사이트 _ LLM 프롬프트 엔지니어링과 한계점

일관된 출력 형식 확보

LLM을 활용하여 문서 ID를 선별하거나 쿼리를 생성할 때, 일관된 출력 형식을 유지하는 것이 중요했습니다. 이를 위해 프롬프트에 출력 예시와 형식에 대한 명확한 지시를 포함하였습니다.

LLM의 불확실성 처리

LLM은 동일한 입력에 대해 일관되지 않은 출력을 생성할 수 있으므로, 이러한 불확실성을 최소화하기 위한 전략이 필요했습니다. 예를 들어, 다수의 모델을 ensemble 하거나, 출력의 신뢰도를 평가하는 방법 등을 고려하였습니다.

LLM의 역할 한계

LLM이 모든 상황에서 최적의 판단을 내리지 못하는 것을 확인했습니다. 특히, 문맥이 부족한 경우나 추론이 필요한 경우에는 LLM의 응답이 불안정할 수 있었습니다. 따라서 LLM의 활용 범위를 명확히 하고, 필요한 경우 다른 모델이나 알고리즘으로 보완하였습니다.

04 최종 인사이트

다단계 접근법의 효과성

데이터 전처리부터 임베딩, 검색, 재순위화, 그리고 LLM의 활용까지 이어지는 다단계 접근법이 전체적인 성능 향상에 크게 기여했습니다.

모델 및 설정 간의 균형

최신 모델이나 복잡한 알고리즘을 사용하는 것만이 능사가 아니며, 각 단계에서의 설정과 모델 선택이 전체 시스템의 효율성과 성능에 미치는 영향을 고려해야 했습니다.

협업과 지식 공유의 중요성

팀원 간의 긴밀한 협업과 실험 결과의 공유를 통해 빠른 문제 해결과 성능 향상을 이룰 수 있었습니다. 다양한 아이디어를 적극적으로 시도하고 피드백을 주고받는 과정이 프로젝트의 성공에 핵심적이었습니다.

현실적인 제약 사항 고려

성능 향상을 위해 복잡도를 무작정 높이는 것보다는, 검색 시간과 시스템 자원 등 현실적인 제약 사항을 고려하여 효율적인 솔루션을 찾는 것이 중요했습니다.

05

프로젝트 결과

05 프로젝트 결과:






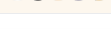
Public 3등

리더보드 [중간 순위] 리더보드 [최종 순위]

대회 진행 중 제공되는 리더보드는 평가 데이터셋 일부를 사용하여 채점된 결과입니다.

🔄 새로고침

마지막 업데이트: 2024.10.25 11:18:53

순위	팀이름	팀멤버	MAP	MRR	제출횟수	최종 제출
내등수 3	IR 2조		0.9288	0.9348	175	2d
1	ir 7팀		0.9561	0.9591	84	6d
2	IR 8 조		0.9530	0.9576	168	1d
3	IR 2조		0.9288	0.9348	175	2d
4	IR_3조		0.9167	0.9212	61	3d
5	IR 1조		0.9106	0.9136	93	17h



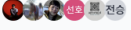



Private1등

리더보드 [중간 순위] 리더보드 [최종 순위]

대회 종료 시점에는 나머지 채점에 활용되지 않았던 나머지 평가 데이터셋을 사용하여 때문에 대회 최종 순위는 변동될 수 있습니다.

🔄 새로고침

마지막 업데이트: 2024.10.25 11:19:22

순위	팀이름	팀멤버	MAP	MRR	제출횟수	최종 제출
내등수 1	IR 2조		0.9189	0.9197	175	3d
1	IR 2조		0.9189	0.9197	175	3d
2	ir 7팀		0.9136	0.9167	84	6d
3	IR_3조		0.9068	0.9061	61	6d
4	IR6		0.9061	0.9091	110	2d
5	IR 8 조		0.8962	0.8970	168	1d



reranking(co...twice



0.9227
0.9273

0.9258
0.9288

2024.10.22 08:26

완료

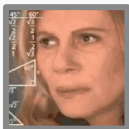


public과 private 모두 점수가 0.92대인 robust 모델은 combined_dialogue 활용 모델

06

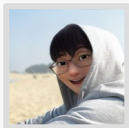
회고

진행소감



이윤재

인간의 독해능력이 얼마나 복잡한 과정을 거치는지 CS로 학습한 느낌. Reranker의 우수한 성능에 놀랐다. Train에 성공하지 못해서 아쉽다. Prompt engineering의 중요성을 알았다. LLM의 방대하고 체계적인 지식 수준에서 나오는 불완전한 아웃풋이 마치 인간과 똑같았다. Reranker의 능력을 뛰어넘은 LLM의 독해능력은 도메인, 카테고리, hierarchy이 잘 갖춰진 확장된 배경지식과 동의어, 반대어 등의 학습과 긴 호흡으로 매 문장 옳은지 아닌지를 판단하는 처리 능력에서 나온다.



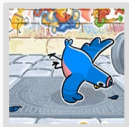
이재명

챗봇을 어떻게 만들 수 있는지 조금 이해가 되었다.



장은지

검색 성능과 검색 시간 사이의 균형을 맞추는 것이 어려웠다. 성능을 높이기 위해 여러 단계를 추가하다 보면 검색 시간이 길어지기도 하고, 검색 시간을 줄이려면 단계를 줄여야 할지 고민해보기도 하고 여러 가지 실험을 해볼 수 있는 좋은 기회였다. 실제로 여러가지 아이디어들을 시도했지만, 이번 대회가 부트캠프 동안 쌓아온 지식의 총집합처럼 느껴졌기 때문에, 만약 앞에서 더 열심히 공부했다더라면 이번 기회에 더 많은 것을 적용해볼 수 있었을 것이라는 아쉬움이 있다. 그리고 뛰어나신 팀원분들 덕분에 많이 배울 수 있었다.



전백찬

부트캠프의 시작과 끝을 RAG와 함께 보내서 의미있었고, 3번 연속으로 대회를 우승하게 되어 신기했다.



최지미

서비스에 적용할 수 있는 기술을 사용하고 관련 학습 할 수 있어서 좋았다.

07

Q&A