

Ch_11 리랭커(Reranker)

두 단계 검색(Two-Stage Retrieval System)에서 사용되는 핵심 컴포넌트

- 대규모 데이터셋에서 효율적이고 정확한 검색을 수행하기 위해 설계됨
- 주로 Retriever가 찾아낸 문서들의 순위를 재조정하는 역할
 - 초기 검색 결과의 정확도를 향상시키는 것을 목표

Retriever: 대규모 문서 집합에서 관련성 있는 후보 문서들을 빠르게 추출

Reranker: 이 후보 문서들을 더 정교하게 분석하여 최종적인 순위를 결정

작동 원리

1. Retriever로부터 초기 검색 결과를 입력받음.
2. 쿼리와 각 후보 문서를 쌍으로 결합하여 처리함.
3. 복잡한 모델(주로 트랜스포머 기반)을 사용하여 각 쿼리-문서 쌍의 관련성을 평가함.
4. 평가 결과에 따라 문서들의 순위를 재조정 함.
5. 최종적으로 재순위화된 결과를 출력함.

I. Cross Encoder Reranker

| 검색된 문서들의 순위를 재조정하여 질문에 가장 관련성 높은 문서를 상위로 올림

1. 개요 (Overview)

Cross Encoder Reranker는

RAG(Retrieval-Augmented Generation) 시스템의 검색 성능을 향상시키기 위해 사용되
는 기술이다.

- Retriever가 가져온 문서 후보들 중
 질문과 가장 관련성 높은 문서를 다시 정렬(reranking) 하는 역할
- Hugging Face의 Cross Encoder 모델을 활용해 구현 가능

2. 목적 (Purpose)

- 검색된 문서들의 순위를 재조정
 - 질문(Query)에 가장 적합한 문서를 상위로 배치
 - 최종적으로 RAG 응답 품질 향상
-

3. 구조 (Architecture)

입력 방식

- 질문(Query) + 문서(Document) 를 동시에 하나의 입력으로 처리

출력

- 질문-문서 간의 직접적인 유사도 점수
-

4. 작동 방식 (How it Works)

1. 질문과 문서를 하나의 입력으로 결합
 2. **Self-Attention 메커니즘**을 통해
 - 질문과 문서를 동시에 분석
 - 단어 간 상호작용을 정밀하게 고려
 3. 질문-문서 간 **의미적 유사도 점수**를 직접 계산
 4. 점수를 기준으로 문서 재정렬 (Reranking)
-

5. 장점 (Advantages)

- 정확한 유사도 측정
 - 질문과 문서 사이의 **심층적인 의미론적 관계 파악**
 - 검색 결과 품질 크게 개선
 - RAG 시스템 전체 성능 향상
-

6. 한계점 (Limitations)

- 연산 비용이 높음
 - 처리 시간이 상대적으로 김
 - 대규모 문서 전체에 직접 적용하기 어려움
-

7. 실제 사용 방식 (Practical Usage)

일반적인 RAG 파이프라인:

1. Bi-Encoder (Retriever)

- 빠르게 대량 문서에서 후보 추출

2. Cross Encoder (Reranker)

- 상위 문서만 대상으로 정확한 재정렬

| 속도는 Bi-Encoder, 정확도는 Cross Encoder

8. 구현 방법 (Implementation)

사용 가능한 모델

- Hugging Face Cross Encoder 모델
- `BAAI/bge-reranker` 계열 모델

프레임워크 통합

- LangChain
 - `CrossEncoderReranker` 컴포넌트 제공
 - 비교적 쉽게 RAG 파이프라인에 결합 가능

9. Reranker 사용 시 문서 수 설정

- 일반적으로 상위 5~10개 문서에 대해 reranking 수행
- 최적의 문서 수는:
 - 실험과 평가를 통해 결정 필요

10. Trade-offs (고려 사항)

항목	Trade-off
정확도	↑ 하지만 처리 시간 증가
성능	↑ 하지만 계산 비용 증가
검색 속도	↓ 하지만 관련성 정확도 향상
시스템 요구사항	하드웨어 자원 고려 필요

11. 핵심 요약 (Key Takeaways)

- Cross Encoder Reranker는 RAG 성능을 끌어올리는 핵심 요소
- 정확하지만 느림 → 제한된 문서 수에만 사용
- Bi-Encoder + Cross Encoder 조합이 실무 표준
- 모델 선택과 문서 수 설정이 성능에 큰 영향

II. Cohere Reranker

기업이 인간-기계 상호작용을 개선할 수 있도록 돕는 자연어 처리 모델을 제공하는 캐나다의 스타트업.

`retriever` 에서 Cohere의 rerank endpoint를 사용하는 방법을 보여줌.

- Reranker를 API로 제공하는 사례

III. Jina Reranker

문서 압축 및 `retrieval` 을 위해 **Jina Reranker**를 사용하는 방법을 보여줌.

- API도 있고 로컬도 있음 (실습에서는 API로 활용)

IV. FlashRank Reranker

FI기존 검색 및 retrieval 파이프라인에 재순위를 추가하기 위한 초경량 및 초고속 Python 라이브러리.

SoTA cross-encoders를 기반으로 함.

문서 압축 및 `retrieval` 을 위해 flashrank를 사용하는 방법을 보여줌.

- 로컬 라이브러리

세 서비스 비교

Cohere Reranker

- Cross Encoder를 API로 제공

- 코드 몇 줄이면 바로 사용
- 비용 있음 & 외부 서버 의존

“Reranker를 서비스로 쓰는 사례”

Jina Reranker

- API도 있고 로컬도 있음
- 실습이나 데모에서는 API를 많이 씀
- 선택지가 유연함

“API/로컬 둘 다 되는 중간 포지션”

FlashRank

- Cross Encoder를 아주 가볍게 로컬에서 실행
- 빠르고 비용 없음
- 대규모 서비스보다는 실험/경량 환경에 적합

“초경량 로컬 Cross Encoder Reranker”