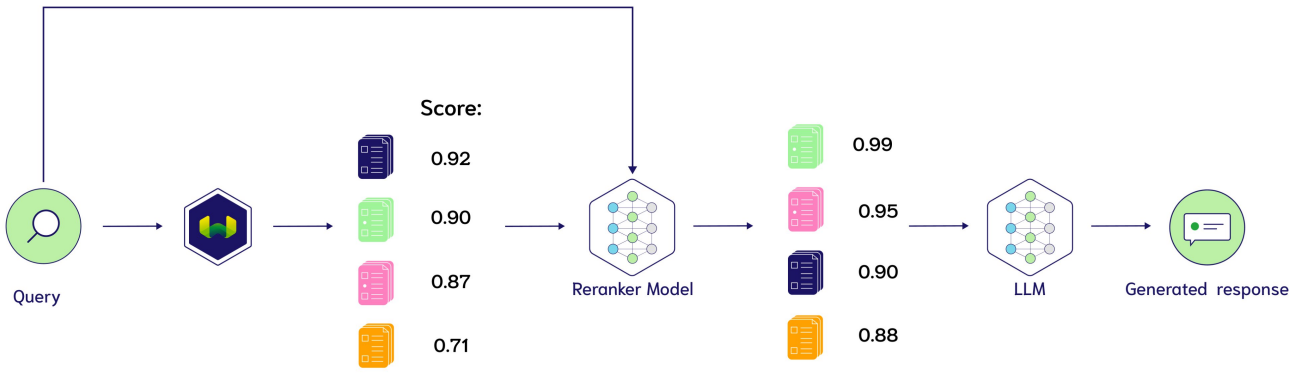
리랭커(Reranker)

Fast campus

Reranker(리랭커)는 현대적인 두 단계 검색 시스템(Two-Stage Retrieval System)에서 사용되는 핵심 컴포넌트입니다. 대규모 데이터셋에서 효율적이고 정확한 검색을 수행하기 위해 설계되었으며, 주로 첫 번째 단계인 Retriever가 찾아낸 문서들의 순위를 재조정하는 역할을 합니다.

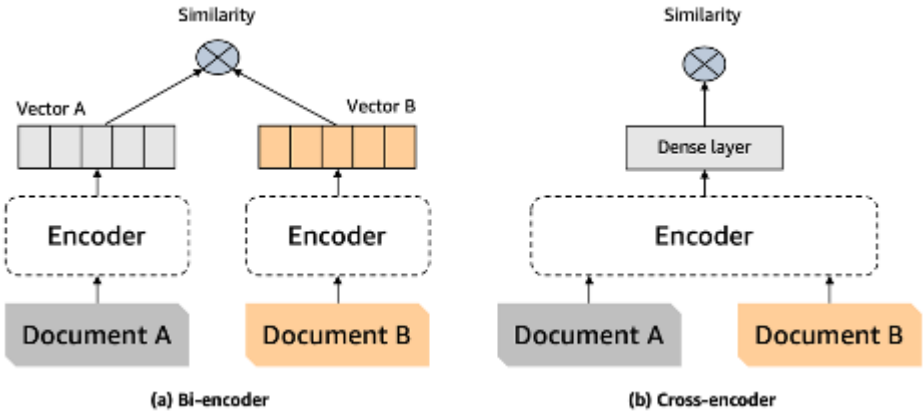


> 출처: https://x.com/ecardenas300/status/1732472798272974915

개요

Reranker는 검색 시스템의 두 번째 단계에서 작동하며, 초기 검색 결과의 정확도를 향상시키는 것을 목표로 합니다. Retriever가 대규모 문서 집합에서 관련성 있는 후보 문서들을 빠르게 추출한 후, Reranker는 이 후보 문서들을 더 정교하게 분석하여 최종적인 순위를 결정합니다.

작동 원리



> 출처: https://aws.amazon.com/ko/blogs/tech/korean-reranker-rag/

1. Retriever로부터 초기 검색 결과를 입력받습니다.
2. 쿼리와 각 후보 문서를 쌍으로 결합하여 처리합니다.
3. 복잡한 모델(주로 트랜스포머 기반)을 사용하여 각 쿼리-문서 쌍의 관련성을 평가합니다.
4. 평가 결과에 따라 문서들의 순위를 재조정합니다.
5. 최종적으로 재순위화된 결과를 출력합니다.

Retrieval 단계의 임베딩 벡터 비교

Retrieval 단계에서는 주로 벡터 임베딩을 사용하여 문서를 검색합니다.

동작 방식

1. 질문과 문서를 각각 벡터로 인코딩합니다.
2. 벡터 간의 유사도(주로 내적)를 계산하여 관련성을 측정합니다.
3. 유사도가 높은 상위 k개의 문서를 선택합니다.

이 방법은 매우 큰 문서 집합에서 빠르게 관련 문서를 찾을 수 있지만, 정확도가 다소 떨어질 수 있습니다.

Reranker의 동작 방식

Reranker는 retrieval 단계 이후에 작동하며, 주로 BERT와 같은 언어 모델을 사용합니다.

동작 과정

1. Retrieval 단계에서 선택된 상위 k개의 문서를 입력으로 받습니다.
2. 각 문서와 원래 질문을 함께 언어 모델에 입력합니다.
3. 모델은 문서와 질문 간의 관련성을 더 정확하게 평가합니다.
4. 평가 결과에 따라 문서들을 재정렬합니다.

Reranker는 문맥을 고려한 심층적인 의미 분석을 수행하여 **더 정확한 결과**를 제공합니다

계산 비용 비교

Retrieval

- 계산 비용이 상대적으로 낮습니다.
- 대규모 문서 집합에서도 빠르게 검색할 수 있습니다.
- 주로 벡터 간 내적 연산을 사용하므로 효율적입니다.

Reranker

- 계산 비용이 retrieval 단계보다 높습니다.
- BERT와 같은 복잡한 언어 모델을 사용하므로 더 많은 연산이 필요합니다.
- 그러나 retrieval 단계에서 선별된 소수의 문서만 처리하므로 전체 시스템의 효율성은 유지됩니다.

성능과 효율성 비교

- Retrieval 단계는 빠르지만 정확도가 다소 떨어질 수 있습니다.
- Reranker는 더 정확하지만 계산 비용이 높습니다.

- 두 방식을 결합하면 빠른 속도와 높은 정확도를 동시에 얻을 수 있습니다.

일반적으로 retrieval 단계에서는 상위 5~10개의 문서를 선택하고, 이를 reranker에 입력하여 최종 결과를 얻습니다. 이러한 방식으로 계산 비용과 정확도 사이의 균형을 유지할 수 있습니다.결론적으로, reranker는 retrieval 단계보다 더 정확한 결과를 제공하지만 계산 비용이 높습니다. 그러나 두 방식을 적절히 조합하면 효율적이고 정확한 검색 시스템을 구축할 수 있습니다

기술적 특징

아키텍처

- 주로 BERT, RoBERTa 등의 트랜스포머 기반 모델 사용
- 교차 인코더(Cross-encoder) 구조 채택

입력 형식

- 일반적으로 `[CLS] Query [SEP] Document [SEP]` 형태로 입력

학습 방법

1. 포인트와이즈(Pointwise): 개별 쿼리-문서 쌍의 관련성 점수 예측
2. 페어와이즈(Pairwise): 두 문서 간의 상대적 관련성 비교
3. 리스트와이즈(Listwise): 전체 순위 목록을 한 번에 최적화

Retriever와의 차이점

특성	Retriever	Reranker
목적	관련 문서 빠른 검색	정확한 순위 조정
처리 방식	간단한 유사도 계산	복잡한 의미 분석
모델 구조	단일 인코더	교차 인코더

특성	Retriever	Reranker
연산 복잡도	낮음	높음
우선순위	속도	정확도
입력 형태	쿼리와 문서 개별 처리	쿼리-문서 쌍 처리
출력	후보 문서 대규모 집합	정확한 순위와 점수
확장성	높음	제한적

장단점

장점

- 검색 정확도 크게 향상
- 복잡한 의미적 관계 모델링 가능
- 첫 단계 검색의 한계 보완

단점

- 계산 비용 증가
- 처리 시간 증가
- 대규모 데이터셋에 직접 적용 어려움

벤치마크

Embedding	WithoutReranker		bge-reranker-base		bge-reranker-large		Cohere-Reranker	
	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR	Hit Rate	MRR
OpenAI	0.876404	0.718165	0.91573	0.832584	**0.910112**	**0.855805**	**0.926966**	**0.86573**
bge-large	0.752809	0.597191	0.859551	0.805243	0.865169	0.816011	0.876404	0.822753
llm-embedder	0.814607	0.587266	0.870787	0.80309	0.876404	0.824625	0.882022	0.830243
Cohere-v2	0.780899	0.570506	0.876404	0.798127	0.876404	0.825281	0.876404	0.815543
Cohere-v3	0.825843	0.624532	0.882022	0.806086	0.882022	0.834644	0.88764	0.836049
Voyage	0.831461	0.68736	0.926966	0.837172	**0.91573**	**0.858614**	**0.91573**	**0.851217**
JinaAI-Small	0.831461	0.614045	0.91573	0.843071	**0.926966**	**0.857303**	**0.926966**	**0.868633**
JinaAI-Base	0.848315	0.68221	0.938202	0.846348	**0.938202**	**0.868539**	**0.932584**	**0.873689**
Google-PaLM	0.865169	0.719476	0.910112	0.833708	**0.910112**	**0.85309**	**0.910112**	**0.855712**

> 출처: https://www.llamaindex.ai/blog/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83