

- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: <https://smith.langchain.com/hub/teddynote/summary-stuff-documents>

✓ CH11 리랭커 (Reranker)

- **Reranker**
 - 현대적인 두 단계 검색 시스템(Two-Stage Retrieval System)에서 사용되는 핵심 컴포넌트
 - 대규모 데이터셋에서 효율적 이고 정확한 검색을 수행하기 위해 설계됨
 - 주로 첫 번째 단계인 Retriever가 찾아낸 문서들의 순위를 재조정하는 역할

- **개요**
 - 검색 시스템의 두 번째 단계에서 작동
 - 목표: 초기 검색 결과의 정확도를 향상시키는 것
 - Retriever가 대규모 문서 집합에서 관련성 있는 후보 문서를 빠르게 추출 → Reranker는 이 후보 문서들을 더 정교하게 분석 → 최종적인 순위 결정

- **작동 원리**
 - a. Retriever로부터 초기 검색 결과를 입력받음
 - b. 쿼리와 각 후보 문서를 쌍으로 결합 → 처리
 - c. 복잡한 모델(주로 트랜스포머 기반)을 사용하여 각 쿼리-문서 쌍의 관련성 평가
 - d. 평가 결과 → 문서들의 순위를 재조정
 - e. 최종적으로 재순위화 된 결과를 출력

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

- **입력 형식**

[CLS] Query [SEP] Document [SEP]

- **학습 방법**

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

- **Retriever** 와의 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
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

▼ 1. **Cross Encoder Reranker**



1) 개요

- **RAG** (검색 증강 생성) 시스템의 성능을 향상시키기 위해 사용되는 기술
 - Hugging Face의 cross encoder 모델 사용 → retriever에서 reranker 구현하는 방법 사용해보기
-



2) 주요 특징 및 작동 방식

- **목적**
 - 검색된 문서들의 순위 재조정 → 질문에 가장 관련성 높은 문서를 상위로 올림
 - **구조**
 - 질문, 문서 동시에 입력으로 받아 처리함
 - **작동 방식**
 - 질문 - 문서를 하나의 입력으로 사용 → 유사도 직접 출력
 - Self-attention 메커니즘 → 질문 - 문서 동시에 분석
 - **장점**
 - 더 정확한 유사도 측정 가능
 - 질문-문서 사이의 의미론적 유사성을 깊이 탐색
 - **한계점**
 - 연산 비용이 높고 시간이 오래 걸림
 - 대규모 문서 집합에 직접 적용하기 어려움
-



3) 실제 사용

- 일반적으로 초기 검색에서 상위 k개의 문서에 대해서만 reranking 수행
 - Bi-encoder로 빠르게 후보 추출 → Cross encoder로 정확도 높이는 방식으로 활용
-



4) 구현

- Hugging Face의 cross encoder 모델 or BAAI/bge-reranker와 같은 모델 사용

- `LangChain` 등의 프레임워크에서 `CrossEncoderReranker` 컴포넌트 통해 쉽게 통합 가능
-

5) `Reranker` 의 주요 장점

- 더 정확한 유사도 측정
 - 심층적인 의미론적 유사성 탐색
 - 검색 결과 개선
 - `RAG` 시스템 성능 향상
 - 유연한 통합
 - 다양한 사전 학습 모델 선택 가능
-

6) `Reranker` 사용 시 문서 수 설정

- 일반적으로 상위 5~10개 문서에 대해 `reranking` 수행
 - 최적의 문서 수 = 실험과 평가를 통해 결정 필요
-

7) `Reranker` 사용시 Trade-offs

- a. 정확도 vs 처리 시간
 - b. 성능 향상 vs 계산 비용
 - c. 검색 속도 vs 관련성 정확도
 - d. 시스템 요구사항 충족
 - e. 데이터셋 특성 고려 간단한 예시를 통한 `Cross Encoder Reranker`의 구현방법
-

```
# 문서 출력 도우미 함수
def pretty_print_docs(docs):
    print(
        f"\n{'-' * 100}\n".join(
            [f"Document {i+1}:\n\n" + d.page_content for i, d in enumerate(docs)]
        )
    )
```

```
from langchain_community.document_loaders import TextLoader
from langchain_community.vectorstores import FAISS
from langchain_huggingface import HuggingFaceEmbeddings
```

```

from langchain_text_splitters import RecursiveCharacterTextSplitter

# 문서 로드
documents = TextLoader("../11_Reranker/data/appendix-keywords.txt").load()

# 텍스트 분할기 설정
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)

# 문서 분할
texts = text_splitter.split_documents(documents)

# 임베딩 모델 설정
embeddingsModel = HuggingFaceEmbeddings(
    model_name="sentence-transformers/msmarco-distilbert-dot-v5"
)

```

- **Hugging Face** 모델 설치 결과 (**38.7s**)

-  허깅페이스 모델 설치 결과

```
# 문서로부터 FAISS 인덱스 생성 및 검색기 설정
```

```

retriever = FAISS.from_documents(texts, embeddingsModel).as_retriever(
    search_kwargs={"k": 10}
)

```

2.4s

```
# 질의 설정
```

```
query = "Word2Vec 에 대해서 알려줄래?"
```

```
# 질의 수행 및 결과 문서 반환
```

```
docs = retriever.invoke(query)
```

```
# 결과 문서 출력
```

```
pretty_print_docs(docs)
```

- 결과 문서 출력 하기 (**0.7s**)

Document 1:

Open Source

정의: 오픈 소스는 소스 코드가 공개되어 누구나 자유롭게 사용, 수정, 배포할 수 있는 소프트웨어를 의미합니다.

예시: 리눅스 운영 체제는 대표적인 오픈 소스 프로젝트입니다.

연관키워드: 소프트웨어 개발, 커뮤니티, 기술 협업

Structured Data

정의: 구조화된 데이터는 정해진 형식이나 스키마에 따라 조직된 데이터입니다. 이는 데이터베이스, 스프레드시트

예시: 관계형 데이터베이스에 저장된 고객 정보 테이블은 구조화된 데이터의 예입니다.

연관키워드: 데이터베이스, 데이터 분석, 데이터 모델링

Parser

Document 2:

정의: LLM은 대규모의 텍스트 데이터로 훈련된 큰 규모의 언어 모델을 의미합니다. 이러한 모델은 다양한 자연

예시: OpenAI의 GPT 시리즈는 대표적인 대규모 언어 모델입니다.

연관키워드: 자연어 처리, 딥러닝, 텍스트 생성

FAISS (Facebook AI Similarity Search)

정의: FAISS는 페이스북에서 개발한 고속 유사성 검색 라이브러리로, 특히 대규모 벡터 집합에서 유사 벡터를

예시: 수백만 개의 이미지 벡터 중에서 비슷한 이미지를 빠르게 찾는 데 FAISS가 사용될 수 있습니다.

연관키워드: 벡터 검색, 머신러닝, 데이터베이스 최적화

Open Source

Document 3:

InstructGPT

정의: InstructGPT는 사용자의 지시에 따라 특정한 작업을 수행하기 위해 최적화된 GPT 모델입니다. 이 모

예시: 사용자가 "이메일 초안 작성"과 같은 특정 지시를 제공하면, InstructGPT는 관련 내용을 기반으로 0

연관키워드: 인공지능, 자연어 이해, 명령 기반 처리

Keyword Search

정의: 키워드 검색은 사용자가 입력한 키워드를 기반으로 정보를 찾는 과정입니다. 이는 대부분의 검색 엔진과

예시: 사용자가 "커피숍 서울"이라고 검색하면, 관련된 커피숍 목록을 반환합니다.

연관키워드: 검색 엔진, 데이터 검색, 정보 검색

Page Rank

Document 4:

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일

예시: 문장 "나는 학교에 간다"를 "나는", "학교에", "간다"로 분할합니다.

연관키워드: 토큰화, 자연어 처리, 구문 분석

Tokenizer

정의: 토큰라이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는

예시: "I love programming."이라는 문장을 ["I", "love", "programming", "."]으로 분할합니

연관키워드: 토큰화, 자연어 처리, 구문 분석

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 작업에 유용합니다.
예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다.
연관키워드: 임베딩, 데이터베이스, 벡터화

SQL

Document 5:

Page Rank

정의: 페이지 랭크는 웹 페이지의 중요도를 평가하는 알고리즘으로, 주로 검색 엔진 결과의 순위를 결정하는 데 사용됩니다.
예시: 구글 검색 엔진은 페이지 랭크 알고리즘을 사용하여 검색 결과의 순위를 정합니다.
연관키워드: 검색 엔진 최적화, 웹 분석, 링크 분석

데이터 마이닝

정의: 데이터 마이닝은 대량의 데이터에서 유용한 정보를 발굴하는 과정입니다. 이는 통계, 머신러닝, 패턴 인식 등을 포함합니다.
예시: 소매업체가 고객 구매 데이터를 분석하여 판매 전략을 수립하는 것은 데이터 마이닝의 예입니다.
연관키워드: 빅데이터, 패턴 인식, 예측 분석

멀티모달 (Multimodal)

Document 6:

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방법입니다.
예시: 사용자가 "태양계 행성"이라고 검색하면, "목성", "화성" 등과 같이 관련된 행성에 대한 정보를 반환합니다.
연관키워드: 자연어 처리, 검색 알고리즘, 데이터 마이닝

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트 데이터를 이해할 수 있습니다.
예시: "사과"라는 단어를 $[0.65, -0.23, 0.17]$ 과 같은 벡터로 표현합니다.
연관키워드: 자연어 처리, 벡터화, 딥러닝

Token

Document 7:

판다스 (Pandas)

정의: 판다스는 파이썬 프로그래밍 언어를 위한 데이터 분석 및 조작 도구를 제공하는 라이브러리입니다. 이는 데이터 분석을 위한 필수 도구입니다.
예시: 판다스를 사용하여 CSV 파일을 읽고, 데이터를 정제하며, 다양한 분석을 수행할 수 있습니다.
연관키워드: 데이터 분석, 파이썬, 데이터 처리

GPT (Generative Pretrained Transformer)

정의: GPT는 대규모의 데이터셋으로 사전 훈련된 생성적 언어 모델로, 다양한 텍스트 기반 작업에 활용됩니다.
예시: 사용자가 제공한 질문에 대해 자세한 답변을 생성하는 챗봇은 GPT 모델을 사용할 수 있습니다.
연관키워드: 자연어 처리, 텍스트 생성, 딥러닝

InstructGPT

Document 8:

멀티모달 (Multimodal)

정의: 멀티모달은 여러 종류의 데이터 모드(예: 텍스트, 이미지, 소리 등)를 결합하여 처리하는 기술입니다.
예시: 이미지와 설명 텍스트를 함께 분석하여 더 정확한 이미지 분류를 수행하는 시스템은 멀티모달 기술의 예입니다.
연관키워드: 데이터 융합, 인공지능, 딥러닝

Document 9:

Crawling

정의: 크롤링은 자동화된 방식으로 웹 페이지를 방문하여 데이터를 수집하는 과정입니다. 이는 검색 엔진 최적화
예시: 구글 검색 엔진이 인터넷 상의 웹사이트를 방문하여 콘텐츠를 수집하고 인덱싱하는 것이 크롤링입니다.
연관키워드: 데이터 수집, 웹 스크래핑, 검색 엔진

Word2Vec

정의: Word2Vec은 단어를 벡터 공간에 매핑하여 단어 간의 의미적 관계를 나타내는 자연어 처리 기술입니다.
예시: Word2Vec 모델에서 "왕"과 "여왕"은 서로 가까운 위치에 벡터로 표현됩니다.
연관키워드: 자연어 처리, 임베딩, 의미론적 유사성
LLM (Large Language Model)

Document 10:

JSON

정의: JSON(JavaScript Object Notation)은 경량의 데이터 교환 형식으로, 사람과 기계 모두에게 읽기
예시: {"이름": "홍길동", "나이": 30, "직업": "개발자"}는 JSON 형식의 데이터입니다.
연관키워드: 데이터 교환, 웹 개발, API

Transformer

정의: 트랜스포머는 자연어 처리에서 사용되는 딥러닝 모델의 한 유형으로, 주로 번역, 요약, 텍스트 생성 등에
예시: 구글 번역기는 트랜스포머 모델을 사용하여 다양한 언어 간의 번역을 수행합니다.
연관키워드: 딥러닝, 자연어 처리, Attention

HuggingFace

- 기본 retriever 를 ContextualCompressionRetriever 로 감싸기
- CrossEncoderReranker → HuggingFaceCrossEncoder 사용해 반환된 결과를 재정렬함
 - 다국어 지원 BGE Reranker: [bge-reranker-v2-m3](#)


```
from langchain.retrievers import ContextualCompressionRetriever
from langchain.retrievers.document_compressors import CrossEncoderReranker
from langchain_community.cross_encoders import HuggingFaceCrossEncoder

# 모델 초기화
model = HuggingFaceCrossEncoder(model_name="BAAI/bge-reranker-v2-m3")

# 상위 3개의 문서 선택
compressor = CrossEncoderReranker(model=model, top_n=3)

# 문서 압축 검색기 초기화
compression_retriever = ContextualCompressionRetriever(
    base_compressor=compressor, base_retriever=retriever
)

# 압축된 문서 검색
compressed_docs = compression_retriever.invoke("Word2Vec 에 대해서 알려줄래?")
```

- bge-reranker-v2-m3 설치 결과 (2m 27.3s)
 -  bge-reranker-v2-m3 설치 결과

```
# 문서 출력
pretty_print_docs(compressed_docs)
```

- compressed docs 출력 결과

Document 1:

Crawling

정의: 크롤링은 자동화된 방식으로 웹 페이지를 방문하여 데이터를 수집하는 과정입니다. 이는 검색 엔진 최적화
 예시: 구글 검색 엔진이 인터넷 상의 웹사이트를 방문하여 콘텐츠를 수집하고 인덱싱하는 것이 크롤링입니다.
 연관키워드: 데이터 수집, 웹 스크래핑, 검색 엔진

Word2Vec

정의: Word2Vec은 단어를 벡터 공간에 매핑하여 단어 간의 의미적 관계를 나타내는 자연어 처리 기술입니다.
 예시: Word2Vec 모델에서 "왕"과 "여왕"은 서로 가까운 위치에 벡터로 표현됩니다.
 연관키워드: 자연어 처리, 임베딩, 의미론적 유사성

LLM (Large Language Model)

Document 2:

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일

예시: 문장 "나는 학교에 간다"를 "나는", "학교에", "간다"로 분할합니다.

연관키워드: 토큰화, 자연어 처리, 구문 분석

Tokenizer

정의: 토큰라이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는

예시: "I love programming."이라는 문장을 ["I", "love", "programming", "."]으로 분할합니다.

연관키워드: 토큰화, 자연어 처리, 구문 분석

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 쿼리

예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다.

연관키워드: 임베딩, 데이터베이스, 벡터화

SQL

Document 3:

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하

예시: 사용자가 "태양계 행성"이라고 검색하면, "목성", "화성" 등과 같이 관련된 행성에 대한 정보를 반환합

연관키워드: 자연어 처리, 검색 알고리즘, 데이터 마이닝

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해

예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다.

연관키워드: 자연어 처리, 벡터화, 딥러닝

Token

-
- next: **02. Cohere Reranker**
-