---

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

---

## 3. `Jina Reranker`

- 문서 압축 및 `retriever`을 위해 `Jina Reranker` 사용하는 예제
- `Jina Reranker API KEY 발급`

---

### 1) `기본 설정`

```
# API KEY를 환경변수로 관리하기 위한 설정 파일
from dotenv import load_dotenv

# API KEY 정보로드
load_dotenv()                       # true
```

```
from langsmith import Client
from langsmith import traceable

import os

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_V2')
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANGCHAIN_API_KEY') else "설정되지 않음" # API 키 값은 직접 출력하지 않음

if langchain_tracing_v2 == "true" and os.getenv('LANGCHAIN_API_KEY') and langchain_project:
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='{langchain_tracing_v2}')")
    print(f"✅ LangSmith 프로젝트: '{langchain_project}'")
    print(f"✅ LangSmith API Key: {langchain_api_key_status}")
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.")
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다. 다음을 확인하세요:")
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설정되어 있지 않습니다 (현재: '{langchain_tracing_v2}').")
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습니다.")
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습니다.")
```

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
✅ LangSmith 프로젝트: 'LangChain-prantice'
✅ LangSmith API Key: 설정됨
-> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

### 2) `Jina Rerankder`

```
# 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_text_splitters import RecursiveCharacterTextSplitter
from langchain_huggingface import HuggingFaceEmbeddings
```

```
import warnings

# 경고 무시
warnings.filterwarnings("ignore")

# 임베딩
embeddings = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2",
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
    )

embeddings = embeddings

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

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

# 문서 분할
texts = text_splitter.split_documents(documents)                        # 9.0s
```

```
# 검색기 초기화
retriever = FAISS.from_documents(
    texts, embeddings).as_retriever(search_kwargs={"k": 10})

# 질의문
query = "Word2Vec 에 대해서 설명해줘."

# 문서 검색
docs = retriever.invoke(query)

# 문서 출력
pretty_print_docs(docs)
```

- `retriever`로 출력하기 (`0.5s`)

```
Document 1:

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다.
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 텍스트를 이해하고 처리할 수 있게 합
예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다.
연관키워드: 자연어 처리, 벡터화, 딥러닝

Token
----------------------------------------------------------------------------------------------------
Document 2:

Parser

정의: 파서는 주어진 데이터(문자열, 파일 등)를 분석하여 구조화된 형태로 변환하는 도구입니다. 이는 프로그래밍 언어의 구문 분석이나 파일 데이터 처리에
예시: HTML 문서를 구문 분석하여 웹 페이지의 DOM 구조를 생성하는 것은 파싱의 한 예입니다.
연관키워드: 구문 분석, 컴파일러, 데이터 처리

TF-IDF (Term Frequency-Inverse Document Frequency)

정의: TF-IDF는 문서 내에서 단어의 중요도를 평가하는 데 사용되는 통계적 척도입니다. 이는 문서 내 단어의 빈도와 전체 문서 집합에서 그 단어의 희소성을
예시: 많은 문서에서 자주 등장하지 않는 단어는 높은 TF-IDF 값을 가집니다.
연관키워드: 자연어 처리, 정보 검색, 데이터 마이닝

Deep Learning
----------------------------------------------------------------------------------------------------
Document 3:

HuggingFace

정의: HuggingFace는 자연어 처리를 위한 다양한 사전 훈련된 모델과 도구를 제공하는 라이브러리입니다. 이는 연구자와 개발자들이 쉽게 NLP 작업을 수행
예시: HuggingFace의 Transformers 라이브러리를 사용하여 감정 분석, 텍스트 생성 등의 작업을 수행할 수 있습니다.
```

연관키워드: 자연어 처리, 딥러닝, 라이브러리

Digital Transformation

정의: 디지털 변환은 기술을 활용하여 기업의 서비스, 문화, 운영을 혁신하는 과정입니다. 이는 비즈니스 모델을 개선하고 디지털 기술을 통해 경쟁력을 높이는
예시: 기업이 클라우드 컴퓨팅을 도입하여 데이터 저장과 처리를 혁신하는 것은 디지털 변환의 예입니다.
연관키워드: 혁신, 기술, 비즈니스 모델

Crawling
---
Document 4:

JSON

정의: JSON(JavaScript Object Notation)은 경량의 데이터 교환 형식으로, 사람과 기계 모두에게 읽기 쉬운 텍스트를 사용하여 데이터 객체를 표현
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 사용됩니다. 이는 Attention 메커니즘을 기
예시: 구글 번역기는 트랜스포머 모델을 사용하여 다양한 언어 간의 번역을 수행합니다.
연관키워드: 딥러닝, 자연어 처리, Attention

HuggingFace
---
Document 5:

Page Rank

정의: 페이지 랭크는 웹 페이지의 중요도를 평가하는 알고리즘으로, 주로 검색 엔진 결과의 순위를 결정하는 데 사용됩니다. 이는 웹 페이지 간의 링크 구조를
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 등의 기술을 활용합니다.
예시: 소매업체가 고객 구매 데이터를 분석하여 판매 전략을 수립하는 것은 데이터 마이닝의 예입니다.
연관키워드: 빅데이터, 패턴 인식, 예측 분석

멀티모달 (Multimodal)
---
Document 6:

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다.
예시: 문장 "나는 학교에 간다"를 "나는", "학교에", "간다"로 분할합니다.
연관키워드: 토큰화, 자연어 처리, 구문 분석

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다.
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 분석 작업에 사용됩니다.
예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다.
연관키워드: 임베딩, 데이터베이스, 벡터화

SQL
---
Document 7:

DataFrame

정의: DataFrame은 행과 열로 이루어진 테이블 형태의 데이터 구조로, 주로 데이터 분석 및 처리에 사용됩니다.
예시: 판다스 라이브러리에서 DataFrame은 다양한 데이터 타입의 열을 가질 수 있으며, 데이터 조작과 분석을 용이하게 합니다.
연관키워드: 데이터 분석, 판다스, 데이터 처리

```
Attention 메커니즘

정의: Attention 메커니즘은 딥러닝에서 중요한 정보에 더 많은 '주의'를 기울이도록 하는 기법입니다. 이는 주로 시퀀스 데이터(예: 텍스트, 시계열 데이
예시: 번역 모델에서 Attention 메커니즘은 입력 문장의 중요한 부분에 더 집중하여 정확한 번역을 생성합니다.
연관키워드: 딥러닝, 자연어 처리, 시퀀스 모델링

판다스 (Pandas)
--------------------------------------------------------------------------------------------------------
Document 8:

Open Source

정의: 오픈 소스는 소스 코드가 공개되어 누구나 자유롭게 사용, 수정, 배포할 수 있는 소프트웨어를 의미합니다. 이는 협업과 혁신을 촉진하는 데 중요한 역
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 등에서 쉽게 검색하고 분석할 수 있습니다.
예시: 관계형 데이터베이스에 저장된 고객 정보 테이블은 구조화된 데이터의 예입니다.
연관키워드: 데이터베이스, 데이터 분석, 데이터 모델링

Parser
--------------------------------------------------------------------------------------------------------
Document 9:

판다스 (Pandas)

정의: 판다스는 파이썬 프로그래밍 언어를 위한 데이터 분석 및 조작 도구를 제공하는 라이브러리입니다. 이는 데이터 분석 작업을 효율적으로 수행할 수 있게
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. 이는 입력된 텍스트에 기반하여 자연스러운 언
예시: 사용자가 제공한 질문에 대해 자세한 답변을 생성하는 챗봇은 GPT 모델을 사용할 수 있습니다.
연관키워드: 자연어 처리, 텍스트 생성, 딥러닝

InstructGPT
--------------------------------------------------------------------------------------------------------
Document 10:

Deep Learning

정의: 딥러닝은 인공신경망을 이용하여 복잡한 문제를 해결하는 머신러닝의 한 분야입니다. 이는 데이터에서 고수준의 표현을 학습하는 데 중점을 둡니다.
예시: 이미지 인식, 음성 인식, 자연어 처리 등에서 딥러닝 모델이 활용됩니다.
연관키워드: 인공신경망, 머신러닝, 데이터 분석

Schema

정의: 스키마는 데이터베이스나 파일의 구조를 정의하는 것으로, 데이터가 어떻게 저장되고 조직되는지에 대한 청사진을 제공합니다.
예시: 관계형 데이터베이스의 테이블 스키마는 열 이름, 데이터 타입, 키 제약 조건 등을 정의합니다.
연관키워드: 데이터베이스, 데이터 모델링, 데이터 관리

DataFrame
```

---

### 3) `JinaReranker를 사용한 재정렬 수행`

- **`Jina Reranker`** → 압축기를 사용해 기본 `retriever`를 **`ContextualCompressionRetriever`** 로 감싸기

```
from ast import mod
from langchain.retrievers import ContextualCompressionRetriever
from langchain_community.document_compressors import JinaRerank
from dotenv import load_dotenv
import os

# .env 파일 로드
load_dotenv()
```

```
# JINA_API_KEY 로드
jina_api_key = os.getenv("JINA_API_KEY")

# JinaRerank 압축기 초기화
compressor = JinaRerank(
    model="jina-reranker-v2-base-multilingual",
    top_n=3,
    jina_api_key=jina_api_key)

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

# 관련 문서 검색 및 압축
compressed_docs = compression_retriever.invoke(
    "Word2Vec 에 대해서 설명해줘."
)                                                                    # 1.3s
```

```
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)]
        )
    )
```

```
# 압축된 문서의 보기 좋게 출력

pretty_print_docs(compressed_docs)
```

- `Jina Reranker`로 정렬

```
Document 1:

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다.
예시: 문장 "나는 학교에 간다"를 "나는", "학교에", "간다"로 분할합니다.
연관키워드: 토큰화, 자연어 처리, 구문 분석

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다.
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 분석 작업에 사용됩니다.
예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다.
연관키워드: 임베딩, 데이터베이스, 벡터화

SQL
----------------------------------------------------------------------------------------------------
Document 2:

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다.
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 텍스트를 이해하고 처리할 수 있게 합
예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다.
연관키워드: 자연어 처리, 벡터화, 딥러닝

Token
----------------------------------------------------------------------------------------------------
Document 3:

Parser

정의: 파서는 주어진 데이터(문자열, 파일 등)를 분석하여 구조화된 형태로 변환하는 도구입니다. 이는 프로그래밍 언어의 구문 분석이나 파일 데이터 처리에
```

예시: HTML 문서를 구문 분석하여 웹 페이지의 DOM 구조를 생성하는 것은 파싱의 한 예입니다.
연관키워드: 구문 분석, 컴파일러, 데이터 처리

TF-IDF (Term Frequency-Inverse Document Frequency)

정의: TF-IDF는 문서 내에서 단어의 중요도를 평가하는 데 사용되는 통계적 척도입니다. 이는 문서 내 단어의 빈도와 전체 문서 집합에서 그 단어의 희소성
예시: 많은 문서에서 자주 등장하지 않는 단어는 높은 TF-IDF 값을 가집니다.
연관키워드: 자연어 처리, 정보 검색, 데이터 마이닝

Deep Learning

---


- *next:* ***`04. FlashRank Reranker`***


---