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

---

시멘틱 청커 (SemanticChunker)

- **텍스트를 의미론적 유사성에 기반에 분할**
- 텍스트: **문장 단위**로 분할 → **3개의 문장**씩 **그룹화** → **임베딩 공간**에서 **유사한 문장**들을 **병합**하는 과정
- 참고: **Greg Kamradt의 노트북**

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install -qU langchain_experimental langchain_openai
```

```
# data/appendix-keywords.txt 파일을 열어서 f라는 파일 객체 생성하기
with open("../07_Text_Splitter/data/appendix-keywords.txt") as f:
    file = f.read()                                          # 파일의 내용을 읽어서 file 변수에 저장
```

```
# 파일으로부터 읽은 내용 일부 출력하기

print(file[:500])
```

- 셀 출력

```
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

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
```

`SemanticChunker` 생성

- **`SemanticChunker`**
  - `LangChain`의 실험적 기능 중 하나
  - **텍스트를 `의미론적으로 유사한 청크로 분할`하는 역할**
  - **이를 통해 텍스트 데이터를 보다 `효과적으로 처리하고 분석`할 수 있음**

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

# API 키 정보 로드
load_dotenv()                                    # True
```

- **`SemanticChunker` 사용 → 텍스트를 `의미적으로 관련된 청크`로 분할**

- 참고: **`Gemini API Embedding`**
- `API`로 `gemini-embedding` 사용하기

```
    # API 키를 환경변수로 관리하기 위한 설정 파일
    from dotenv import load_dotenv
    from google import genai
    from google.genai import types
    import os

    # API 키 정보 로드
    load_dotenv()

    client = genai.Client(api_key=os.environ["GOOGLE_API_KEY"])

    result = client.models.embed_content(
            model="gemini-embedding-001",
            contents="What is the meaning of life?",
            )

    print(result.embeddings)
```

- 셀 출력 (0.7s)

```
Both GOOGLE_API_KEY and GEMINI_API_KEY are set. Using GOOGLE_API_KEY.
```

```
[ContentEmbedding(                       # 텍스트 엠베딩 = 숫자로 이루어진 리스트
values=[                                 # 텍스트의 의미 = 수치로 표현
    -0.022374554,
    -0.004560777,
    0.013309286,
    -0.0545072,
    -0.02090443,
    <... 3067 more items ...>,           # 일부분만 출력함
]
)]
```

- **`ERROR`** 발생
- 참고: **`LangChain에서 Gemini-Embedding 모델 사용하기`**

```
# 테스트

import google.generativeai as genai

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

result = genai.embed_content(
    model="models/gemini-embedding-001",
    content="What is the meaning of life?",
    task_type="retrieval_document"                                  # 명시 추가
)
print(result['embedding'][:10])                                     # 성공 시 임베딩 벡터 출력
```

- 셀 출력 (2.0s)

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1758246073.593504  190364 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste

- 컴퓨터가 인터넷에 연결할 때 사용하는 특별한 보안 시스템(ALTS)에 관련된 경고 메시지
- ALTS 인증 무시됨 = 특별한 보안 시스템을 사용하지 않기로 결정했다는 의미
- GCP에서 실행되지 않음
- 신뢰할 수 없는 ALTS가 활성화되지 않음 = 신뢰할 수 없는 보안 시스템을 사용하지 않기로 결정했다는 의미

→ GCP에서 작동하지 않을 때 나타나는 메시지
→ 로컬에서 사용하는 경우 나타날 수 있는 일반적인 메시지
```

```
[-0.017152296, -0.0006449693, 0.012324803, -0.08141434, -0.0023675598, 0.011479777, 0.0008151301, 0.00025921015,
```

```
from langchain_experimental.text_splitter import SemanticChunker
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from dotenv import load_dotenv
import os

load_dotenv()

# API 키 확인
if not os.getenv("GOOGLE_API_KEY"):
    os.environ["GOOGLE_API_KEY"] = input("Enter your Google API key: ")

# Gemini 임베딩 모델 생성 (task_type 명시)
embeddings = GoogleGenerativeAIEmbeddings(
    model="models/gemini-embedding-001",
    task_type="retrieval_document",                                  # 또는 "semantic_similarity"
    google_api_key=os.getenv("GOOGLE_API_KEY")
)

text_splitter = SemanticChunker(embeddings)
```

- 셀 출력

```
E0000 00:00:1758246173.630190  190364 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untruste
```

텍스트 분할

- **`text_splitter`** → **`file`** **텍스트를** **문서 단위**로 분할

```
# 긴 텍스트를 작은 배치로 나누어 호출 (타임아웃 방지, 이전 오류 = 런타임 오류)

from langchain_text_splitters import RecursiveCharacterTextSplitter

batch_splitter = RecursiveCharacterTextSplitter(chunk_size=100000, chunk_overlap=0)         # 대략 10만자 단위

batches = batch_splitter.split_text(file)                                                   # file을 배치로 분할

all_chunks = []

for batch in batches:
    chunks = text_splitter.split_text(batch)
    all_chunks.extend(chunks)

print(all_chunks[0])                                                                        # 첫 번째 청크 출력
```

- 셀 출력 (2.3s)

```
Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계
```

```
Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. 예시: 문장 "나는 학교에 간다"를

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다. 예시: "I love programm

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
```

텍스트 분할

- **`text_splitter` → `file` 텍스트를 문서 단위로 분할해보기**

```
# 교재대로 분할해보기

chunks = text_splitter.split_text(file)         # 1.9s 소요
```

- 분할된 청크 확인해보기

```
# 분할된 청크 중 첫 번째 청크 출력해보기

print(chunks[0])
```

- 셀 출력

```
Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. 예시: 문장 "나는 학교에 간다"를

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다. 예시: "I love programm

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
```

- **`create_document()` 함수 → 청크를 문서로 변환하기**

```
# text_splitter → 분할하기
docs = text_splitter.create_documents([file])

# 분할된 문서 중 첫 번째 문서의 내용을 출력해보기
print(docs[0].page_content)
```

- 셀 출력 (1.9s)

```
Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. 예시: 문장 "나는 학교에 간다"

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다. 예시: "I love programm

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
```

`Breakpoints`

- 이 chunker는 문장을 **분리** 할 시점을 결정하여 작동
  - **두 문장 간의 임베딩 차이** 를 살펴봄으로써 이루어짐
  - 그 **차이** 가 **특정 임계값** 을 **넘으면 문장이 분리**
- 참고 영상

---

1) `Percentile`

- 기본적인 분리 방식 = **백분위수** (**`Percentile`**) 를 기반으로 함
- **문장 간의 모든 차이** 를 **계산** → **지정한 백분위수** 를 **기준** 으로 **분리**

```
# 시멘틱 청커 초기화
text_splitter = SemanticChunker(
    embeddings,                    # gemini-embedding 모델 사용
    # 분할 기준점 유형 = 백분위수로 설정하기
    breakpoint_threshold_type="percentile",
    breakpoint_threshold_amount=70,
)
```

- 분할된 결과 확인하기

```
docs = text_splitter.create_documents([file])
for i, doc in enumerate(docs[:5]):
    print(f"[Chunk {i}]", end="\n\n")
    print(doc.page_content)                                    # 분할된 문서 중 첫 번째 문서의 내용 출력하기
    print("===" * 20)
```

- 셀 출력 (2.1s)

```
[Chunk 0]

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계

Embedding
```

```
정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합
============================================================
[Chunk 1]

예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다.
============================================================
[Chunk 2]

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.
============================================================
[Chunk 3]

예시: "I love programming."이라는 문장을 ["I", "love", "programming", "."]으로 분할합니다. 연관키워드: 토큰화, 자연어 처리, 구문 분

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
============================================================
[Chunk 4]

예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화

SQL

정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양
============================================================
```

- `docs`의 길이 출력해보기

```
print(len(docs))                    # 27
```

---

2) `Standard Deviation`

- **`지정한` breakpoint_threshold_amount `표준편차보다 큰 차이가 있는 경우 분할`**
- `breakpoint_threshold_type` 매개변수 = `"standard_deviation"` 설정 → **`청크 분할 기준` = `표준편차` 기준으로 지정**

```
# 시멘틱 청커 초기화
text_splitter = SemanticChunker(
    embeddings,                     # gemini-embedding 모델 사용
    # 분할 기준점 유형 = 표준 편차로 설정하기
    breakpoint_threshold_type="standard_deviation",
    breakpoint_threshold_amount=1.25,
)
```

- 분할된 결과 확인하기

```
# text_splitter를 사용하여 분할하기

docs = text_splitter.create_documents([file])           # 2.2s 소요
```

```
docs = text_splitter.create_documents([file])
for i, doc in enumerate(docs[:5]):
    print(f"[Chunk {i}]", end="\n\n")
```

```
    print(doc.page_content)                                    # 분할된 문서 중 첫 번째 문서의 내용 출력하기
    print("===" * 20)
```

- 셀 출력 (1.8s)

```
[Chunk 0]

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합
============================================================
[Chunk 1]

예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. 예시: 문장 "나는 학교에 간다"를

Tokenizer

정의: 토크나이저는 텍스트 데이터를 토큰으로 분할하는 도구입니다. 이는 자연어 처리에서 데이터를 전처리하는 데 사용됩니다. 예시: "I love programm

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
============================================================
[Chunk 2]

예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화

SQL

정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양
============================================================
[Chunk 3]

예시: SELECT * FROM users WHERE age > 18;은 18세 이상의 사용자 정보를 조회합니다. 연관키워드: 데이터베이스, 쿼리, 데이터 관리

CSV

정의: CSV(Comma-Separated Values)는 데이터를 저장하는 파일 형식으로, 각 데이터 값은 쉼표로 구분됩니다. 표 형태의 데이터를 간단하게 저장하고

JSON

정의: JSON(JavaScript Object Notation)은 경량의 데이터 교환 형식으로, 사람과 기계 모두에게 읽기 쉬운 텍스트를 사용하여 데이터 객체를 표현합

Transformer

정의: 트랜스포머는 자연어 처리에서 사용되는 딥러닝 모델의 한 유형으로, 주로 번역, 요약, 텍스트 생성 등에 사용됩니다. 이는 Attention 메커니즘을 기
============================================================
[Chunk 4]

예시: 구글 번역기는 트랜스포머 모델을 사용하여 다양한 언어 간의 번역을 수행합니다. 연관키워드: 딥러닝, 자연어 처리, Attention

HuggingFace

정의: HuggingFace는 자연어 처리를 위한 다양한 사전 훈련된 모델과 도구를 제공하는 라이브러리입니다. 이는 연구자와 개발자들이 쉽게 NLP 작업을 수행

Digital Transformation

정의: 디지털 변환은 기술을 활용하여 기업의 서비스, 문화, 운영을 혁신하는 과정입니다. 이는 비즈니스 모델을 개선하고 디지털 기술을 통해 경쟁력을 높이는
============================================================
```

- `docs`의 길이 출력해보기

```
print(len(docs))                    # 14
```

---

3) `Interquartile`

- **`사분위수 범위`(`interquartile range`)를 사용하여 청크를 `분할`**
- `breakpoint_threshold_type` 매개변수 = `"interquartile"` 설정 → **청크 분할 기준을 `사분위수 범위`로 지정**

```
# 시멘틱 청커 초기화
text_splitter = SemanticChunker(
    embeddings,                     # gemini-embedding 모델 사용
    # 분할 기준점 유형 = 사분위수로 설정하기
    breakpoint_threshold_type="interquartile",
    breakpoint_threshold_amount=0.5,
)
```

```
# text_splitter를 사용하여 분할하기
docs = text_splitter.create_documents([file])

# 결과 출력하기
for i, doc in enumerate(docs[:5]):
    print(f"[Chunk {i}]", end="\n\n")
    print(doc.page_content)                             # 분할된 문서 중 첫 번째 문서의 내용 출력하기
    print("===" * 20)
```

- 셀 출력 (2.0s)

```
[Chunk 0]

Semantic Search

정의: 의미론적 검색은 사용자의 질의를 단순한 키워드 매칭을 넘어서 그 의미를 파악하여 관련된 결과를 반환하는 검색 방식입니다. 예시: 사용자가 "태양계

Embedding

정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합
============================================================
[Chunk 1]

예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝

Token

정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다.
============================================================
[Chunk 2]

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 programm

VectorStore

정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다.
============================================================
[Chunk 3]

예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화
```

```
SQL

정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양
============================================================
[Chunk 4]

예시: SELECT * FROM users WHERE age > 18;은 18세 이상의 사용자 정보를 조회합니다. 연관키워드: 데이터베이스, 쿼리, 데이터 관리

CSV

정의: CSV(Comma-Separated Values)는 데이터를 저장하는 파일 형식으로, 각 데이터 값은 쉼표로 구분됩니다. 표 형태의 데이터를 간단하게 저장하고
============================================================
```

- `docs`의 길이 출력해보기

```
print(len(docs))                    # 22
```

- **`Breakpoints`별 임베딩 차이**

	분리 방식 (기준)	분할된 문서 (p1)
① Percentile	백분위수	[Chunk 0] 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 Embedding 정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합니다. == [Chunk 1] 예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝 Token 정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. == [Chunk 2] 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. == [Chunk 3] 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. == [Chunk 4] 예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화 SQL 정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양한 작업을 수행할 수 있 ==
② Standard Deviation	표준편차	[Chunk 0] 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 Embedding 정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합니다. == [Chunk 1] 예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝 Token 정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. == [Chunk 2] 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" VectorStore 정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다. == [Chunk 3] 예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화 SQL 정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양한 작업을 수행할 수 있 == [Chunk 4]

	분리 방식 (기준)	분할된 문서 (p1)
③ Interquartile	사분위수 범위	예시: SELECT * FROM users WHERE age > 18;은 18세 이상의 사용자 정보를 조회합니다. 연관키워드: 데이터베이스, 쿼리, 데이터 관리 CSV 정의: CSV(Comma-Separated Values)는 데이터를 저장하는 파일 형식으로, 각 데이터 값은 쉼표로 구분됩니다. 표 형태의 데이터를 간단하게 저장하고 교환할 때 사용됩니다 ========== [Chunk 0] 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 Embedding 정의: 임베딩은 단어나 문장 같은 텍스트 데이터를 저차원의 연속적인 벡터로 변환하는 과정입니다. 이를 통해 컴퓨터가 텍스트를 이해하고 처리할 수 있게 합니다. ========== [Chunk 1] 예시: "사과"라는 단어를 [0.65, -0.23, 0.17]과 같은 벡터로 표현합니다. 연관키워드: 자연어 처리, 벡터화, 딥러닝 Token 정의: 토큰은 텍스트를 더 작은 단위로 분할하는 것을 의미합니다. 이는 일반적으로 단어, 문장, 또는 구절일 수 있습니다. ========== [Chunk 2] 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" VectorStore 정의: 벡터스토어는 벡터 형식으로 변환된 데이터를 저장하는 시스템입니다. 이는 검색, 분류 및 기타 데이터 분석 작업에 사용됩니다. ========== [Chunk 3] 예시: 단어 임베딩 벡터들을 데이터베이스에 저장하여 빠르게 접근할 수 있습니다. 연관키워드: 임베딩, 데이터베이스, 벡터화 SQL 정의: SQL(Structured Query Language)은 데이터베이스에서 데이터를 관리하기 위한 프로그래밍 언어입니다. 데이터 조회, 수정, 삽입, 삭제 등 다양한 작업을 수행할 수 있 ========== [Chunk 4] 예시: SELECT * FROM users WHERE age > 18;은 18세 이상의 사용자 정보를 조회합니다. 연관키워드: 데이터베이스, 쿼리, 데이터 관리 CSV 정의: CSV(Comma-Separated Values)는 데이터를 저장하는 파일 형식으로, 각 데이터 값은 쉼표로 구분됩니다. 표 형태의 데이터를 간단하게 저장하고 교환할 때 사용됩니다 ==========

- *next:* ***`코드 분할 (Python, Markdown, JAVA, C++, C#, GO, JS, Latex 등)`***
