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

---

텍스트(TextLoader)

TXT Loader

- `.txt` 확장자를 가지는 파일을 로더로 로드하는 방법 살펴보기

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

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

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install langchain-community
```

```
from langchain_community.document_loaders import TextLoader                # TextLoader 임포트

# 텍스트 로더 생성
loader = TextLoader("../06_Document_Loader/data/appendix-keywords.txt")     # 텍스트 파일 경로

# 문서 로드
docs = loader.load()

# 문서의 수 출력
print(f"문서의 수: {len(docs)}\n")

# 메타데이터 출력
print("[메타데이터]\n")
print(docs[0].metadata)

# 앞부분 미리보기 출력
print("\n========== [앞부분] 미리보기 =========\n")
print(docs[0].page_content[:500])                                           # 앞부분 500자 출력
```

- 셀 출력 (0.3s)

```
문서의 수: 1

[메타데이터]

{'source': '../06_Document_Loader/data/appendix-keywords.txt'}

========== [앞부분] 미리보기 =========

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
```

`TextLoader`를 통한 파일 인코딩 자동 감지

- **`TextLoader` 클래스 사용**: 디렉토리에서 `임의의 파일 목록`을 `대량`으로 `로드`할 때 유용한 몇 가지 전략 살펴보기

- 먼저 문제를 설명하기 위해 임의의 인코딩으로 여러 개의 텍스트를 로드하기
  - **`silent_errors`**: 디렉토리 로더에 `silent_errors` 매개변수 전달 → **`로드할 수 없는 파일을 건너뛰고` 로드 프로세스 계속**
  - **`autodetect_encoding`**: 로더 클래스에 `자동 감지_인코딩` 전달 → **`실패하기 전에 파일 인코딩을 자동으로 감지하도록 요청`**

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install chardet
```

```
from langchain_community.document_loaders import DirectoryLoader        # DirectoryLoader 임포트

# 디렉토리 경로 설정
path = "../06_Document_Loader/data"

# 텍스트 로더 설정
text_loader_kwargs = {"autodetect_encoding": True}                      # 자동 인코딩 감지 설정

# 디렉토리 로더 생성
loader = DirectoryLoader(
    path,                                                               # 디렉토리 경로
    glob="**/*.txt",                                                    # 텍스트 파일 패턴
    loader_cls=TextLoader,                                              # 로더 클래스
    silent_errors=True,                                                 # 오류 무시 설정
    loader_kwargs=text_loader_kwargs,                                   # 로더 설정
)


# 문서 로드
docs = loader.load()
```

- `../data/appendix-keywords.txt` 파일과 **파일명이 유사한 파생 파일들** = 모두 인코딩 방식이 다른 파일들

```
doc_sources = [doc.metadata["source"] for doc in docs]
doc_sources                                                 # type(doc_sources) = <class 'list'>
```

```
['../06_Document_Loader/data/appendix-keywords-CP949.txt',
 '../06_Document_Loader/data/reference.txt',
 '../06_Document_Loader/data/appendix-keywords-EUCKR.txt',
 '../06_Document_Loader/data/chain-of-density.txt',
 '../06_Document_Loader/data/appendix-keywords.txt',
 '../06_Document_Loader/data/appendix-keywords-utf8.txt']
```

- 셀 출력

```
['../06_Document_Loader/data/appendix-keywords-CP949.txt',
'../06_Document_Loader/data/reference.txt',
'../06_Document_Loader/data/appendix-keywords-EUCKR.txt',
'../06_Document_Loader/data/chain-of-density.txt',
'../06_Document_Loader/data/appendix-keywords.txt',
'../06_Document_Loader/data/appendix-keywords-utf8.txt']
```

- 문서 하나씩 열어보기

```
# ../data/reference.txt

print("[메타데이터]\n")
print(docs[1].metadata)
print("\n========== [앞부분] 미리보기 =========\n")
print(docs[1].page_content[:500])
```

- 셀 출력

```
[메타데이터]

{'source': '../06_Document_Loader/data/reference.txt'}

========== [앞부분] 미리보기 ==========

1.
제목: [Digital Insight 2023-5] ChatGPT의 파급효과와 기관의 LLM 도입 전략
출처: https://www.nia.or.kr/site/nia_kor/ex/bbs/View.do?cbIdx=82618&bcIdx=26165&parentSeq=26165
파일명: [DI]_ChatGPT의_파급_효과와_기관의_LLM_도입_전략.pdf

2.
제목: [IF_23-6호]_인공지능_기술_발전과_일자리의_미래_최종
출처: https://www.nia.or.kr/site/nia_kor/ex/bbs/View.do?cbIdx=25932&bcIdx=25938&parentSeq=25938
파일명: [IF_23-6호]_인공지능_기술_발전과_일자리의_미래_최종.pdf
```

```
# ../data/appendix-keywords-EUCKR.txt
print("[메타데이터]\n")
print(docs[2].metadata)
print("\n========== [앞부분] 미리보기 =========\n")
print(docs[2].page_content[:500])
```

- 셀 출력

```
[메타데이터]

{'source': '../06_Document_Loader/data/appendix-keywords-EUCKR.txt'}

========== [앞부분] 미리보기 ==========

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
```

```
# ../data/chain-of-density.txt
print("[메타데이터]\n")
print(docs[3].metadata)
print("\n========== [앞부분] 미리보기 ==========\n")
print(docs[3].page_content[:500])
```

- 셀 출력

```
[메타데이터]

{'source': '../06_Document_Loader/data/chain-of-density.txt'}

========== [앞부분] 미리보기 ==========

Selecting the “right” amount of information to include in a summary is a difficult task.
A good summary should be detailed and entity-centric without being overly dense and hard to follow. To better und
```

```
# ../data/appendix-keywords.txt
print("[메타데이터]\n")
print(docs[4].metadata)
print("\n========== [앞부분] 미리보기 ==========\n")
print(docs[4].page_content[:500])
```

- 셀 출력

```
[메타데이터]

{'source': '../06_Document_Loader/data/appendix-keywords.txt'}

========== [앞부분] 미리보기 ==========

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
```

```
# ../data/appendix-keywords-utf8.txt
print("[메타데이터]\n")
print(docs[5].metadata)
print("\n========== [앞부분] 미리보기 ==========\n")
print(docs[5].page_content[:500])
```

- 셀 출력

```
[메타데이터]

{'source': '../06_Document_Loader/data/appendix-keywords-utf8.txt'}

========== [앞부분] 미리보기 ==========

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

- *next:* `JSON`
