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

---

`JSON`

- `.json` 확장자를 가지는 파일을 로더로 로드하는 방법 살펴보기
- 참고

```
# 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 jq                          # JSON 데이터 파싱, 필터링, 변환, 쿼리하는데 사용
```

- 사전에 `VS Code` 터미널에 설치할 것

```
      pip install langchain-community
```

```
# 필요한 라이브러리 임포트
import json                             # json 모듈 임포트
from pathlib import Path                # pathlib 모듈 임포트
from pprint import pprint               # pprint 모듈 임포트

# 파일 경로 설정
file_path = "../06_Document_Loader/data/people.json"

# JSON 파일 읽기
data = json.loads(Path(file_path).read_text())

# 데이터 출력
pprint(data)
```

- 셀 출력

```
[{'address': {'city': '서울', 'street': '312번지', 'zipCode': '83795'},
'age': 31,
'carOwnership': True,
'hobbies': ['요리', '음악 감상', '사진 촬영'],
'isMarried': True,
'name': '박시우',
'phoneNumbers': ['483-4639-1933', '947-4179-7976']},
{'address': {'city': '서울', 'street': '877번지', 'zipCode': '36780'},
'age': 31,
'carOwnership': True,
'hobbies': ['여행', '음악 감상', '등산'],
'isMarried': False,
'name': '정수아',
'phoneNumbers': ['337-5721-3227', '387-3768-9586']},
{'address': {'city': '서울', 'street': '175번지', 'zipCode': '89067'},
'age': 43,
```

```
'carOwnership': True,
'hobbies': ['등산', '독서', '게임'],
'isMarried': False,
'name': '최도윤',
'phoneNumbers': ['354-5563-4638', '471-9212-1826']},
{'address': {'city': '서울', 'street': '690번지', 'zipCode': '70635'},
'age': 22,
'carOwnership': False,
'hobbies': ['여행', '등산', '게임'],
'isMarried': False,
'name': '정민준',
'phoneNumbers': ['468-2796-2152', '922-5760-7030']},
{'address': {'city': '서울', 'street': '151번지', 'zipCode': '79118'},
'age': 79,
'carOwnership': False,
'hobbies': ['게임', '영화 감상', '음악 감상'],
'isMarried': True,
'name': '이민준',
'phoneNumbers': ['751-2823-8259', '722-7267-9516']},
{'address': {'city': '서울', 'street': '855번지', 'zipCode': '21216'},
'age': 64,
'carOwnership': True,
'hobbies': ['독서', '등산', '요리'],
'isMarried': False,
'name': '최도윤',
'phoneNumbers': ['462-4433-5968', '483-1709-4850']},
{'address': {'city': '서울', 'street': '683번지', 'zipCode': '75013'},
'age': 72,
'carOwnership': True,
'hobbies': ['여행', '사진 촬영', '독서'],
'isMarried': True,
'name': '최지훈',
'phoneNumbers': ['382-2779-3692', '835-4343-5346']},
{'address': {'city': '서울', 'street': '547번지', 'zipCode': '33986'},
'age': 77,
'carOwnership': False,
'hobbies': ['독서', '등산', '음악 감상'],
'isMarried': True,
'name': '정도윤',
'phoneNumbers': ['136-2831-1021', '818-9721-7208']},
{'address': {'city': '서울', 'street': '304번지', 'zipCode': '69380'},
'age': 44,
'carOwnership': True,
'hobbies': ['독서', '게임', '여행'],
'isMarried': False,
'name': '정하린',
'phoneNumbers': ['423-5001-2734', '256-4271-3750']},
{'address': {'city': '서울', 'street': '924번지', 'zipCode': '77191'},
'age': 26,
'carOwnership': True,
'hobbies': ['등산', '독서', '여행'],
'isMarried': False,
'name': '김예준',
'phoneNumbers': ['668-1157-6180', '815-9997-6459']},
{'address': {'city': '서울', 'street': '539번지', 'zipCode': '67491'},
'age': 48,
'carOwnership': True,
'hobbies': ['요리', '게임', '등산'],
'isMarried': True,
'name': '최수아',
'phoneNumbers': ['745-5529-4411', '437-3892-3668']},
{'address': {'city': '서울', 'street': '358번지', 'zipCode': '70195'},
'age': 41,
'carOwnership': True,
'hobbies': ['여행', '영화 감상', '요리'],
'isMarried': False,
'name': '이서연',
'phoneNumbers': ['914-2071-3446', '539-6835-4629']},
```

```
{'address': {'city': '서울', 'street': '741번지', 'zipCode': '82600'},
'age': 61,
'carOwnership': True,
'hobbies': ['영화 감상', '독서', '여행'],
'isMarried': True,
'name': '이지훈',
'phoneNumbers': ['709-3578-3445', '907-3295-1822']},
{'address': {'city': '서울', 'street': '932번지', 'zipCode': '57742'},
'age': 47,
'carOwnership': True,
'hobbies': ['사진 촬영', '등산', '요리'],
'isMarried': False,
'name': '최서연',
'phoneNumbers': ['508-9125-7029', '939-1920-5084']},
{'address': {'city': '서울', 'street': '603번지', 'zipCode': '30841'},
'age': 23,
'carOwnership': False,
'hobbies': ['등산', '독서', '여행'],
'isMarried': True,
'name': '이민준',
'phoneNumbers': ['891-2980-9497', '811-3249-9899']},
{'address': {'city': '서울', 'street': '464번지', 'zipCode': '91295'},
'age': 52,
'carOwnership': False,
'hobbies': ['게임', '음악 감상', '요리'],
'isMarried': False,
'name': '박하은',
'phoneNumbers': ['499-4872-5904', '140-3733-7715']},
{'address': {'city': '서울', 'street': '401번지', 'zipCode': '71129'},
'age': 61,
'carOwnership': True,
'hobbies': ['음악 감상', '요리', '여행'],
'isMarried': False,
'name': '정수아',
'phoneNumbers': ['672-6315-8675', '975-1259-1656']},
{'address': {'city': '서울', 'street': '356번지', 'zipCode': '40080'},
'age': 19,
'carOwnership': False,
'hobbies': ['등산', '영화 감상', '요리'],
'isMarried': False,
'name': '이지훈',
'phoneNumbers': ['853-1953-3723', '408-3476-1336']},
{'address': {'city': '서울', 'street': '940번지', 'zipCode': '60335'},
'age': 52,
'carOwnership': True,
'hobbies': ['요리', '사진 촬영', '등산'],
'isMarried': True,
'name': '최도윤',
'phoneNumbers': ['290-8270-9786', '483-1765-4028']},
{'address': {'city': '서울', 'street': '289번지', 'zipCode': '59793'},
'age': 18,
'carOwnership': True,
'hobbies': ['여행', '요리', '영화 감상'],
'isMarried': True,
'name': '최주원',
'phoneNumbers': ['460-4533-7245', '344-2344-7362']}]
```


코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

`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`
