Ch 06. Document Loader

상태	완료
최종 편집 일시	@2025년 12월 13일 오후 8:45

https://wikidocs.net/233775

> LangChain 의 문서 로더(Document Loader)를 사용해 **다양한 형식의 데이터 파일로부터 문서로 로드** 할 수 있습니다.
>
> 로드한 문서는 `Document` 객체로 로드되며 `page_content` 에는 내용이 `metadata` 에는 메타데이터를 포함합니다.
>
> 예를 들어 간단한 `pdf` 파일을 로드하거나, Word, CSV, Excel, JSON 등을 로드하기 위한 문서 로더가 각각 존재합니다.
>
> 문서 로더는 `load()` , `aload()` , `lazy_load()` 등 다양한 기능을 제공합니다.

01. 도큐먼트(Document) 의 구조

Document

LangChain 의 기본 문서 객체입니다.

속성

- `page_content` : 문서의 내용을 나타내는 String.
- `metadata` : 문서의 메타데이터를 나타내는 딕셔너리
    - `document.metadata[key] = value` 를 통해 커스터마이징이 가능하다

주요 Loader

- `PyPDFLoader`: PDF 파일
- `CSVLoader`: CSV 파일
- `UnstructuredHTMLLoader`: HTML 파일
- `JSONLoader`: JSON 파일
- `TextLoader`: 텍스트 파일
- `DirectoryLoader`: 디렉토리

> 디렉토리를 로드한다는게 무슨 말일까?
>
> - `DirectoryLoader`를 사용하여 디렉토리 내의 모든 문서를 로드할 수 있다.
>   - 정말 말 그대로 디렉토리를 로드함으로써 내부 파일을 컨트롤할 수 있게 되는 것 하다
> - `DirectoryLoader` 인스턴스를 생성할 때 문서가 있는 디렉토리의 경로와 해당 문서를 식별할 수 있는 glob 패턴을 지정합니다.
>   - glob 패턴이란?
>
>   `"include": ["next-env.d.ts", "**/*.ts", "**/*.tsx"],` 에서 처럼 여러 와일드카드 문자를 사용하여 다수의 파일 집합을 지정하는 방식
>
>   claude code와 같은 cli 에서 어시스트를 해주는 서비스에서 자주 볼 수 있다.

02. PDF

https://docs.langchain.com/oss/python/langchain/overview

아래 표기된 숫자는 등수를 나타냅니다. (The lower, the better)https://velog.io/@autorag/PDF-%ED%95%9C%EA%B8%80-%ED%85%8D%EC%8A%A4%ED%8A%B8-%EC%B6%94%EC%B6%9C-%EC%8B%A4%ED%97%98#%EC%B4%9D%ED%8F%89

	PDFMiner	PDFPlumber	PyPDFium2	PyMuPDF	PyPDF2
Medical	1	2	3	4	5
Law	3	1	1	3	5
Finance	1	2	2	4	5
Public	1	1	1	4	5
Sum	5	5	7	15	20

→ 그때 그때 로드할 문서의 종류에 따라 Loader를 선택할 수 있다.

03. 한글(HWP)

한글(HWP)은 한글과컴퓨터에서 개발한 워드프로세서로, 한국의 대표적인 문서 작성 프로그램입니다.

파일 확장자로 .hwp를 사용하며, 기업, 학교, 정부 기관 등에서 널리 활용되고 있습니다. 그렇기 때문에 대한민국 개발자라면 `.hwp` 문서를 처리해야 하는 경험을 해보았을 것입니다. (혹은 해볼 예정이다)

아쉽게도 LangChain 에는 아직 integration 이 되지 않아 직접 구현한 `HWPLoader` 를 사용해야 합니다.

→ PDF와 형식만 다르지 유사하게 로드하고 메타데이터를 구할 수 있다

---

04. CSV

https://docs.python.org/3/library/csv.html

파이썬의 `csv module` 을 기본적으로 참고하여 활용하기 때문에 인자를 통해 커스텀이 가능하다

```
# 컬럼정보:
# PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked

# CSV 파일 경로
loader = CSVLoader(
    file_path="./data/titanic.csv",
    csv_args={
        "delimiter": ",",  # 구분자
        "quotechar": '"',  # 인용 부호 문자
        "fieldnames": [
            "Passenger ID",
            "Survival (1: Survived, 0: Died)",
            "Passenger Class",
            "Name",
            "Sex",
            "Age",
            "Number of Siblings/Spouses Aboard",
            "Number of Parents/Children Aboard",
            "Ticket Number",
            "Fare",
            "Cabin",
            "Port of Embarkation",
        ],  # 필드 이름
    },
)

# 데이터 로드
docs = loader.load()

# 데이터 출력
print(docs[1].page_content)
```

- pandas 의 dataframe을 로드하는 `DataFrameLoader` 도 존재한다

05. Excel

CSV와 마찬가지로 html 파일 형식으로 출력할 수 있고, pandas로 읽은다음 해당 dataframe을 로드하여 정보를 분석할 수도 있다

---

06. Word

- 이는 `Word` 문서를 하류에서 사용할 수 있는 문서 형식으로 로드하는 방법을 다룹니다.

> 하류에서 사용할 수 있는 문서 형식이 무엇일까?
>
> - 처리/활용 측면에서 다른 프로그램이나 처리 과정에서 바로 사용할 수 있는 형식
> - 재가공, 분석, 변환 없이 바로 활용 가능한 구조
> - **하류(downstream)**
>
> : 앞 단계에서 처리된 결과를 **다음 단계에서 사용하는 과정**
>
> 예: 입력 → 처리 → 출력 → 다음 처리/분석 → 최종 활용

- 위키독스에서 로드되는 element의 개수와 실제로 로드함으로써 나오는 개수가 왜 다를까? (데이터는 같은 데이터를 사용한다)
  - GPT가 예상하는 차이점 유발 포인트
    - **loader가 "요소 단위(elements)"로 분할하는 규칙**
      - 문단/표/리스트/머리말 등 처리 방식에 따라 개수가 달라짐
    - **라이브러리/환경/버전 차이**
      - 요소 인식 알고리즘이 달라질 수 있음

---

07. PowerPoint

자료형만 다르고 별 차이 없음

---

08. 웹 문서(WebBaseLoader)

`WebBaseLoader` 는 웹 기반 문서를 로드하는 로더입니다.

`bs4` 라이브러리를 사용하여 웹 페이지를 파싱합니다.

- `bs4.SoupStrainer` 를 사용하여 파싱할 요소를 지정합니다.
- `bs_kwargs` 매개변수를 사용하여 `bs4.SoupStrainer` 의 추가적인 인수를 지정합니다.

https://reference.langchain.com/python/

→ 웹크롤링 작업을 수행할 때 유용하게 쓸 수 있을 것 같다 (이제 더이상 html구조를 한땀 한땀 분석할 수 없다

---

09. 텍스트(TextLoader)

여기서도 여러개의 파일을 로드할때 glob 패턴을 활용하여 접근할 수 있다 또한 encoding 형식을 자동으로 탐지해주는 것이 상당히 편해 보인다.

```
from langchain_community.document_loaders import DirectoryLoader

path = "data/"

text_loader_kwargs = {"autodetect_encoding": True}

loader = DirectoryLoader(
    path,
    glob="**/*.txt",
    loader_cls=TextLoader,
    silent_errors=True,
    loader_kwargs=text_loader_kwargs,
)
docs = loader.load()
```

10. JSON

https://docs.langchain.com/oss/python/langchain/overview

공식 문서를 참고해서 JSON 보기

- JSON도 txt 파일과 마찬가지로 인코딩에 민감하기 때문에 명시적으로 작성함으로써 인코딩 및 디코딩 오류를 회피해야한다

```
import json
from pathlib import Path
from pprint import pprint


file_path = "week5Data/people.json"

data = json.loads(Path(file_path).read_text(encoding="utf-8"))
pprint(data)
```

11. Arxiv

Arxiv

arXiv은 물리학, 수학, 컴퓨터 과학, 정량 생물학, 정량 금융, 통계, 전기공학 및 시스템 과학, 경제학 분야의 200만 편의 학술 논문을 위한 오픈 액세스 아카이브입니다. API 도큐먼트

Arxiv 문서 로더에 접근하려면 `arxiv`, `PyMuPDF` 및 `langchain-community` 통합 패키지를 설치해야 합니다.

`PyMuPDF` 는 arxiv.org 사이트에서 다운로드한 PDF 파일을 텍스트 형식으로 변환합니다

→ Arxiv로 로드 및 분석할 수 있다니 신기하다

- 논문의 전체 내용이 아닌 요약본을 출력하고자 한다면, `get_summaries_as_docs()` 함수를 호출하면 됩니다. → 요약도 가능하다

```
# 문서 요약 로딩
docs = loader.get_summaries_as_docs()

# 첫 번째 문서 접근
print(docs[0].page_content)
```

---

12. UpstageLayoutAnalysisLoader

`UpstageLayoutAnalysisLoader` 는 Upstage AI에서 제공하는 문서 분석 도구로, LangChain 프레임워크와 통합되어 사용할 수 있는 문서 로더입니다.

주요 특징: - PDF, 이미지 등 다양한 형식의 문서에서 레이아웃 분석 수행 - 문서의 구조적 요소(제목, 단락, 표, 이미지 등)를 자동으로 인식 및 추출 - OCR 기능 지원 (선택적)

UpstageLayoutAnalysisLoader는 단순한 텍스트 추출을 넘어 문서의 구조를 이해하고 요소 간 관계를 파악하여 보다 정확한 문서 분석을 가능하게 합니다.

→ UpStage API가 필요하다https://console.upstage.ai/docs/getting-started

UpstageLayoutAnalysisLoader

주요 파라미터

- `file_path` : 분석할 문서 경로
- `output_type` : 출력 형식 [(기본값)'html', 'text']
- `split` : 문서 분할 방식 ['none', 'element', 'page']
- `use_ocr=True` : OCR 사용
- `exclude=["header", "footer"]` : 헤더, 푸터 제외

---

13. LlamaParser

LlamaParse는 LlamaIndex에서 개발한 문서 파싱 서비스로, 대규모 언어 모델(LLM)을 위해 특별히 설계되었습니다.

- PDF, Word, PowerPoint, Excel 등 다양한 문서 형식 지원
- 자연어 지시를 통한 맞춤형 출력 형식 제공
- 복잡한 표와 이미지 추출 기능
- JSON 모드 지원

- 외국어 지원

LlamaParse는 독립형 API로 제공되며, LlamaCloud 플랫폼의 일부로도 사용 가능합니다. 이 서비스는 문서를 파싱하고 정제하여 검색 증강 생성(RAG) 등 LLM 기반 애플리케이션의 성능을 향상시키는 것을 목표로 합니다.

사용자는 무료로 하루 1,000페이지를 처리할 수 있으며, 유료 플랜을 통해 추가 용량을 확보할 수 있습니다. LlamaParse는 현재 공개 베타 버전으로 제공되고 있으며, 지속적으로 기능이 확장되고 있습니다.

> 이미지 추출을 지원한다는게 큰 장점인 것 같다

• 링크: https://cloud.llamaindex.ai

MultiModal Model 로 파싱

주요 파라미터

- `use_vendor_multimodal_model` : 멀티모달 모델 사용 여부를 지정합니다. `True`로 설정하면 외부 벤더의 멀티모달 모델을 사용합니다.
- `vendor_multimodal_model_name` : 사용할 멀티모달 모델의 이름을 지정합니다. 여기서는 "openai-gpt4o"를 사용하고 있습니다.
- `vendor_multimodal_api_key` : 멀티모달 모델 API 키를 지정합니다. 환경 변수에서 OpenAI API 키를 가져옵니다.
- `result_type` : 파싱 결과의 형식을 지정합니다. "markdown"으로 설정되어 있어 결과가 마크다운 형식으로 반환됩니다.
- `language` : 파싱할 문서의 언어를 지정합니다. "ko"로 설정되어 한국어로 처리됩니다.
- `skip_diagonal_text` : 대각선 텍스트를 건너뛸지 여부를 결정합니다.
- `page_separator` : 페이지 구분자를 지정할 수 있습니다.