CH06 - 문서 로더(Document Loader)

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

I. 도큐먼트(Document) 의 구조

Document

LangChain에서 사용하는 **기본 문서 객체**

- **page_content**: 문서의 실제 텍스트
- **metadata**: 출처, 페이지 번호 등 부가 정보

```
Document(page_content="텍스트", metadata={})
```

Document Loader

파일을 읽어 **Document 객체(List)**로 변환하는 도구

주요 Loader

- PyPDFLoader (PDF)
- CSVLoader (CSV)
- TextLoader (TXT)

- JSONLoader (JSON)
- DirectoryLoader (폴더)

PyPDFLoader 핵심 메서드

- **load()** → 페이지별 Document 리스트 반환
- **load_and_split()** → 로드 + 텍스트 분할
- **lazy_load()** → generator 방식 로드
- **aload()** → 비동기 로드

II. PDF

PDF란?

- ISO 32000 표준 문서 포맷
- 텍스트 + 이미지 포함, OS/환경 독립적

LangChain에서 PDF 처리

PDF → **Document 객체(List)** 로 변환

파서 종류에 따라 **속도, 정확도, 메타데이터, OCR 지원**이 다름

주요 PDF Loader 한눈 정리

- **PyPDFLoader**
  - 가장 기본
  - 페이지 단위 Document
  - 메타데이터: source, page
- **PyPDFLoader (OCR)**
  - 이미지 기반 PDF 대응
  - 스캔 문서에 유용
- **PyMuPDFLoader**

    - 속도 빠름
    - 메타데이터 매우 풍부 (작성자, 생성일 등)
- **PDFPlumberLoader**
    - PyMuPDF와 유사
    - 표/레이아웃 비교적 잘 유지
- **PDFMinerLoader**
    - 텍스트 추출 정확도 높음
    - HTML 변환 가능 → 구조 분석에 유리
- **UnstructuredPDFLoader**
    - 문단/제목/리스트 등 **구조 인식**
    - mode="elements" 사용 시 요소 단위 분리 가능
- **PyPDFium2Loader**
    - 가볍고 안정적
    - 기본적인 PDF 처리용

AutoRAG 실험 결과 요약

> PDFMiner / PDFPlumber 성능 우수
>
> (Medical, Finance, Public 문서에서 특히 강함)

핵심 한 줄

> PDF Loader는 문서 특성(OCR, 구조, 속도)에 맞게 선택해야 한다.

III. 한글(HWP)

HWP란?

- 한글과컴퓨터에서 개발한 문서 포맷
- 확장자: **.hwp**
- 국내 기업·학교·정부에서 가장 많이 사용

LangChain에서 HWP 처리

- **공식 LangChain Loader 없음**
- → **커스텀 HWPLoader 사용**

HWPLoader 사용

- `langchain-teddynote` 패키지 제공

```
from langchain_teddynote.document_loadersimport HWPLoader

loader = HWPLoader("./data/디지털 정부혁신 추진계획.hwp")
docs = loader.load()
```

- 결과: **List[Document]**
- `page_content`에 텍스트 추출

핵심 한 줄

> HWP는 LangChain에 기본 지원이 없어커스텀 HWPLoader로 처리해야 한다.

IV. CSV

CSV란?

- 쉼표(`,`)로 값을 구분하는 텍스트 파일
- **한 줄 = 하나의 데이터 레코드**

CSVLoader

- **CSV 한 행 = 하나의 Document**
- metadata에 **row 번호** 포함

```
from langchain_community.document_loadersimport CSVLoader

loader = CSVLoader("./data/titanic.csv")
docs = loader.load()
```

```
metadata: {'source': './data/titanic.csv', 'row': 0}
```

CSV 로딩 커스터마이징

- delimiter, quotechar, fieldnames 설정 가능

```
CSVLoader(file_path, csv_args={...})
```

---

source_column

- 특정 컬럼 값을 **문서 출처(source)** 로 사용

```
CSVLoader(file_path, source_column="PassengerId")
```

→ QA 체인에서 **행 단위 출처 추적**에 유용

UnstructuredCSVLoader

- CSV를 **테이블 구조**로 로드
- `mode="elements"` 사용 시 **HTML 테이블 제공**

DataFrameLoader

- Pandas DataFrame → Document 변환
- 특정 컬럼을 page_content로 지정

```
DataFrameLoader(df, page_content_column="Name")
```

- 나머지 컬럼은 metadata로 저장
- `lazy_load()` 지원 (대용량 처리에 유리)

핵심 한 줄

> CSV는 보통 한 행씩 Document로 변환하며,source_column과 DataFrameLoader가 실무에서 특히 유용하다.

V. Excel

Excel이란?

- Microsoft Excel 파일
- 확장자: **.xlsx, .xls**
- 표 형태 데이터 저장에 사용

UnstructuredExcelLoader

- Excel 파일을 **Document 객체로 로드**
- 기본적으로 **1개의 Document**로 로드됨

```
from langchain_community.document_loadersimport UnstructuredExcelLoa
der

loader = UnstructuredExcelLoader("titanic.xlsx", mode="elements")
docs = loader.load()
```

- **page_content**: 셀의 원시 텍스트
- **metadata["text_as_html"]**: 표를 HTML 형태로 제공

DataFrameLoader (권장)

- Excel → Pandas DataFrame → Document
- **행 단위 처리** 가능

```
import pandasas pd
df = pd.read_excel("titanic.xlsx")

from langchain_community.document_loadersimport DataFrameLoader
loader = DataFrameLoader(df, page_content_column="Name")
docs = loader.load()
```

- page_content: 지정한 컬럼
- metadata: 나머지 컬럼

핵심 한 줄

> Excel은 DataFrameLoader가 가장 실무적이며,표 구조가 필요하면
> UnstructuredExcelLoader를 사용한다.

VI. Word

Microsoft Word란?

- Microsoft의 워드 프로세서
- 확장자: **.docx**

Docx2txtLoader

- `.docx` 파일을 **단순 텍스트**로 로드
- 결과: **1개의 Document**

```
from langchain_community.document_loadersimport Docx2txtLoader

loader = Docx2txtLoader("sample-word-document.docx")
docs = loader.load()
```

UnstructuredWordDocumentLoader

- 문서의 **구조(제목, 문단 등)** 인식
- 기본: **1개의 Document**

```
from langchain_community.document_loadersimport UnstructuredWordDo
cumentLoader
loader = UnstructuredWordDocumentLoader("sample-word-document.doc
x")
docs = loader.load()
```

---

mode="elements"

- 제목, 문단 등 **요소 단위로 분리**
- 여러 개의 Document 생성

```
UnstructuredWordDocumentLoader(file_path, mode="elements")
```

핵심 한 줄

> Word 문서는 간단하면 Docx2txt,구조가 필요하면
> UnstructuredWordDocumentLoader를 사용한다.

VII. PowerPoint

Microsoft PowerPoint란?

- Microsoft의 프레젠테이션 도구
- 확장자: **.pptx**

UnstructuredPowerPointLoader

- PowerPoint 파일을 **Document 객체로 로드**
- 기본: **1개의 Document**

```
from langchain_community.document_loadersimport UnstructuredPowerPo
intLoader

loader = UnstructuredPowerPointLoader("sample-ppt.pptx")
docs = loader.load()
```

---

mode="elements"

- 슬라이드의 **제목, 텍스트 요소 단위 분리**
- 여러 개의 Document 생성

```
UnstructuredPowerPointLoader(file_path, mode="elements")
```

---

metadata 예시

- source, filename, page_number, category 등 포함

핵심 한 줄

> PowerPoint는 UnstructuredPowerPointLoader를 사용하며,요소 단위 분석이 필요하면 mode="elements"를 사용한다.

VIII. 웹 문서(WebBaseLoader)

WebBaseLoader란?

- **웹 페이지(URL)** 를 불러와 Document로 변환
- 내부적으로 **BeautifulSoup(bs4)** 사용

기본 사용

```
from langchain_community.document_loadersimport WebBaseLoader

loader = WebBaseLoader(web_paths=[url])
docs = loader.load()
```

- 결과: **URL 하나당 1 Document**
- metadata에 **source(URL)** 포함

---

bs4.SoupStrainer

- **필요한 HTML 요소만 선택적으로 파싱**
- 기사 본문, 제목 등만 추출할 때 사용

```
bs_kwargs = {
"parse_only": bs4.SoupStrainer("div", attrs={...})
}
```

옵션들

- **User-Agent 설정**: 봇 차단 회피
- **verify=False**: SSL 인증 오류 우회
- **proxies**: IP 차단 우회 가능

여러 URL 로드

- URL 리스트 전달 → **순서대로 Document 리스트 반환**

```
WebBaseLoader(web_paths=[url1, url2])
```

비동기 로드

- `aload()` 지원
- `requests_per_second`로 요청 속도 제한 조절

핵심 한 줄

> WebBaseLoader는 웹 문서를 가져와HTML 요소 단위로 필요한 내용만 추출할 수 있다.

IX. 텍스트(TextLoader)

TXT란?

- `.txt` 확장자의 **순수 텍스트 파일**
- 가장 단순한 문서 형식

TextLoader

- 텍스트 파일을 **1개의 Document**로 로드

```
from langchain_community.document_loadersimport TextLoader

loader = TextLoader("data/file.txt")
docs = loader.load()
```

- `page_content` : 파일 전체 텍스트
- `metadata` : source(파일 경로)

여러 TXT 파일 로드 (DirectoryLoader)

- 디렉토리 내 TXT 파일들을 한 번에 로드

```
from langchain_community.document_loadersimport DirectoryLoader

loader = DirectoryLoader(
    path="data/",
    glob="**/*.txt",
    loader_cls=TextLoader,
    silent_errors=True,
    loader_kwargs={"autodetect_encoding":True},
)
docs = loader.load()
```

인코딩 관련 옵션

- **autodetect_encoding**

  → UTF-8, EUC-KR, CP949 등 자동 감지

- **silent_errors**

  → 인코딩 오류 파일은 건너뛰고 계속 로드

핵심 한 줄

> TXT는 TextLoader로 간단히 로드하며,여러 파일·인코딩 처리에는 DirectoryLoader가 유용하다.

X. JSON

JSON이란?

- `.json` 확장자의 **구조화된 데이터 형식**
- key–value, 리스트, 중첩 구조 표현에 적합

JSONLoader

- JSON 파일에서 **원하는 필드만 추출**하여 Document로 변환
- 내부적으로 **jq 문법** 사용 → 선택적 파싱 가능

```
from langchain_community.document_loadersimport JSONLoader

loader = JSONLoader(
    file_path="data/people.json",
    jq_schema=".[].phoneNumbers",
    text_content=False,
)
docs = loader.load()
```

특징

- **jq_schema**: 추출할 JSON 경로 지정
- **JSON 항목 하나 = 하나의 Document**
- metadata에 `source`, `seq_num` 포함

핵심 한 줄

> JSONLoader는 jq 문법으로JSON에서 필요한 데이터만 골라 Document로 만들 수 있다.

XI. Arxiv

arXiv이란?

- 물리학, 컴퓨터과학, 수학 등 **학술 논문 오픈 아카이브**
- 200만 편 이상의 논문 제공

ArxivLoader

- 키워드로 논문 검색 후 **논문 PDF + 메타데이터 로드**
- 내부적으로 **arxiv API + PyMuPDF** 사용

```
from langchain_community.document_loadersimport ArxivLoader

loader = ArxivLoader(
    query="Chain of thought",
    load_max_docs=2,
    load_all_available_meta=True,
)
docs = loader.load()
```

주요 옵션

- **query**: 검색 키워드
- **load_max_docs**: 최대 로드 논문 수
- **load_all_available_meta**
  - `True`: 전체 메타데이터
  - `False`: 핵심 메타데이터만

메타데이터 예시

- Title, Authors, Published
- Summary (초록)
- entry_id, categories, links(PDF/HTML)

요약만 로드

- 논문 **초록만 Document로 생성**

```
docs = loader.get_summaries_as_docs()
```

---

lazy_load()

- 논문을 **하나씩 지연 로드**

- 대량 논문 처리 시 메모리 절약

핵심 한 줄

> ArxivLoader는 키워드 기반으로논문과 메타데이터를 한 번에 가져올 수 있다.

XII. UpstageLayoutAnalysisLoader

UpstageLayoutAnalysisLoader란?

- **Upstage AI**에서 제공하는 고급 문서 분석 로더
- PDF / 이미지 문서의 **레이아웃 구조**를 이해해 로드

주요 특징

- 제목, 단락, 표, 이미지 등 **구조 요소 자동 인식**
- **OCR 지원** (스캔 문서 가능)
- 단순 텍스트 추출보다 **정확한 문서 이해** 가능

사용 준비

- 패키지 설치: `langchain-upstage`
- `.env`에 **UPSTAGE_API_KEY** 설정

기본 사용

```
from langchain_upstageimport UpstageLayoutAnalysisLoader

loader = UpstageLayoutAnalysisLoader(
    file_path,
    output_type="text",
    split="page",
    use_ocr=True,
```

```
    exclude=["header","footer"],
)
docs = loader.load()
```

주요 파라미터

- **output_type**: `"text"` | `"html"`
- **split**: `"none"` | `"element"` | `"page"`
- **use_ocr**: OCR 사용 여부
- **exclude**: header, footer 제거

핵심 한 줄

> UpstageLayoutAnalysisLoader는OCR + 레이아웃 분석으로 문서 구조까지 이해하는 고급 로더다.

XIII. LlamaParser

LlamaParse란?

- **LlamaIndex**에서 제공하는 문서 파싱 서비스
- LLM(RAG)에 최적화된 **고급 문서 파서**
- PDF, Word, PPT, Excel 등 다양한 형식 지원

주요 특징

- 표·이미지·복잡한 레이아웃 **정확 추출**
- **자연어 지시(parsing_instruction)** 지원
- Markdown / Text / JSON 출력
- 다국어 지원

- 하루 **무료 1,000페이지** 처리 가능

사용 준비

- 패키지 설치: `llama-parse`, `llama-index-*`
- `.env`에 **LLAMA_CLOUD_API_KEY** 설정

기본 사용

```
from llama_parseimport LlamaParse

parser = LlamaParse(result_type="markdown", language="ko")
```

- `SimpleDirectoryReader`와 함께 사용
- 결과는 **LlamaIndex Document**

LangChain 변환

```
docs = [doc.to_langchain_format()for docin documents]
```

멀티모달 파싱

- GPT-4o 등 **멀티모달 모델 사용 가능**
- 표·이미지 포함 문서에 특히 강력

사용자 정의 지시

```
"표를 markdown 형식으로 추출해줘"
```

→ 원하는 형식으로 결과 제어 가능

핵심 한 줄

> LlamaParse는LLM용 문서 파싱 끝판왕(표·레이아웃·지시 기반 파싱까지 지원)이다.