CH06 Document Loader

Date	@2025년 12월 20일
Category	GDGoC
Study	LangChain 스터디
day of the week	Sat
form	스터디
done	☑

왜 Document Loader가 중요한가

LangChain에서 Document Loader는 단순히 파일을 읽는 유틸리티가 아니다.

이는 **외부 데이터(문서, 웹, 테이블, API 응답 등)를 LLM 파이프라인으로 유입시키는 최초의 관문(Ingestion Layer)**이며, 이 단계에서의 설계 선택은 이후 모든 단계(텍스트 분할, 임베딩, 검색, 생성)의 품질을 결정한다.

실제로 RAG 시스템에서 발생하는 많은 오류는

- 임베딩 모델의 문제가 아니라
- Retriever의 문제가 아니라
- **초기 문서 로딩과 구조 보존 실패**에서 시작된다.

따라서 Document Loader는 "전처리 단계"가 아니라 **지식 표현의 첫 번째 설계 단계**로 이해해야 한다.

1. Document 객체의 기본 구조와 철학

LangChain의 모든 Loader는 공통적으로 `Document` 객체를 반환한다.

이 객체는 다음 두 요소로 구성된다.

- `page_content` : LLM이 직접 처리할 텍스트 본문
- `metadata` : 해당 텍스트가 어떤 맥락에서 왔는지를 설명하는 보조 정보

중요한 점은, Document는 단순한 문자열 컨테이너가 아니라 **"내용(content)과 출처(context)를 결합한 최소 단위"**라는 것이다.

메타데이터에는 파일 경로, 페이지 번호, 작성자, 생성일, 카테고리, URL 등이 포함될 수 있으며, 이 정보는 이후 다음과 같은 역할을 한다.

- 검색 결과 필터링 (예: 특정 문서만 검색)
- 답변 출처 표시 (Grounding)
- 중요도 가중치 부여
- 개인정보·보안 규칙 적용

즉, metadata는 RAG 시스템에서 **신뢰성과 투명성을 담당하는 핵심 요소**다.

2. Document Loader의 동작 방식: 구조적 분류

LangChain의 다양한 Loader들은 겉보기엔 많아 보이지만, 실제로는 세 가지 축을 기준으로 체계적으로 분류할 수 있다.

2.1 로딩 방식 기준

첫 번째는 **문서를 언제, 어떻게 메모리에 올리는가**이다.

- `load()`

  모든 문서를 한 번에 로드한다.

  소규모 데이터나 실습 환경에 적합하지만, 대용량 문서에는 비효율적이다.

- `lazy_load()`

  Generator 기반으로 문서를 하나씩 순차 처리한다.

  대규모 PDF, 논문 코퍼스, 로그 파일 등에 필수적이다.

- `aload()`

  비동기 방식으로 문서를 로드한다.

  웹 크롤링이나 외부 API 호출과 결합할 때 효과적이다.

이 선택은 단순 성능 문제가 아니라 **시스템 안정성과 확장성 문제**다.

2.2 분할 전략 기준

두 번째 축은 **문서를 어떤 단위로 나누는가**이다.

- 페이지 단위

  PDF와 같이 페이지 개념이 중요한 문서에 적합하다.

- 요소(element) 단위

  제목, 문단, 리스트, 표 등 문서의 구조적 요소를 유지한다.

  검색 정확도와 의미 보존에 매우 유리하다.

- 분할 없음

  문서 전체를 하나의 Document로 유지한다.

  요약(Summarization)에는 유리하지만, 검색에는 불리하다.

분할 전략은 이후 TextSplitter와도 강하게 연결되며, **"어디서 의미를 끊을 것인가"에 대한 설계 결정**이다.

2.3 파싱 깊이 기준

세 번째 축은 **문서를 얼마나 '이해'하려 하는가**이다.

- 단순 텍스트 추출

  빠르지만 레이아웃, 의미 구조가 손실된다.

- 레이아웃 기반 파싱

  제목/본문/표/리스트를 구분한다.

- OCR 및 멀티모달 파싱

  스캔 문서, 이미지 기반 문서까지 처리한다.

최근 Loader들은 단순 추출을 넘어 **문서 구조 복원(Document Understanding)** 방향으로 진화하고 있다.

3. PDF Loader 계열: 가장 중요한 로더 군

PDF는 가장 흔하지만 가장 까다로운 문서 형식이다.

텍스트가 시각적 좌표에 기반해 저장되기 때문에, "읽는 순서"와 "의미 구조"가 파일에 명시되어 있지 않은 경우가 많다.

그래서 LangChain에는 여러 PDF Loader가 존재하며, 이는 중복이 아니라 **트레이드오프의 결과**다.

- PyPDFLoader

  가장 기본적인 선택지로, 단순하고 안정적이다.

- PyMuPDFLoader

  페이지 수, 생성일, 작성 도구 등 풍부한 메타데이터를 제공한다.

- PDFPlumberLoader

  표(table) 추출에 상대적으로 강점이 있다.

- UnstructuredPDFLoader

  문단, 제목, 리스트 등 요소 단위로 문서를 분해한다.

- PDFMiner 계열

  HTML로 변환하여 글꼴 크기, 위치 정보를 활용할 수 있다.

이 단계에서의 선택은 **"텍스트를 쓸 것인가, 구조를 살릴 것인가"**라는 질문에 대한 답이다.

4. 구조화 데이터 Loader: CSV · Excel · JSON

CSV, Excel, JSON은 본질적으로 구조화된 데이터다.

이 Loader들의 핵심 역할은 **"행(row) 혹은 객체(object)를 자연어 Document로 변환하는 것"**이다.

CSV / Excel

- 기본 CSVLoader는 한 행을 하나의 Document로 변환한다.
- UnstructuredCSVLoader는 HTML 테이블 표현을 함께 제공한다.
- DataFrameLoader는 Pandas와 결합되어 분석 친화적이다.

여기서 중요한 설계 포인트는 다음과 같다.

- LLM에게 "데이터를 읽게 할 것인가"
- 아니면 "데이터에 대해 질문하게 할 것인가"

이 선택에 따라 Loader와 이후 파이프라인이 완전히 달라진다.

JSONLoader

JSONLoader는 단순 로더라기보다는 **데이터 선택기(Data Selector)**에 가깝다.

`jq_schema`를 통해 필요한 필드만 추출할 수 있으며, API 응답, 로그 데이터, 설정 파일을 RAG로 연결할 때 매우 유용하다.

5. 문서 포맷 Loader: HWP · Word · PowerPoint

실무 환경에서는 PDF보다 HWP, Word, PowerPoint 문서가 더 중요한 경우도 많다.

- HWP

  공식 LangChain 지원이 없어 커스텀 Loader(HWPLoader)를 사용한다.

  한국 환경에서는 매우 현실적인 문제다.

- Word / PowerPoint

  대부분 Unstructured 기반으로 처리되며,

  mode="elements" 옵션을 통해 문서 구조를 유지할 수 있다.

이 계열의 공통 특징은 **"사람이 작성한 문서의 논리 구조를 최대한 보존하려는 시도"**다.

6. 웹 및 외부 지식 Loader

WebBaseLoader는 웹 페이지를 Document로 변환한다.

이때 핵심은 전체 페이지가 아니라 **의미 있는 부분만 추출하는 것**이다.

- bs4.SoupStrainer로 DOM 요소를 제한
- User-Agent, 요청 빈도, SSL 설정 고려

웹 로딩은 기술 문제뿐 아니라 윤리적·법적 고려까지 포함한 설계 영역이다.

7. 연구·학술 Loader: Arxiv

ArxivLoader는 단순 PDF 로더가 아니라 논문의 메타데이터(저자, 초록, 출판일, 카테고리)를 함께 제공한다.

특히 `get_summaries_as_docs()` 는 "논문 전체"가 아니라 "초록 중심 RAG"를 가능하게 하여 연구 보조 시스템에서 매우 강력하다.

8. 고급 문서 이해 계층: Layout & LLM 기반 파서

여기서부터 Document Loader는 단순한 전처리를 넘어 **의미 해석 단계**로 진입한다.

UpstageLayoutAnalysisLoader

- 제목, 단락, 표, OCR을 통합적으로 인식
- 정책 문서, 보고서, 공공 데이터에 적합
- 문서를 "레이아웃 단위 지식"으로 변환

LlamaParse

- LLM을 사용해 문서를 Markdown/JSON으로 재구성
- 표, 이미지, 섹션 의미 복원
- 사용자 정의 파싱 지시 가능

이는 중요한 전환점을 의미한다.

Loader가 더 이상 규칙 기반이 아니라, 추론 기반으로 이동하고 있다는 점이다.

9. 전체 구조 요약

LangChain의 Document Loader 계층은 다음과 같은 역할을 수행한다.

1. 외부 데이터를 LLM 친화적 형식으로 변환한다.
2. 문서의 구조와 맥락을 가능한 한 보존한다.
3. 이후 검색·생성 단계의 품질을 결정한다.
4. 단순 로딩에서 문서 이해(Document Understanding)로 진화 중이다.

결론

LangChain의 Document Loader는 "파일을 읽는 코드 모음"이 아니라 **LLM 시스템의 지식 표현 방식을 설계하는 핵심 계층**이다.

Loader 선택은 단순 편의가 아니라

- 검색 정확도
- 답변 신뢰성
- 시스템 확장성
- 유지보수 비용

까지 좌우하는 구조적 결정이다.