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

---

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

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

디렉토리에서 문서를 로드하는 방법

`DirectoryLoader`

- `LangChain`의 **`DirectoryLoader`** = `디스크`에서 파일을 읽어 `LangChain Document` 객체로 변환하는 기능을 구현
  - 와일드카드 패턴을 포함하여 `파일 시스템`에서 `로드`하는 방법
  - `파일 I/O`에 `멀티스레딩`을 사용하는 방법
  - 특정 파일 유형(예: 코드)을 파싱하기 위해 사용자 정의 로더 클래스를 사용하는 방법
  - 디코딩으로 인한 오류와 같은 오류를 처리하는 방법

- **`DirectoryLoader`** = 기본적으로 **`UnstructuredLoader`**를 **`loader_cls`** kwarg로 사용
  - **`Unstructured`**
    - `PDF`와 `HTML`과 같은 다양한 형식의 파싱을 지원
    - `마크다운(.md)` 파일을 읽기 위해 사용해보기

```
from langchain_community.document_loaders import DirectoryLoader

# 디렉토리 로더 초기화
loader = DirectoryLoader(
    "../docs/",
    glob="**/*.md")

# 문서 로드
docs = loader.load()

# 문서 개수 계산
len(docs)
```

- 셀 출력

```
50
```

옵션_1: `show_progress=True`

- **`진행 상태 표시줄`** 표시: **`show_progress=True`**

```
# 디렉토리 로더 설정
loader = DirectoryLoader(
        "../",                          # 경로 설정
        glob="**/*.md",                 # 원하는 파일 형식 설정
        show_progress=True              # 옵션 설정
```

```
            )
```

- 사전에 `VS Code` 터미널에 설치할 것

```
    pip install tqdm
```

```
from langchain_community.document_loaders import DirectoryLoader

# 디렉토리 로더 초기화
loader = DirectoryLoader(
    "../docs/",
    glob="**/*.md",
    show_progress=True,
    )

# 문서 로드
docs = loader.load()

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")

# 첫 번째 문서의 전체 내용 출력
# 슬라이싱 [:100] 부분을 제거하여 파일 전체 내용 출력해보기
if docs:
    print("\n첫 번째 문서의 내용:\n")
    print(docs[10].page_content)
else:
    print("\n조건에 맞는 문서를 찾지 못했습니다.")
```

- 셀 출력 (3.3s)

```
0%|          | 0/50 [00:00<?, ?it/s]
4%|▍         | 2/50 [00:00<00:02, 18.45it/s]
10%|▉         | 5/50 [00:00<00:03, 13.11it/s]
14%|█▍        | 7/50 [00:00<00:03, 11.41it/s]
18%|█▊        | 9/50 [00:00<00:03, 11.22it/s]
24%|██▍       | 12/50 [00:00<00:02, 14.50it/s]
28%|██▊       | 14/50 [00:01<00:02, 14.84it/s]
32%|███▏      | 16/50 [00:01<00:03, 10.95it/s]
40%|████      | 20/50 [00:01<00:02, 13.72it/s]
44%|████▍     | 22/50 [00:01<00:02, 10.66it/s]
58%|█████▊    | 29/50 [00:02<00:01, 16.20it/s]
66%|██████▌   | 33/50 [00:02<00:00, 19.07it/s]
72%|███████▏  | 36/50 [00:02<00:00, 17.41it/s]
76%|███████▌  | 38/50 [00:02<00:00, 14.90it/s]
82%|████████▏ | 41/50 [00:02<00:00, 16.34it/s]
86%|████████▌ | 43/50 [00:02<00:00, 16.88it/s]
90%|█████████ | 45/50 [00:03<00:00, 15.15it/s]
96%|█████████▌| 48/50 [00:03<00:00, 17.74it/s]
100%|██████████| 50/50 [00:03<00:00, 15.15it/s]

로드된 문서 개수: 50
```

- 전체 내용 출력

```
첫 번째 문서의 내용:

🛠️ OutputParser / Datetime Troubleshooting 기록

작성일: 2025-08-25 작성자: Jay

1. gemini-2.5 모델의 Datetime Parser 처리 이슈

현재 날짜 출력 묻는 프롬프트에 무조건 2024-05-15 으로 답하는 현상

1) 문제: LLM이 현재 시간 요청 시 고정된 날짜 (2024-05-15) 반환
```

output2의 첫 프롬프트: 대한민국의 현재 날짜는?

구체적인 장소를 제공하여 현재 날짜를 출력할 것으로 기대

but → 출력값: 2024-05-15

두번째 시도: 현재 날짜는? or 오늘 날짜는?

현재, 오늘 이라는 단어를 통해 현재 날짜를 알 수 있을 것으로 기대

but → 출력값: 2024-05-15

세번째 시도: LLM이 학습된 데이터의 일부가 특정 기준 날짜를 가지고 있을 수 있을 가능성

즉, 2024-05-25가 기본 설정값 으로 설정된 경우일 가능성

기본값을 무시하도록 프롬프트 수정

기본 날짜 설정을 무시하고, 오늘 날짜를 알려주세요.

오늘 날짜를 시스템 시간에 맞게 출력해주세요.

여전히 2024-05-15로 출력

2) 원인: Gemini 모델의 실시간 시스템 시간 접근 제한

gemini-2.5-flash-lite = 모델 자체에 실시간 시스템 시간에 접근할 수 있는 제한이 없을 것으로 추측

추측 이유

이유_1: 모델의 시간 처리 방식

Gemini-2.5-Flash-light 모델 = 학습된 데이터에 기반하여 응답을 생성

실시간 날짜 정보를 사용할 수 없거나 시스템 시간에 접근할 수 없을 수 있음

이유_2: 프롬프트 자체에 시스템 시간 요청이 불가능

LLM이 훈련된 데이터와 규칙을 바탕으로 작동

따라서 실시간 시스템 시간을 모델에 직접 반영하는 것은 일반적인 사용 방식에서는 불가능할 수 있음

3) 해결: Python datetime.now()로 시스템 시간을 프롬프트에 주입

해결방법

해결방법_1: Python 코드를 사용하여 시스템 날짜를 가져오기

Python 코드로 시스템 날짜를 동적으로 가져오고, 그 값을 프롬프트에 넣어서 모델을 호출하기

성공!

```python # 시스템 시간을 코드에서 가져와 프롬프트에 전달하기 import datetime

current_date = datetime.datetime.now().strftime("%Y-5m-%d")

# 프롬프트로 만들기
df_datetime = f"오늘 날짜는 {current_date}입니다."

# 체인 실행하기
output3 = chain_datetime.invoke({"question":df_datetime})

# 결과 출력해보기 '
print(output3.strftime("%Y-%m-%d"))                          # 2025-05-25 (0.7s)

```

해결방법_2: 실시간 시간에 접근 가능한 LLM모델 사용하기

ChatGPT의 플러그인 및 API 기능을 통해 시간 관련 작업 처리 가능

Google PaLM

Google Cloud의 AI API

PaLM 모델 = 실시간 API 호출, 외부 데이터 소스에 통합해 시스템 시간 사용하는 APP 구축 가능

Claude: 시간, 날짜에 대한 문제를 해결할 수 있는 API 제공

4) 교훈: LLM의 한계를 이해하고 외부 도구로 보완하는 하이브리드 접근의 필요성을 알게 됨

```
# md파일의 일부만 출력해보기
from langchain_community.document_loaders import DirectoryLoader

# 디렉토리 로더 초기화
loader = DirectoryLoader(
    "../docs/",
    glob="**/*.md",
    show_progress=True,
    )

# 문서 로드
docs = loader.load()

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")

# 첫 번째 문서의 전체 내용 출력
# 슬라이싱 [:100] 부분을 제거하여 파일 전체 내용 출력해보기
if docs:
    print("\n첫 번째 문서의 내용:\n")
    print(docs[10].page_content[:100])
else:
    print("\n조건에 맞는 문서를 찾지 못했습니다.")
```

- 셀 출력 (3.3s)

```
0%|          | 0/50 [00:00<?, ?it/s]
4%|▍         | 2/50 [00:00<00:02, 18.45it/s]
10%|▉         | 5/50 [00:00<00:03, 13.11it/s]
14%|█▍        | 7/50 [00:00<00:03, 11.41it/s]
18%|█▊        | 9/50 [00:00<00:03, 11.22it/s]
24%|██▍       | 12/50 [00:00<00:02, 14.50it/s]
28%|██▊       | 14/50 [00:01<00:02, 14.84it/s]
32%|███▏      | 16/50 [00:01<00:03, 10.95it/s]
40%|████      | 20/50 [00:01<00:02, 13.72it/s]
44%|████▍     | 22/50 [00:01<00:02, 10.66it/s]
58%|█████▊    | 29/50 [00:02<00:01, 16.20it/s]
66%|██████▌   | 33/50 [00:02<00:00, 19.07it/s]
72%|███████▏  | 36/50 [00:02<00:00, 17.41it/s]
76%|███████▌  | 38/50 [00:02<00:00, 14.90it/s]
82%|████████▏ | 41/50 [00:02<00:00, 16.34it/s]
86%|████████▌ | 43/50 [00:02<00:00, 16.88it/s]
90%|█████████ | 45/50 [00:03<00:00, 15.15it/s]
96%|█████████▌| 48/50 [00:03<00:00, 17.74it/s]
100%|██████████| 50/50 [00:03<00:00, 15.15it/s]

로드된 문서 개수: 50
```

- 일부 출력: `[:100]`

```
첫 번째 문서의 내용:

🛠️ OutputParser / Datetime Troubleshooting 기록

작성일: 2025-08-25 작성자: Jay

1. gemini-2.5 모델의 Datetime
```

옵션_2: `use_multithreading=True`

- **진행 상태 표시줄** 표시: `use_multithreding=True`

```
    # 디렉토리 로더 설정
    loader = DirectoryLoader(
            "../",                          # 경로 설정
            glob="**/*.md",                 # 원하는 파일 형식 설정
            use_multithreding=True          # 옵션 설정
            )
```

```
from langchain_community.document_loaders import DirectoryLoader

# 디렉토리 로더 초기화
loader = DirectoryLoader(
    "../docs/",
    glob="**/*.md",
    use_multithreading=True,            # 멀티스레드 사용하기
    )

# 문서 로드
docs = loader.load()

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")

# 내용 출력해보기
if docs:
    print(f"\n{docs[11]}번째 문서의 내용:\n")
    print(docs[11].page_content[:100])
    print(f"\n{docs[12]}번째 문서의 내용:\n")
    print(docs[12].page_content[:100])
else:
    print("\n조건에 맞는 문서를 찾지 못했습니다.")
```

- 셀 출력

```
로드된 문서 개수: 50

page_content='🛠️ [문제 이름 또는 현상 요약]

❗ 문제 설명

무엇이 예상과 다르게 작동했는가?

발생한 에러 메시지, 로그, 증상 등

[터미널 로그 또는 에러 메시지 예시]


🔍 원인 분석/추정

가능한 원인들 (환경 문제, 버전 충돌 등)

🧪 시도한 해결 방법

첫 번째 시도(실패/성공 여부 표기)
```

두 번째 시도

추가 시도...

✅ 최종 해결 방법

실제로 문제를 해결한 조치

해결 경위 및 참고사항

💡 추가 참고 사항/팁

향후 재발 방지를 위한 교훈, 체크리스트

실습 환경/OS/패키지 버전 등

이 문서는 트러블슈팅 작성 참고용 템플릿입니다. 복사하여 새 이슈 작성 시 활용하세요.' metadata={'source': '../docs/troubleshooting/_te

🛠️ [문제 이름 또는 현상 요약]

❗ 문제 설명

무엇이 예상과 다르게 작동했는가?

발생한 에러 메시지, 로그, 증상 등

[터미널 로그 또는 에러 메시지 예시]

🔍 원

page_content='🛠️ LangChain 프로젝트 트러블슈팅 가이드

작성일: 2025-08-98 작성자: Jay

pyenv 가상환경 관련 이슈

문제: ModuleNotFoundError가 계속 발생

pip install claude-api-py 성공했지만 from claude_api import Client 실패

ModuleNotFoundError: No module named 'claude_api' 지속 발생

증상: pip으로 설치 → import에서 실패

원인: python 명령어가 가상환경 Python을 참조하지 않음

해결책: 1. pyenv which python 확인 2. PYENV_VERSION 환경변수와 .python-version 파일 일치 확인 3. pyenv rehash 실행

문제: 환경변수 충돌

원인: PYENV_VERSION과 .python-version 파일이 다른 환경 지정

해결책: unset PYENV_VERSION 후 로컬 설정 활용 - unset PYENV_VERSION - 환경변수 충돌 해결 - pip cache purge - 손상된 캐시 제거

패키지 설치 관련 이슈

문제: claude-api-py 설치 후 import 실패

증상: Successfully installed 메시지는 나오지만 실제 사용 불가

해결책: unofficial-claude-api 또는 공식 anthropic SDK 사용 - pip uninstall claude-api-py - 문제 패키지 제거 - pip install

교훈

pyenv 환경에서는 환경변수와 로컬 설정 일치 확인 필수

패키지 설치 성공 != 실제 사용 가능

비공식 패키지보다 공식 SDK 사용 권장

참고

날짜: 2025-08-09

환경: macOS, pyenv, Python 3.13.5, 해당_env 가상환경

소요시간: 약 1시간 (피할 수 있었던 삽질...)' metadata={'source': '../docs/troubleshooting/pyenv_troubleshooting_20250809.

🛠️ LangChain 프로젝트 트러블슈팅 가이드

작성일: 2025-08-98 작성자: Jay

pyenv 가상환경 관련 이슈

문제: ModuleNotFoundError가 계속

```
from langchain_community.document_loaders import DirectoryLoader

# 디렉토리 로더 초기화
loader = DirectoryLoader(
    "../docs/",
    glob="**/*.md",
    use_multithreading=True,            # 멀티스레드 사용하기
    )

# 문서 로드
docs = loader.load()

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")

# 내용 출력해보기
if docs:
    title = docs[11].metadata.get('title', '제목 없음')
    if title == '제목 없음':
        # docs[n]번째 문서의 내용으로 제목 정하기
        title = f"docs[{11}]번째 문서"
    print(f"\n{title} 문서의 내용:\n")
    print(docs[11].page_content[:100])

else:
    print("\n조건에 맞는 문서를 찾지 못했습니다.")
```

- 셀 출력 (3.3s)

로드된 문서 개수: 50

docs[11]번째 문서 문서의 내용:

🛠️ [문제 이름 또는 현상 요약]

❗ 문제 설명

무엇이 예상과 다르게 작동했는가?

발생한 에러 메시지, 로그, 증상 등

[터미널 로그 또는 에러 메시지 예시]

🔍 원

`loader_cls` 변경

- `loader_cls`
  - 기본 값 = `UnstructuredLoader` 클래스 사용
  - 로더를 사용자 정의하려면 `loader_cls` kwarg에 로더 클래스를 지정해야 함

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

# DirectoryLoader 인스턴스 생성
loader = DirectoryLoader(
    "../docs/",                     # 로드할 디렉토리 경로
    glob="**/*.md",                 # 로드할 파일 패턴 (모든 .md 파일)
    loader_cls=TextLoader           # loader_cls 매개변수를 사용하여 TextLoader 클래스를 지정
)

# 문서 로드
docs = loader.load()

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")     # 로드된 문서 개수: 50
```

```
# 문서 페이지 내용 출력

print(docs[13].page_content[:100])
```

- 셀 출력 (일부 출력해보기)

```
# 🛠️ LangChain Hub / LangSmith Troubleshooting 기록

> 작성일: 2025-08-10
> 작성자: Jay
---

## 1. usern
```

- **`Unstructured`는 `Markdown` 헤더 파싱 ↔ `TextLoader`는 그렇지 않음**

`PythonLoader`

- **`PythonLoader`**: `Python` 소스 코드 파일을 로드해야 하는 경우

```
from langchain_community.document_loaders import PythonLoader

# 현재폴더(.) 의 .py 파일을 모두 조회하여 PythonLoader 로 로드
loader = DirectoryLoader(
    ".",                        # 현재 폴더에서 모두 검색하고 싶을 경우
    glob="**/*.py",
    loader_cls=PythonLoader
    )

# 문서 로드
docs = loader.load()
# docs

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")
```

- 셀 출력

```
로드된 문서 개수: 3
```

- 실제 폴더 구조로 확인해보기

```
.                                       # 현재 폴더
|
├── 00_Document_Loader.ipynb
├── 00_Document_Loader.pdf
├── 01_PDF_Loader.ipynb
├── 01_PDF_Loader.pdf
├── 02_HWP-Loader.ipynb
├── 02_HWP-Loader.pdf
├── 03_CSV_Loader.ipynb
├── 03_CSV_Loader.pdf
├── 04_Excel_Loader.ipynb
├── 04_Excel_Loader.pdf
├── 05_Word_Loader.ipynb
├── 05_Word_Loader.pdf
├── 06_PowerPoint_Loader.ipynb
├── 06_PowerPoint_Loader.pdf
├── 07_WebBase_Loader.ipynb
├── 07_WebBase_Loader.pdf
├── 08_TXT_Loader.ipynb
├── 08_TXT_Loader.pdf
├── 09_JSON_Loader.ipynb
├── 09_JSON_Loader.pdf
├── 10_Arxiv_Loader.ipynb
├── 11_Directory_Loader.ipynb
├── custom_hwp_loader.py                # python 파일_1
├── custom_hwp_loader2.py               # python 파일_2
├── data
|   ├── appendix-keywords-CP949.txt
|   ├── appendix-keywords-EUCKR.txt
|   ├── appendix-keywords-utf8.txt
|   ├── appendix-keywords.txt
|   ├── audio_utils.py                  # python 파일_3
|   ├── chain-of-density.txt
|   ├── client.html
|   ├── people.json
|   ├── reference.txt
|   ├── sample-ppt.pptx
|   ├── sample-word-document.docx
|   ├── SPRI_AI_Brief_2023년12월호_F.pdf
|   ├── titanic.csv
|   ├── titanic.xlsx
|   ├── 디지털 정부혁신 추진계획.hwp
|   └── 디지털_정부혁신_추진계획.pdf
|
├── figures
|   |
|   ├── ***.jpg
|   |
|   └── ***.jpg
└── Img
```

```
from langchain_community.document_loaders import PythonLoader

# PythonLoader로 로드
loader = DirectoryLoader(
    "../06_Document_Loader/data/",                  # 특정 폴더의 특정 내용 출력해보기
    glob="**/*.py",
    loader_cls=PythonLoader
    )
```

```
# 문서 로드
docs = loader.load()
# docs

# 로드된 문서 개수 출력
print(f"로드된 문서 개수: {len(docs)}")

# 내용 출력해보기
if docs:
    print(f"\n{docs[0]}번째 문서의 내용:\n")
    print(docs[0].page_content[:100])

else:
    print("\n조건에 맞는 문서를 찾지 못했습니다.")
```

- 셀 출력

```
로드된 문서 개수: 1
```

```
page_content='import re
import os
from pytube import YouTube
from moviepy.editor import AudioFileClip, VideoFileClip
from pydub import AudioSegment
from pydub.silence import detect_nonsilent


def extract_abr(abr):
    youtube_audio_pattern = re.compile(r"\d+")
    kbps = youtube_audio_pattern.search(abr)
    if kbps:
        kbps = kbps.group()
        return int(kbps)
    else:
        return 0


def get_audio_filepath(filename):
    # audio 폴더가 없으면 생성
    if not os.path.isdir("audio"):
        os.mkdir("audio")

    # 현재 스크립트의 절대 경로 얻기
    current_directory = os.path.abspath("")

    # 파일 경로 생성
    audio_file_path = os.path.join(current_directory, "audio", filename)

    return audio_file_path


def convert_mp4_to_wav(mp4_file_path, wav_file_path):
    # MP4 파일 로드
    audio_clip = AudioFileClip(mp4_file_path)

    # WAV 형식으로 오디오 추출 및 저장
    audio_clip.write_audiofile(wav_file_path, fps=44100, nbytes=2, codec="pcm_s16le")


def download_audio_from_youtube(link):
    # YouTube 객체 생성
    yt = YouTube(link)

    # mp4 오디오만 필터링
    mp4_files = dict()

    # "audio/mp4" 타입의 스트림만 필터링
```

```
    for stream in yt.streams.filter(only_audio=True):
        mime_type = stream.mime_type
        abr = stream.abr
        if mime_type == "audio/mp4":
            abr = extract_abr(abr)
            mp4_files[abr] = stream

    # 키를 기준으로 정렬
    sorted_keys = sorted(mp4_files.keys())
    # 가장 큰 키를 사용하여 값 가져오기
    largest_value = mp4_files[sorted_keys[-1]]
    filename = largest_value.download()

    # 현재 스크립트의 절대 경로 얻기
    current_directory = os.path.abspath("")

    new_filename = os.path.basename(filename.replace(".mp4", ".wav"))

    new_filepath = os.path.join(current_directory, "audio", new_filename)

    # mp4 파일을 wav 파일로 변환
    convert_mp4_to_wav(filename, new_filepath)

    # mp4 파일 삭제
    os.remove(filename)
    return new_filepath


def extract_audio_from_video(video_filepath):
    # MP4 파일 로드
    video = VideoFileClip(video_filepath)
    audio_filepath = get_audio_filepath(video_filepath.replace(".mp4", ".wav"))
    video.audio.write_audiofile(audio_filepath)
    return audio_filepath


class AudioChunk:
    def __init__(self, filepath, min_silence_len=350, silence_thresh=-35):
        self.audio = AudioSegment.from_file(filepath, format="wav")
        self.filepath = filepath
        self.min_silence_len = min_silence_len
        self.silence_thresh = silence_thresh
        self.detect_nonsilent_from_audio()

    @staticmethod
    def make_audio_chunks(audio, non_silent_times):
        audio_chunks = []
        for start, end in non_silent_times:
            audio_chunks.append((audio[start:end], start, end))
        return audio_chunks

    def detect_nonsilent_from_audio(self):
        non_silent_audio_times = detect_nonsilent(
            self.audio,
            min_silence_len=self.min_silence_len,
            silence_thresh=self.silence_thresh,
        )

        non_silent_audios_output = AudioSegment.empty()

        for i in range(len(non_silent_audio_times)):
            non_silent_audios_output += self.audio[
                non_silent_audio_times[i][0] : non_silent_audio_times[i][1]
            ]
        self.audio_chunks = self.make_audio_chunks(self.audio, non_silent_audio_times)
        self.non_silent_audios_output = non_silent_audios_output
        print(f"분석에 사용할 전체 오디오 조각 개수: {len(non_silent_audio_times)}")
```

```
    def audio_splits(self, split_time=100):
        splits = int(self.audio.duration_seconds // split_time + 1)
        audios = []
        for s in range(splits):
            start = s * split_time * 1000
            end = start + split_time * 1000
            audios.append(self.audio[start:end])
        return audios

    ' metadata={'source': '../06_Document_Loader/data/audio_utils.py'}번째 문서의 내용:

    import re
    import os
    from pytube import YouTube
    from moviepy.editor import AudioFileClip, VideoFileCl
```

- *next:* `UpstageLayoutAnalysisLoader`
