https://youtu.be/1aQH5x49Nic?si=dtgtDRfXAefOO7V2

파이 웹 심포지움 3회차 (2025.05.31)

장고 기반으로 PDF 파싱부터 문서 기반 RAG까지

당신의 파이썬/장고 페이스메이커가 되겠습니다. ;-)

파이썬사랑방, 이진석 (me@pyhub.kr)

다룰 내용

- django-pyhub-rag 라이브러리와 업스테이지 Document Parse API 통합
  - PDF 파싱 + 이미지 추출, 이미지 설명 작성까지 1번의 명령으로 수행
- 장고 모델 기반의 강력한 문서 기반 RAG 시스템을 구현하는 전체 과정(장고 프로젝트 생성부터)을 라이브 시연

```
# 라이브러리 설치
pip install--upgrade "django-pyhub-rag[all]"

# 변환 명령 : UPSTAGE_API_KEY, OPENAI_API_KEY 환경변수 필요
#   - 추출된 이미지 저장, 이미지에 대한 설명 작성 => jsonl 생성
python -m pyhub.parser upstage --enable-image-descriptor ./doc.pdf

# django 프로젝트 생성 및 VectorDocument 모델 구현 및 마이그레이션

# 지정 VectorDocument로 데이터 저장 및 자동 임베딩
python manage.py load_jsonl myrag.VectorDocument ./output/doc.jsonl
```

```
from myrag.models import VectorDocument

# 유사 문서 검색
user_input = "..."  # 유저 질의
docs = VectorDocument.objects.similarity_search(user_input)
human_content = f"## Context\n{docs}\n\n##Question#{user_input}"

from pyhub.llm import OpenAILLM

# LLM 에게 <지식 + 질의> 요청
llm = OpenAILLM(model="gpt-4o")
reply = llm.ask(human_content)
print(reply.text)
print(reply.usage)
```

이진석

파이썬 사랑방 주인장

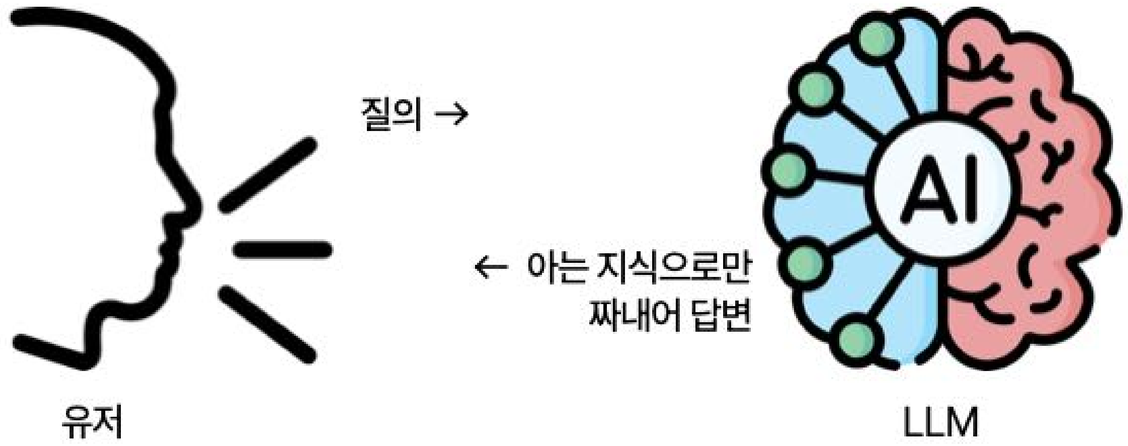
- 파이썬/장고의 높은 생산성을 애정하는 1인
  - 파이썬/장고 고인물
  - 요즘 **RAG**/**MCP**에 꽂혀있어요.
- Ask Django 페이스북 그룹, 운영자
  - https://facebook.com/groups/askdjango
- 인프런, 컨텐츠 크리에이터
  - 파이썬/**장고** 웹서비스 개발 완벽 가이드 with 리액트 (장고 4.2 기준)
  - **장고** 설계철학으로 시작하는 파이썬/장고 입문
  - 파이썬/**장고**로 결제 시작하기 (Feat. 아임포트) - 기본편
  - 파이썬/**장고**로 웹채팅 서비스 만들기 (Feat. Channels) - 기본편
  - ChatGPT 영어 상황극 채팅 서비스 만들기 (Feat. 파이썬/**장고**채널스)

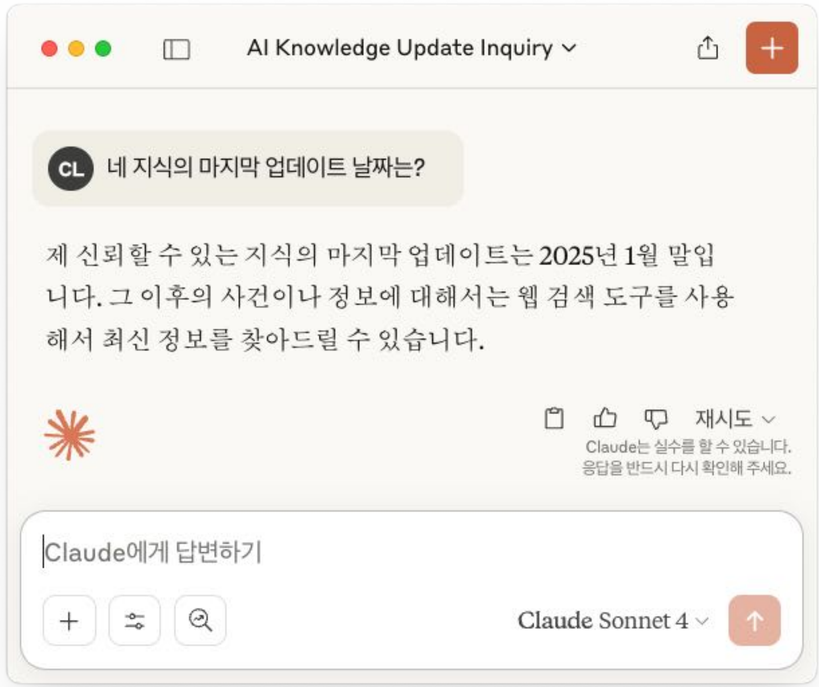
RAG

(검색 증강 생성, Retrieval Augmented Generation)

환각 (hallucination)

사람도 모르는 분야에 대해서는 틀린 답을 합니다. 구글도 다 아는 건 아니죠.



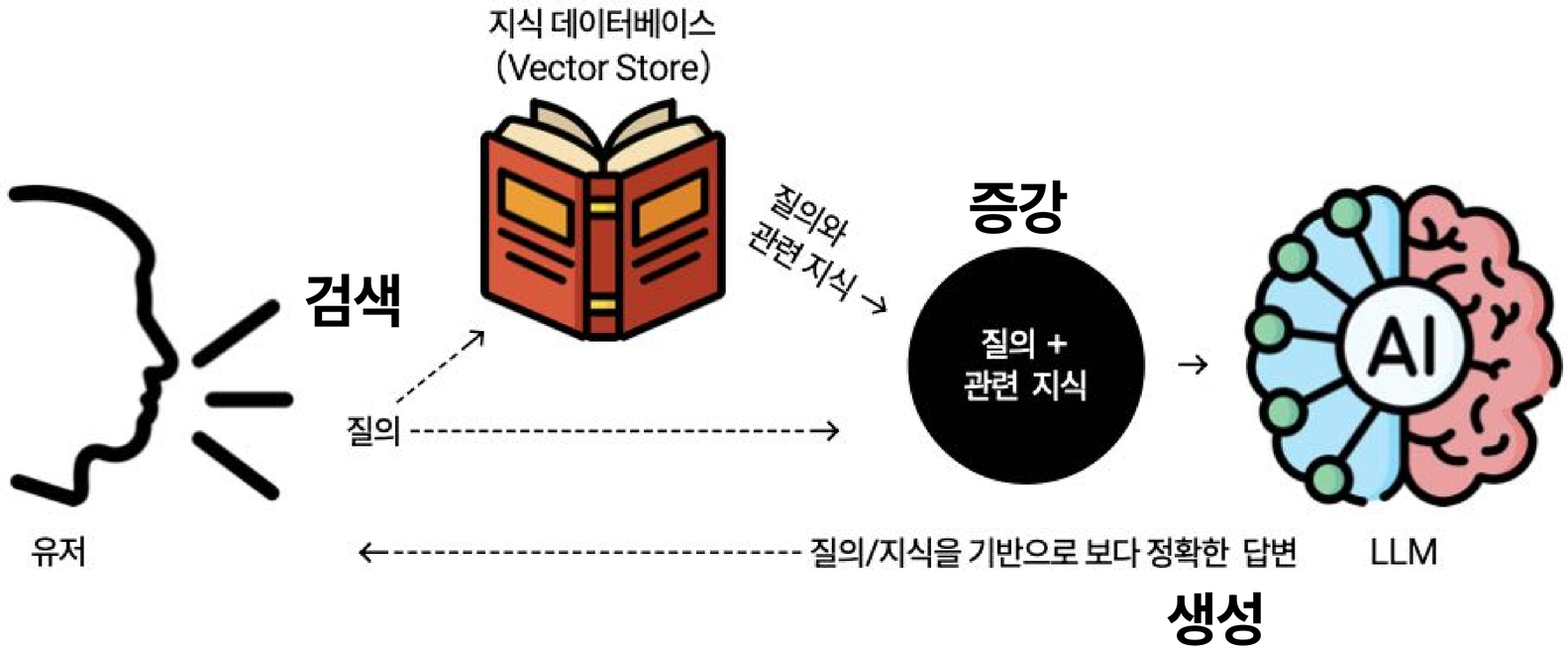


2025년 5월 30일에 질문

관련 지식을 보충하여, 질문과 함께 질의

검색 증강 생성

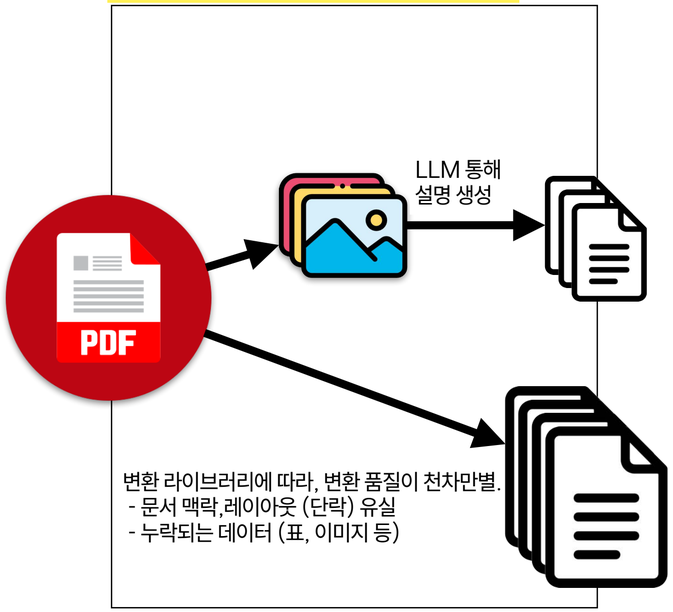
질의 내용과 **유사한 내용**의 레코드를 찾아주는 데이터베이스
질의 시점의 최신 정보 제공 가능



지식 데이터베이스에 지식 넣기

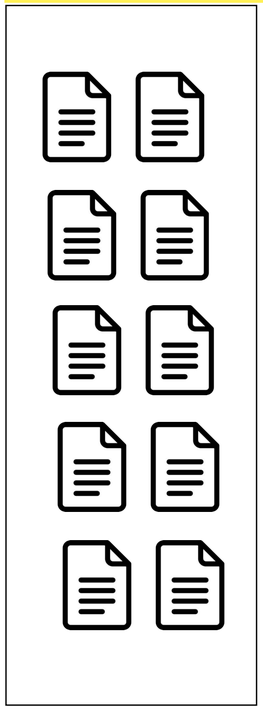
서비스 준비 단계

문서 변환 (문서 포맷을 살려, 텍스트 화)



업스테이지 Document Parse API
애저 AI Document Intelligence
구글 Document AI
라이브러리
- PDFMinder, PDFPlumber, PyPDFium2, PyMuPDF, PyPDF2 등
- unstructured, python-pptx
- Docling
- MarkItDown

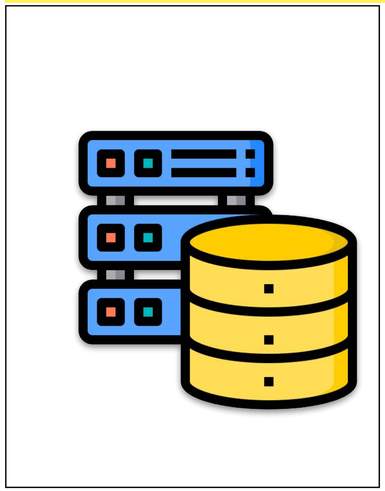
문서 분할 (의미있는 단위의 문서로 나누기)



임베딩 (문서의 의미를 수치화)

0.0	0.3	0.5	...
0.5	0.6	0.3	...
0.9	0.4	0.9	...
0.7	0.6	0.0	...
0.1	0.6	0.5	...
0.4	0.8	0.2	...
0.7	0.3	0.1	...
0.4	0.7	0.9	...
0.7	0.1	0.6	...
0.2	0.9	0.7	...

벡터 스토어 저장 (유사 문서 검색 시, 활용)



PostgreSQL

qdrant

Chroma

FAISS
Scalable Search With Facebook AI

SQLite

문서 변환

문서 레이아웃을 살려 Plain Text 로 변환

업스테이지 Document Parse

django-pyhub-rag 라이브러리에서는 다양한 파싱 API 및 파싱 라이브러리 지원 예정

소개 페이지 : https://www.upstage.ai/products/document-parse

- 지원 포맷
  - PDF, JPEG, PNG, BMP, TIFF, HEIC, DOCS, PPTX, XLSX, HWP, HWPX
- 최대 용량
  - 50MB
- 파일 당 최대 페이지 수
  - 동기 API : 100 페이지 (100페이지를 초과하면, 처음 100페이지만 처리)
  - 비동기 API : 1000 페이지
- 지원 문자셋 : 알파벳/숫자, 한글/한자
  - 베타 : 중국어 한자 (Hanzi), 일본어 한자 (Kanji)

API 공식문서 : https://console.upstage.ai/docs/capabilities/document-digitization/document-parsing

API 요청 예시

업스테이지 Document Parse API 활용 ($0.01 / 페이지)

시트 및 안전 장치

좌석

본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.

운전석[A]
1: 전후 위치 조절
2: 등받이 각도 조절
3: 좌석 높낮이 조절
4: 쿠션 각도 조절(사양 적용 시)
5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)
6: 시트 볼스터 조절
7: 릴렉션 컴포트 시트 스위치(사양 적용 시)
8: 다리 받침대 조절 스위치(전·후 조절)
9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)
10: 헤드레스트 조절

동승석[B], [C]
1: 전후 위치 조절
34

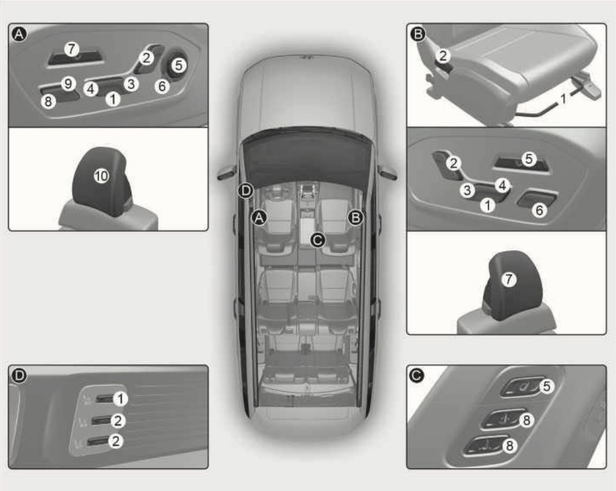
```python
import json
import requests
import os

# API Key 지정 : https://console.upstage.ai/api-keys?api=document-parsing
API_KEY = os.getenv("UPSTAGE_API_KEY")

filepath = "./samples/현대 싼타페 2025 Hyundai Owner's Manual.pdf"  # 입력 파일 : 743 페이지 ($7.43)

url = "https://api.upstage.ai/v1/document-digitization"
headers = {"Authorization": f"Bearer {API_KEY}"}

files = {"document": open(filepath, "rb")}
data = {
    "ocr": "force",
    "model": "document-parse",
    "align_orientation": True,              # 문서 방향 자동 조정
    "coordinates": True,                    # 위치 인식
    "output_formats": "['markdown']",       # or 'html', 'text'
    # 지정 레이아웃 타입의 컨텐츠를 base64 이미지로서 받기
    # https://console.upstage.ai/docs/capabilities/document-digitization/document-parsing#understanding-the-outputs
    "base64_encoding": "['chart', 'equation', 'figure', 'table']",
}
response = requests.post(url, headers=headers, files=files, data=data)

# API 응답을 파일에 쓰기
with open("./samples/현대 싼타페 2025 Hyundai Owner's Manual.json", "wt", encoding="utf-8") as f:
    json.dump(response.json(), f, indent=4, ensure_ascii=False)
```

샘플 코드 : https://console.upstage.ai/docs/capabilities/document-digitization/document-parsing#example
샘플 파일 : https://github.com/pyhub-kr/django-pyhub-rag/tree/main/samples

API 응답

시트 및 안전 장치

좌석



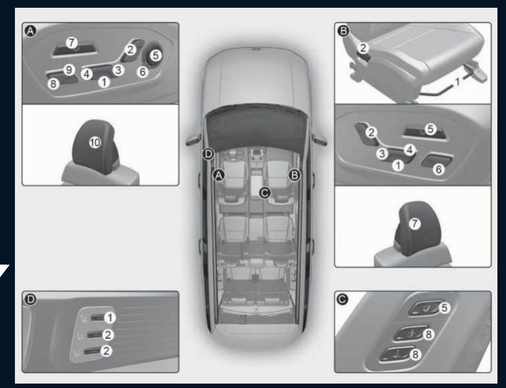
본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.

운전석[A]
1: 전후 위치 조절
2: 등받이 각도 조절
3: 좌석 높낮이 조절
4: 쿠션 각도 조절(사양 적용 시)
5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)
6: 시트 볼스터 조절
7: 릴렉션 컴포트 시트 스위치(사양 적용 시)
8: 다리 받침대 조절 스위치(전·후 조절)
9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)
10: 헤드레스트 조절

동승석[B], [C]
1: 전후 위치 조절
34

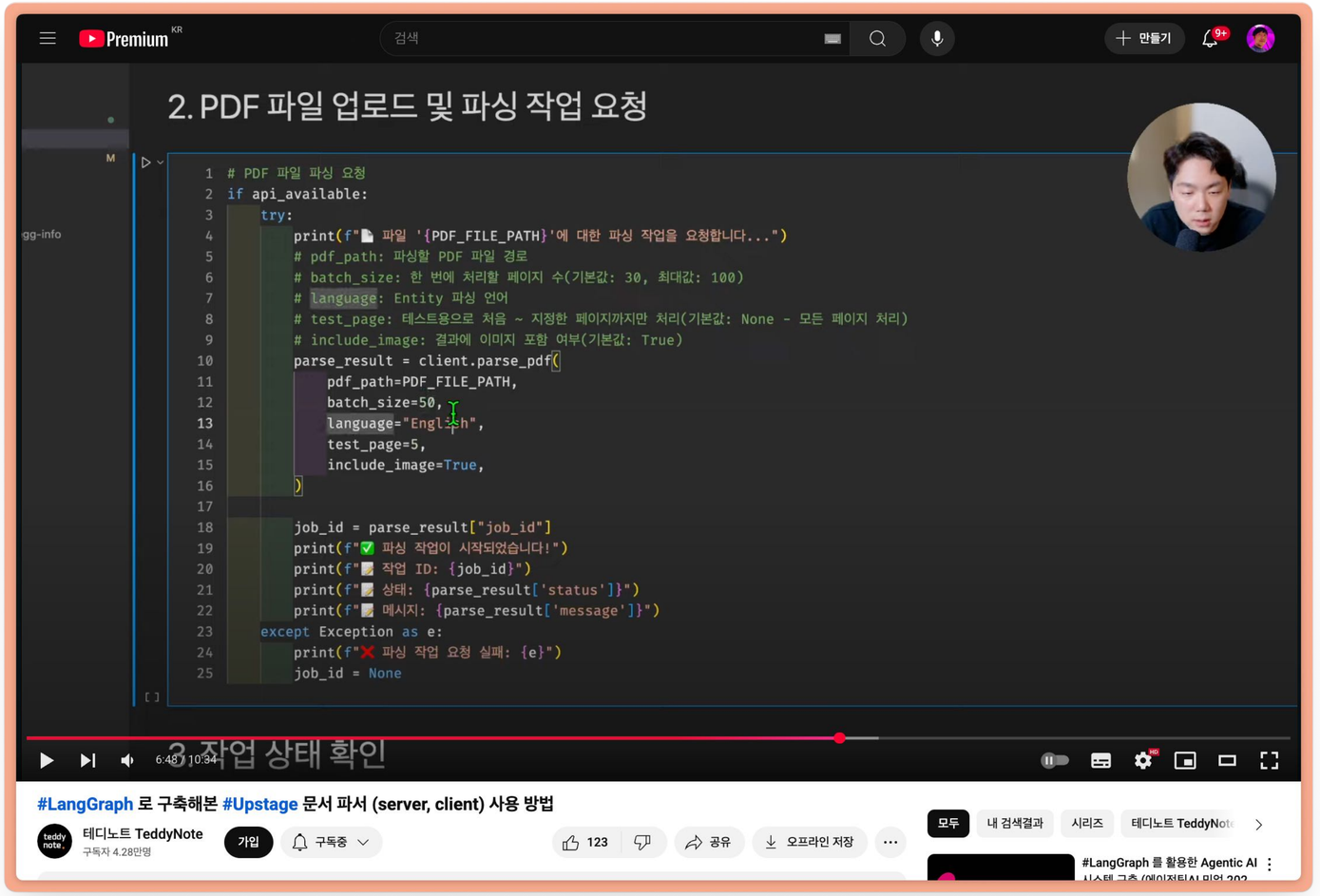

```
{
    "api": "2.0",
    "content": {
        "html": "",
        "markdown": "# 시트 및 안전 장치\n\n# 좌석\n\n![image](/image/placeholder)\n1\n2\n10\n3\nB\n8\n8\n\n본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.\n\n# 운전석[A]\n\n- 1: 전후 위치 조절\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)\n- 6: 시트 볼스터 조절\n- 7: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 8: 다리 받침대 조절 스위치(전·후 조절)\n- 9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)\n- 10: 헤드레스트 조절\n\n# 동승석[B], [C]\n\n1: 전후 위치 조절\n34",
        "text": ""
    },
    "elements": [
        {
            "category": "heading1",
            "content": {
                "html": "",
                "markdown": "# 시트 및 안전 장치",
                "text": ""
            },
            "coordinates": [
                { "x": 0.0653, "y": 0.0308 },
                { "x": 0.2204, "y": 0.0308 },
                { "x": 0.2204, "y": 0.0497 },
                { "x": 0.0653, "y": 0.0497 }
            ],
            "id": 0,
            "page": 1
        },

"category": "heading1",
"markdown": "# 좌석",

"base64_encoding": "/9j/2wCEAAIBAQEBAQIBAQ BASE64 데이터 생략
"category": "figure",
"markdown": "![image](/image/placeholder)\n1\n2\n10\n3\nB\n8\n8",

"category": "caption",
"markdown": "본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.",

"category": "heading1",
"markdown": "# 운전석[A]",

"category": "list",
"markdown": "- 1: 전후 위치 조절\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)\n- 6: 시트 볼스터 조절\n- 7: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 8: 다리 받침대 조절 스위치(전·후 조절)\n- 9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)\n- 10: 헤드레스트 조절\n",

"category": "heading1",
"markdown": "# 동승석[B], [C]",

"category": "paragraph",
"markdown": "1: 전후 위치 조절\n34",
```


여러 이미지에 대해 동일한 이름 지정



django-pyhub-rag 라이브러리 만든 계기

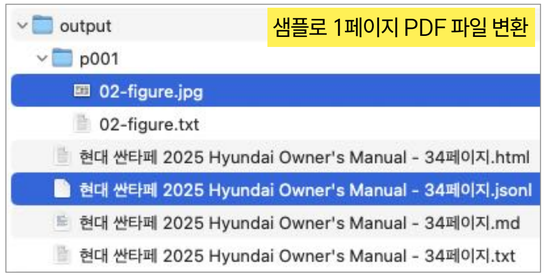
테디노트 : LangGraph로 구축해본 Upstage 문서 파서 (server, client) 사용 방법 (2025.03.10)



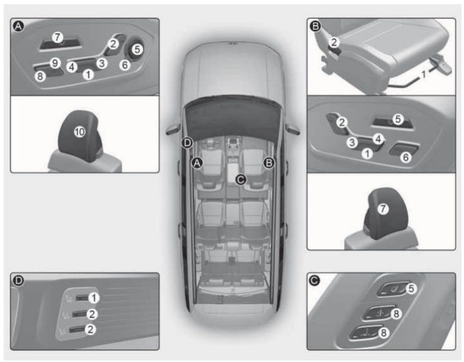
django-pyhub-rag 활용

Document Parse + VLM (Vision Language Model) 통해 이미지 설명 생성

```
pip install --upgrade "django-pyhub-rag[all]"

# 변환 명령 : UPSTAGE_API_KEY, OPENAI_API_KEY 환경변수 필요
python -m pyhub.parser upstage --enable-image-descriptor 파일경로.pdf
python -m pyhub.parser upstage -i 파일경로.pdf
# API 캐시 활성화 : 파일 시스템 캐시, 최대 5000개, 타임아웃 3일
python -m pyhub.parser upstage --enable-cache -i 파일경로.pdf
```



샘플로 1페이지 PDF 파일 변환



생성된 jsonl 파일


```
{
  "page_content": "# 시트 및 안전 장치\n\n# 좌석\n\n![image](p001/02-figure.jpg)\n1\n2\n10\n3\nB\n8\n8\n\n본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.\n\n# 운전석[A]\n\n- 1: 전후 위치 조절\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)\n- 6: 시트 볼스터 조절\n- 7: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 8: 다리 받침대 조절 스위치(전·후 조절)\n- 9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)\n- 10: 헤드레스트 조절\n\n\n# 동승석[B], [C]\n\n1: 전후 위치 조절\n34",

  "metadata": {
    "total_pages": 1,
    "api": "2.0",
    "model": "document-parse-250404",
    "page": 1,
    "image_descriptions": "<image name='p001/02-figure.jpg'><title>자동차 내부 기능 설명</title>\n<details>이 이미지는 자동차 내부의 다양한 기능과 조작 버튼을 설명하는 다이어그램입니다. 각 섹션은 특정 기능을 나타내며, 번호가 매겨져 있습니다.</details>\n<entities>자동차, 시트 조정 버튼, 헤드레스트, 도어 패널, 내부 구조</entities>\n<hypothetical_questions>이러한 기능들이 운전자의 편안함에 어떤 영향을 미칠까요? 자동차 내부 디자인이 안전성에 미치는 영향은 무엇일까요? 다양한 기능이 사용자 경험을 어떻게 개선할 수 있을까요?</hypothetical_questions></image>",
    "source": "현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.pdf"
  }
}
```


페이지 번호 디렉토리에 자동으로 저장

pyhub.parser upstage -i 명령

API 키 누락으로 인한 오류 상황

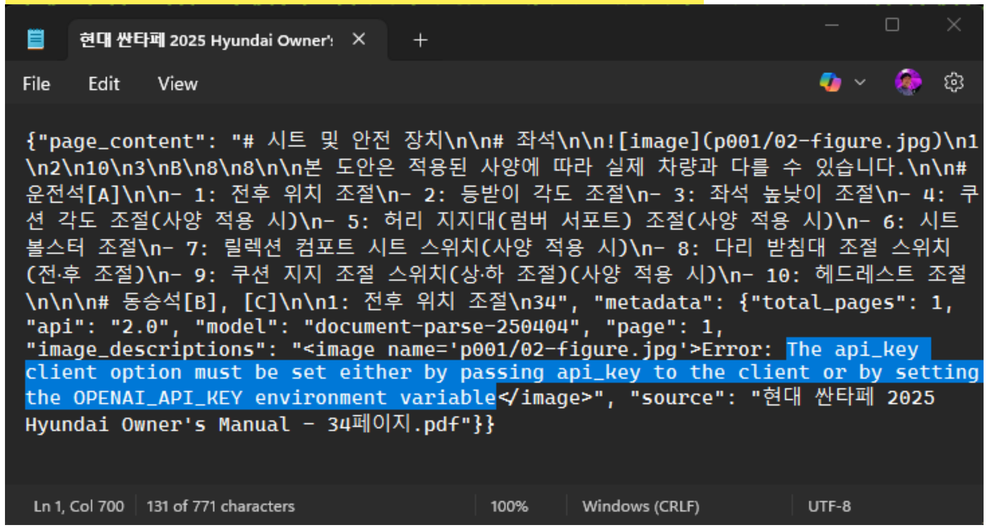
UPSTAGE_API_KEY 환경변수를 누락한 상황

```
> py -m pyhub.parser upstage -i '.\현대 싼타페 2025 Hyundai Owner''s Manual - 34페이지.pdf'
Usage: python -m pyhub.parser upstage [OPTIONS] [INPUT_PATH]
Try 'python -m pyhub.parser upstage --help' for help.
╭─ Error ──────────────────────────────────────────────────────────────────────╮
│ Invalid value: --upstage-api-key 옵션이나 UPSTAGE_API_KEY 환경 변수를 통해 Upstage API Key를 │
│ 설정해주세요.                                                                │
╰──────────────────────────────────────────────────────────────────────────────╯
```

--upstage-api-key 인자를 통해 키를 지정한 상황

```
> py -m pyhub.parser upstage -i --upstage-api-key 'up_                    D' '.\현대 싼
타페 2025 Hyundai Owner''s Manual - 34페이지.pdf'
[2025-05-30 14:59:37,277] PDF 파일 : 총 1 페이지
[2025-05-30 14:59:38,740] Upstage document-parse-250404 (2.0) API 요청에서 8개의 요소를 찾았습니
다.
[2025-05-30 14:59:38,743] 8개의 요소에서 1개의 이미지를 찾았습니다.
[2025-05-30 14:59:38,743] gpt-4o-mini 모델을 통해 이미지 설명을 생성합니다.
[2025-05-30 14:59:38,748] request describe_images [1/1] : p001/02-figure.jpg
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.jsonl 경로에 1개의 Document를
jsonl 포맷으로 생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.md 경로에 통합 문서를
생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.html 경로에 통합 문서를
생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.txt 경로에 통합 문서를
생성했습니다.
```

그렇지만 이미지 설명을 생성을 위한 OPENAI_API_KEY 환경변수를 누락된 상황



샘플 파일/데이터 : https://github.com/pyhub-kr/dump-data/tree/main/document-parse-싼타페-2025-40여페이지

pyhub.parser upstage -i 명령

Document Parse API 호출 + 이미지 설명 작성 + jsonl 파일 생성

파워쉘

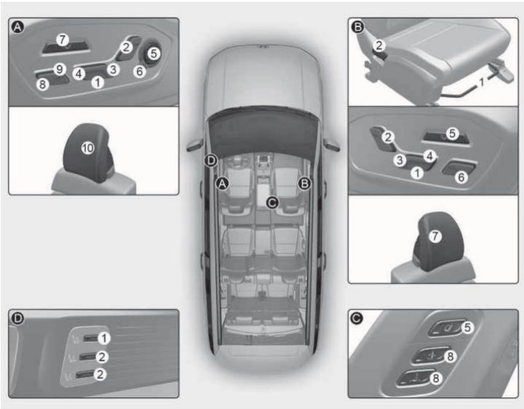
```
~
> $env:OPENAI_API_KEY = "sk-                                                    nH"
~
> $env:UPSTAGE_API_KEY = "up_                                                   D"
~
> py -m pyhub.parser upstage -i ".\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.pdf"
출력 폴더 output이(가) 이미 존재합니다. 삭제 후에 재생성하시겠습니까? [y/N]: y
[2025-05-30 15:05:09,487] PDF 파일 : 총 1 페이지
[2025-05-30 15:05:09,705] Upstage document-parse-250404 (2.0) API 요청에서 8개의 요소를 찾았습니다.
[2025-05-30 15:05:09,734] 8개의 요소에서 1개의 이미지를 찾았습니다.
[2025-05-30 15:05:09,734] gpt-4o-mini 모델을 통해 이미지 설명을 생성합니다.
[2025-05-30 15:05:09,735] request describe_images [1/1] : p001/02-figure.jpg
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.jsonl 경로에 1개의 Document를 jsonl 포맷으로 생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.md 경로에 통합 문서를 생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.html 경로에 통합 문서를 생성했습니다.
성공: output\현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.txt 경로에 통합 문서를 생성했습니다.
```

현대 싼타페 2025 Hyundai Owner's

File Edit View

{"page_content": "# 시트 및 안전 장치\n\n# 좌석\n\n![image](p001/02-figure.jpg)\n1\n2\n10\n3\nB\n8\n8\n\n본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.\n\n# 운전석[A]\n\n- 1: 전후 위치 조절\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)\n- 6: 시트 볼스터 조절\n- 7: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 8: 다리 받침대 조절 스위치(전·후 조절)\n- 9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)\n- 10: 헤드레스트 조절\n\n\n# 동승석[B], [C]\n\n1: 전후 위치 조절\n34", "metadata": {"total_pages": 1, "api": "2.0", "model": "document-parse-250404", "page": 1, "image_descriptions": "<image name='p001/02-figure.jpg'><title>자동차 내부 기능 설명</title>\n<details>이 이미지는 자동차 내부의 다양한 기능과 조작 버튼을 설명하는 다이어그램입니다. 각 섹션은 특정 기능을 나타내며, 번호가 매겨져 있습니다.</details>\n<entities>자동차, 시트 조정 버튼, 헤드레스트, 도어 패널, 내부 구조</entities>\n<hypothetical_questions>이 기능들이 자동차의 안전성과 편안함에 어떤 영향을 미칠까요? 사용자가 이러한 기능을 쉽게 이해하고 사용할 수 있도록 하기 위해 어떤 추가 정보가 필요할까요?</hypothetical_questions></image>", "source": "현대 싼타페 2025 Hyundai Owner's Manual - 34페이지.pdf"}}

Ln 1, Col 883 | 321 of 954 characters | 100% | Windows (CRLF) | UTF-8



```
# macOS 의 경우
export UPSTAGE_API_KEY="..."
export OPENAI_API_KEY="..."
```

샘플) 생성된 jsonl 파일 (랭체인 Document 포맷)

문서의 Element 합치기 전략 : page (디폴트), element, none


```
{"page_content": "# 3. 시트 및 안전 장치\n\n좌석 ····· 34\n앞좌석 ····· 40\n뒷좌석 ····· 55\n열선 시트 ····· 71\n앞좌석 열선 시트 사용 ····· 71\n뒷좌석 열선 시트 사용 ····· 72\n통풍 시트 ····· 74\n앞좌석 통풍 시트 사용 ····· 74\n안전벨트 ····· 75\n안전벨트 구속 장치 ····· 75\n안전벨트 프리텐셔너 ····· 82\n안전벨트 사용 시 유의 사항 ····· 83\n안전벨트의 적절한 관리 ····· 84\n어린이 보호용 장치 (보조 좌석)····· 85\n어린이 보호용 장치의 장착····· 85\n에어백 (보조 구속 장치) ····· 91\n에어백 관련 부품 ····· 92\n에어백 경고등 ····· 94\n에어백 장치의 구성 및 기능 ····· 95\n안전벨트 착용의 중요성 ····· 96\n운전석 에어백/무릎 에어백 ····· 97\n동승석 에어백 ····· 98\n승객 구분 시스템 103)------- 99\n사이드 에어백 ····· 106\n커튼 에어백 ····· 108\n에어백의 작동/비작동 조건 ····· 109\n에어백 정비 ····· 115\n에어백 경고 라벨 ····· 116", "metadata": {"total_pages": 84, "api": "2.0", "model": "document-parse-250404", "page": 1, "source": "현대 싼타페 2025 Hyundai Owner's Manual - 33부터 116페이지.pdf"}}
```

```
{"page_content": "# 시트 및 안전 장치\n\n# 좌석\n\n![image](p002/02-figure.jpg)\n1\n2\n10\n3\nB\n8\n8\n\n본 도안은 적용된 사양에 따라 실제 차량과 다를 수 있습니다.\n\n# 운전석[A]\n\n- 1: 전후 위치 조절\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 허리 지지대(럼버 서포트) 조절(사양 적용 시)\n- 6: 시트 볼스터 조절\n- 7: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 8: 다리 받침대 조절 스위치(전·후 조절)\n- 9: 쿠션 지지 조절 스위치(상·하 조절)(사양 적용 시)\n- 10: 헤드레스트 조절\n\n\n# 동승석[B], [C]\n\n1: 전후 위치 조절\n34", "metadata": {"total_pages": 84, "api": "2.0", "model": "document-parse-250404", "page": 3, "image_descriptions": "<image name='p002/02-figure.jpg'><title>자동차 내부 기능 설명</title>\n<details>이 이미지는 자동차 내부의 다양한 기능과 조작 방법을 설명하는 다이어그램입니다. 각 기능은 번호로 표시되어 있으며, 좌석 조정, 창문 조작, 헤드레스트 조정 등의 세부 사항이 포함되어 있습니다.</details>\n<entities>자동차, 좌석 조정 레버, 창문 스위치, 헤드레스트, 내부 다이어그램</entities>\n<hypothetical_questions>이 기능들이 자동차의 안전성과 편안함에 어떻게 기여할까요? 사용자가 이러한 기능을 쉽게 이해하고 사용할 수 있도록 하기 위해 어떤 추가 정보가 필요할까요?</hypothetical_questions></image>", "source": "현대 싼타페 2025 Hyundai Owner's Manual - 33부터 116페이지.pdf"}}
```

```
{"page_content": "3\n\n- 2: 등받이 각도 조절\n- 3: 좌석 높낮이 조절(사양 적용 시)\n- 4: 쿠션 각도 조절(사양 적용 시)\n- 5: 다리 받침대 조절 스위치(사양 적용 시)\n- 6: 릴렉션 컴포트 시트 스위치(사양 적용 시)\n- 7: 헤드레스트 조절\n- 8: 동승석 측면 워크인 스위치(사양 적용 시)\n\n\n# 운전석[D]\n\n- 1: 에르고 모션\n- 2: 운전석 자세 메모리 시스템\n\n\n| i 알아두기 |\n| --- |\n| 주행 전 좌석의 위치, 각도, 높낮이 등을 조절하여 올바른 운전 자세가 되도록 하십시오. |\n\n\n![image](p003/05-figure.jpg)\n2\n3\n4\n5 2\n4\n5\n1\n3ND\n2", "metadata": {"total_pages": 84, "api": "2.0", "model": "document-parse-250404", "page": 5, "image_descriptions": "<image name='p003/04-table.jpg'><title>\n주행 전 좌석의 위치 및 조정 안내\n</title>\n<details>\n주행 전 좌석의 위치, 각도, 높낮이 등을 조절하여 올바른 운전 자세를 유지하도록 안내합니다.\n</details>\n<entities>\n주행 전 좌석, 위치, 각도, 높낮이, 운전 자세\n</entities>\n<hypothetical_questions>\n- 올바른 운전 자세를 유지하기 위해 좌석 조정 외에 어떤 요소가 중요할까요?\n- 좌석 조정이 운전자의 안전에 미치는 영향은 무엇인가요?\n</hypothetical_questions></image>\n\n<image name='p003/05-figure.jpg'><title>자동차 좌석 조정 및 기능 안내</title>\n<details>이 이미지는 자동차 내부의 좌석 조정 기능을 설명하는 다이어그램입니다. 좌석의 다양한 조정 방법과 기능이 번호로 표시되어 있습니다.</details>\n<entities>자동차, 좌석, 조정 레버, 헤드레스트, 버튼</entities>\n<hypothetical_questions>1. 이 조정 기능이 운전자의 편안함에 어떤 영향을 미칠까요? 2. 좌석 조정 기능이 안전성에 미치는 영향은 무엇일까요? 3. 다양한 좌석 조정 옵션이 소비자 선택에 어떤 역할을 할까요?</hypothetical_questions></image>", "source": "현대 싼타페 2025 Hyundai Owner's Manual - 33부터 116페이지.pdf"}}
```

```
{"page_content": "# 시트 및 안전 장치\n\n# 2열 좌석(5인승, 7인승)[A]\n\n1: 전후 위치 조절\n2: 등받이 각도 조절\n3: 3열 승하차 스위치(7인승, 사양 적용 시)\n4: 헤드레스트 조절\n5: 3열 승하차 스위치(7인승, 사양 적용 시)\n2열 좌석(6인승)[B]\n1: 전후 위치 조절\n2: 등받이 각도 조절\n3: 쿠션 각도 조절\n4: 릴렉션 모드 스위치\n5: 슬라이딩 록킹 해제 스위치\n6: 헤드레스트 조절\n7: 3열 승하차 스위치\n2열 좌석 뒤 [C]\n1: 3열 승하차 스트랩(비상용)\n화물칸[D]\n1: 2열 등받이 접힘 스위치(5인승, 7인승)(사양 적용 시)\n2: 2열 등받이 접힘/펼침 스위치(6인승)(사양 적용 시)\n3열 좌석[E]\n1: 등받이 접힘/펼침 스트랩\n2: 등받이 각도 조절\n3: 헤드레스트 조절", "metadata": {"total_pages": 84, "api": "2.0", "model": "document-parse-250404", "page": 7, "source": "현대 싼타페 2025 Hyundai Owner's Manual - 33부터 116페이지.pdf"}}
```

```
{"page_content": "3\n\n# 인포테인먼트 시스템\n\n![image](p005/02-figure.jpg)\nQ 차량\n공조 밀선/통풍 편의\n>\n앞선/힘을자 분야 기능을 일정합니다.\n시트\n송하차 편의\n중학자 시 사회의 만들 여초의 설립하여 또한철 확보입니다.\n라이트\n스마트 서포트\n도어\n+포츠모드 드는 34도로 주민 no. 기능 쉽게 시작종합니다.\n디지털키 에르고 모션 시트\n입관공학적인 지도 특허물을 숯불하게 미흐 저장, 주방강산맛\n페이 만남성공 닭살 시킵니다.\n송 三\n\n인포테인먼트 시스템의 설정 > 차량 > 시트를 차례로 선택하십시오. 각종 편의 기능을 사용할 수\n있습니다.\n\n# · 열선/통풍 편의\n\n- - 공조 연동 자동 제어: 히터 및 에어컨 시스템과 연동하여 스티어링 휠 열선 및 운전석 열선 시\n- 트/통풍 시트를 자동으로 제어합니다.\n\n\n- · 승하차 편의: 운전석, 동승석(사양 적용 시) 또는 스티어링 휠의 이동 여부를 설정할 수 있습니\n- 다. 자세한 내용은 5장 내 '운전석 자세 메모리 시스템'을 참고하십시오.\n- · 시트 스위치 알림: 시트 스위치 조작 및 위치 변경 정보가 시트 이미지와 함께 표시됩니다.\n- · 스마트 서포트: 스포츠 모드 또는 고속 주행 시 시트 볼스터 지지가 높아집니다.\n- · 에르고 모션 시트\n\n\n- - 컴포트 스트레칭: 컴포트 스트레칭 기능의 세기 및 작동 시간을 선택할 수 있습니다.\n- - 자세 보조: 1시간 주행 후 자동으로 운전석의 엉덩이와 허리 부분을 조절하여 운전자의 자세\n-를 보조해줍니다.\n- - 주행 중 허리 보호 기능: 오랜 시간 연속해서 주행할 경우 일정한 패턴으로 럼버 서포트를 반\n- 복 작동합니다.\n\n\n# i 알아두기\n\n- · 인포테인먼트 시스템에 표시되는 내용은 차량의 사양 구성에 따라 해당 메뉴가 없거나 이 취급\n- 설명서와 다르게 나타날 수 있습니다.\n- · 인포테인먼트 시스템은 업데이트로 변경될 수 있습니다. 자세한 내용은 인포테인먼트 시스템\n- 또는 간편 설명서의 웹 매뉴얼을 참고하십시오.\n", "metadata": {"total_pages": 84, "api": "2.0", "model": "document-parse-250404", "page": 9, "image_descriptions": "<image name='p005/02-figure.jpg'><title>설정 메뉴 화면</title>\n<details>이 이미지는 어두운 배경의 설정 메뉴 화면을 보여줍니다. 메뉴 항목에는 '일반', '시스템', '라이브', '도이', '다시 시작', '파이' 등이 포함되어 있으며, '스마트 서비스' 옵션이 활성화되어 있습니다.</details>\n<entities>설정 메뉴, 스마트 서비스, 메뉴 항목</entities>\n<hypothetical_questions>이 설정 메뉴는 어떤 기기에서 사용되고 있을까요? '스마트 서비스' 기능은 사용자에게 어떤 이점을 제공할까요? 사용자가 이 메뉴를 통해 어떤 설정을 변경할 가능성이 있을까요?</hypothetical_questions></image>", "source": "현대 싼타페 2025 Hyundai Owner's Manual - 33부터 116페이지.pdf"}}
```

이미지 설명 스크립트, 커스텀 지원

```
# pyhub mcptools 설정파일 생성, 경로, 출력, 수정
python -m pyhub toml create # 설정파일 생성
python -m pyhub toml path   # 설정파일 경로 출력
python -m pyhub toml show   # 설정파일 내용 표준출력
python -m pyhub toml edit   # 설정파일 편집
```

~/.pyhub-rag/config.toml

```
[env]
# UPSTAGE_API_KEY = "up_xxxxx..."
# OPENAI_API_KEY = "sk-xxxxx..."
# ANTHROPIC_API_KEY = "sk-ant-xxxxx..."
# GOOGLE_API_KEY = "AIxxxxx...."
# DATABASE_URL = "postgresql://postgres:pw@localhost:5432/postgres"

USER_DEFAULT_TIME_ZONE = "Asia/Seoul"

[prompt_templates.describe_image]
system = """Analyze the given image using the provided context. Then produce a structured output with:

1) A concise title,
2) Key details or insights,
3) Recognized entities or objects,
4) Relevant hypothetical questions.

Output must be coherent and in the specified format."""

user = """# Output Format
<title>...</title>
<details>...</details>
<entities>...</entities>
<hypothetical_questions>...</hypothetical_questions>

{% if context %}# Context
{{ context }}{% endif %}

# Requirements
- Provide a short, descriptive title for the image.
- Clearly summarize the main details.
- Identify and list key entities or objects.
- Pose hypothetical questions that explore deeper implications or ideas.
{% if language %}- Ensure Reply in {{ language }} Language{% endif %}"""
```

장고 템플릿 문법 지원 (조건문, 반복문 가능)

```
[prompt_templates.describe_table]
system = """Analyze the given table based on the provided context and extract structured information.

Generate the following:
1. **Title**: A brief, descriptive title summarizing the table.
2. **Details**: Key insights and numerical values extracted from the table.
3. **Entities**: Important terms, numbers, or categories present in the table.
4. **Hypothetical Questions**: Thought-provoking questions related to the table's data.

Ensure clarity, numerical accuracy, and structured output."""

user = """# Output Format
Structure your response using the following tags:

  - `<title>`: Concise summary of the table.
  - `<details>`: Key insights and numerical data.
  - `<entities>`: List of relevant terms, numbers, or categories.
  - `<hypothetical_questions>`: Open-ended questions based on the table.

{% if context %}# Context
{{ context }}{% endif %}

# Example Output

<title>
Annual Financial Performance of the Company
</title>
<details>
The table displays the company's financial performance for 2024.
Revenue reached $500M, with a net profit of $100M (20% margin).
R&D investment increased to $80M.
</details>
<entities>
Revenue, Net Profit, R&D Investment, Market Share, Total Assets
</entities>
<hypothetical_questions>
- How will the increase in R&D investment impact future profits?
- Can the company sustain its revenue growth over the next five years?
</hypothetical_questions>

# Notes
- Include numerical values and relevant terms.
- Summarize key takeaways from the table.
- Ensure questions encourage deeper analysis.
{% if language %}- Ensure Reply in {{ language }} Language{% endif %}"""
```

장고 템플릿 문법 지원 (조건문, 반복문 가능)

```
❯ uv run -m pyhub.llm describe ./sample1.jpg
<title>Coastal View with Breakwater</title>
<details>This image captures a serene coastal scene featuring a breakwat
er structure. The water is calm with gentle waves, and the sky is clear,
 indicating a sunny day. In the background, there are hills and some bui
ldings, suggesting a developed area near the coast.</details>
<entities>Breakwater, ocean, waves, coastline, buildings, hills</entitie
s>
<hypothetical_questions>What impact does coastal development have on mar
ine ecosystems? How might climate change affect the stability of breakwa
ter structures? What recreational activities could be popular in this co
astal area?</hypothetical_questions>
```

```
> py -m pyhub.llm ask "성심당은 어느 도시에 있음?"
성심당은 대한민국 대전광역시에 위치한 유명한 제과점입니다. 이곳은 특히 빵과 디저트로 유명하며,
많은 사람들에게 사랑받고 있습니다.
~
>
~
> py -m pyhub.llm ask "성심당은 어느 도시에 있음?" --cost
성심당은 대한민국 대전광역시에 위치한 유명한 제과점입니다. 이곳은 특히 빵과 케이크로 유명하며,
많은 사람들에게 사랑받고 있습니다.
        예상 비용
```

항목	값
입력 비용	$ 0.000002
출력 비용	$ 0.000025
총 비용	$ 0.000027
원화 환산	₩ 0.035100

<title>방파제가 있는 해안 풍경</title>
<details>이 이미지는 방파제 구조물이 포함된 평화로운 해안 풍경을 담고 있습니다. 물결은 잔잔하고 파도는 부드러우며, 하늘은 맑아 화창한 날씨임을 나타냅니다. 배경에는 언덕과 몇몇 건물이 보여, 해안 근처에 개발된 지역이 있음을 암시합니다.</details>
<entities>방파제, 바다, 파도, 해안선, 건물, 언덕</entities>
<hypothetical_questions>해안 개발이 해양 생태계에 미치는 영향은 무엇인가요? 기후 변화는 방파제 구조물의 안정성에 어떤 영향을 줄 수 있을까요? 이 해안 지역에서는 어떤 여가 활동이 인기를 끌 수 있을까요?</hypothetical_questions>

랭체인 UpstageDocumentParseParser

동기 방식만 지원

- 배치크기 옵션
  - API 호출 시, 여러 페이지 지원 여부
  - 배치 여부 지정 (1페이지 or 10페이지)
- 문서 머지 전략
  - page, element, none

```python
from langchain_core.document_loaders import BaseBlobParser, Blob


class UpstageDocumentParseParser(BaseBlobParser):
    # ...
    def _get_response(self, files: Dict) -> List:
        headers = {
            "Authorization": f"Bearer {self.api_key}",
        }
        response = requests.post(
            self.base_url,
            headers=headers,
            files=files,
            data={
                "ocr": self.ocr,                                # "auto", "force"
                "model": self.model,                            # "document-parse"
                "output_formats": f"['{self.output_format}']",  # html (default), markdown, text
                "coordinates": self.coordinates,                # True (default)
                "base64_encoding": f"{self.base64_encoding}",   # [] (default)
            },
        )
        response.raise_for_status()
        result = response.json().get("elements", [])
        return result
```

GitHub 저장소 : https://github.com/langchain-ai/langchain-upstage/blob/main/libs/upstage/langchain_upstage/document_parse_parsers.py#L169

django-pyhub-rag UpstageDocumentParseParser

동기/비동기 방식 지원

- 배치크기 옵션
  - API 호출 시, 여러 페이지 지원 여부
  - 배치 크기 직접 지정 지원
- 문서 머지 전략
  - page, element, none
- 다양한 페이지 번호 지정 지원
- LLM통한 이미지 설명 지원
- API 캐싱 지원 (장고 캐싱 기반)
  - 동일 페이지에 대해서는 API 요청 X
    (배치 크기 = 1 지정 필요)
- 비동기 지원
- 장고 File API 기반으로 다양한 스토리지 지원
  - AWS S3, Azure Storage 등 상의 PDF 읽기 가능

```python
# src/pyhub/parser/upstage/parser.py

from django.core.files import File

class UpstageDocumentParseParser:
    # ...

    async def _call_document_parse_api(self, files: dict, timeout: int = DEFAULT_TIMEOUT) -> dict:
        headers = {
            "Authorization": f"Bearer {self.upstage_api_key}",
        }
        data = {
            "ocr": self.ocr_mode,
            "model": self.model,
            "output_formats": "['html', 'text', 'markdown']",
            "coordinates": self.coordinates,
            "base64_encoding": "[" + ",".join(f"'{el}'" for el in self.base64_encoding_category_list) + "]",
        }

        response_data: bytes = await cached_http_async(
            self.api_url,
            method="POST",
            headers=headers,
            data=data,
            files=files,
            timeout=timeout,
            ignore_cache=self.ignore_cache,
            cache_alias="upstage",
        )
        return json.loads(response_data)
```

GitHub 저장소 : https://github.com/pyhub-kr/django-pyhub-rag/blob/main/src/pyhub/parser/upstage/parser.py#L491

pyhub.llm 활용 예시

응답 생성

OpenAI, Anthropic, Google, Upstage, Ollama을 단일 인터페이스로 지원

```python
from pyhub.llm import OpenAILLM
# UpstageLLM, AnthropicLLM, GoogleLLM, OllamaLLM

llm = OpenAILLM(
    # model="gpt-4o-mini",
    # temperature=0.2,
    # max_tokens=1000,
    # api_key=  # default: OPENAI_API_KEY 환경변수
)

reply = llm.ask("hello")
print(reply)
print(reply.usage)
```

```
Hello! How can I assist you today?
Usage(input=8, output=9)
```

```python
# UPSTAGE_API_KEY 환경변수
llm = UpstageLLM(
    # model="solar-mini",
)
```

```
Hello! How can I assist you today? If you have any questions or need hel
p with anything, feel free to ask.
Usage(input=12, output=26)
```

```python
# ANTHROPIC_API_KEY 환경변수
llm = AnthropicLLM(
    # model="claude-3-5-haiku-latest",
)
```

```
Hi there! How are you doing today? Is there anything I can help you with?
Usage(input=8, output=21)
```

```python
# GOOGLE_API_KEY 환경변수
llm = GoogleLLM(
    # model="gemini-2.0-flash",
)
```

```
Hello! How can I help you today?

Usage(input=1, output=10)
```

```python
llm = OllamaLLM(
    # model="mistral",
)
```

```
 Hello! How can I help you today? If you have any questions or need assis
tance with something, feel free to ask. I'm here to help!

If you just want to chat or share some thoughts, that's great too! Let's
talk about whatever is on your mind. :)
None
```

LLM.create

지원 모델명 타입 지정으로 자동 완성 지원


```python
from pyhub.llm import LLM

# 여러 LLM API를 활용해야할 때, model명 지정 만으로 손쉽게 스위칭
llm = LLM.create(model="gemini-1.5-flash")
# llm = GoogleLLM(model="gemini-1.5-flash") 과 동일

reply = llm.ask("hello")
print(reply)
print(reply.usage)
```

stream 지원

Generator[Reply] 반환. 마지막 chunk 응답에 usage

```
In [28]: for reply in llm.ask("대전 여행지 3곳 알려줘.", stream=True):
    ...:     print(reply.text, end="", flush=True)
    ...: print()
    ...: print(reply.usage)
    ...:
대전에는 다양한 매력을 가진 여행지가 많습니다. 다음은 추천할 만한 세 곳
입니다:

1. **대전 엑스포 과학공원**: 1993년 대전 엑스포가 개최된 장소로, 현재는
과학과 기술을 주제로 한 다양한 전시와 체험 프로그램이 운영되고 있습니다.
 특히, 대전시립과학관과 대전컨벤션센터가 있어 가족 단위 방문객에게 적합
합니다.

2. **유성온천**: 대전의 유명한 온천 지역으로, 다양한 온천 시설이 있습니
다. 피로를 풀고 휴식을 취하기 좋은 곳이며, 주변에 맛집
```

선택 지원

OpenAI/Upstage API에서는 Structured Output 활용하여, 보다 견고한 선택 지원

```python
from pyhub.llm import OpenAILLM

llm = OpenAILLM()

reply = llm.ask("과일을 골라줘.", choices=["apple", "car"])
print(repr(reply))
print(reply.choice)        # Optional[str] : 선택할 수 없다면 None
print(reply.choice_index)  # Optional[int] : 선택할 수 없다면 None
print(reply.usage)
```

```
Reply(text='{"choice":"apple","confidence":0.85}', usage=Usage(input=65, output=11), choice='apple', choice_index=0, confidence=0.85)
apple
0
Usage(input=65, output=11)
```

프롬프트 템플릿 지원

캐싱, 템플릿, 모델 등의 장고 기능을 활용할 때에는 pyhub.init() 호출 필요

```python
from pyhub.llm import OpenAILLM

# 프롬프트 템플릿 지원
llm = OpenAILLM(system_prompt="너는 유능한 여행 가이드",
                prompt="{여행지}의 맛집을 10개 소개해줘")
reply = llm.ask({"여행지": "대전"})
print(reply.text[:100], "...")
print(reply.usage)
```

```
대전은 다양한 맛집이 있는 도시로, 지역 특색을 살린 음식부터 다양한 세계 요리까지 즐길 수 있습니다. 아래는 대전에서 추천하는 맛집 10곳입니다.

1. **대전 한우촌** ...
Usage(input=31, output=565)
```

```python
from pyhub import init
from pyhub.llm import OpenAILLM

init()

# 프롬프트 템플릿 지원
llm = OpenAILLM(
    system_prompt="너는 유능한 여행 가이드",
    prompt="""
다음은 맛집 {{ 맛집 }}의 대표 메뉴.

{% for 메뉴 in 메뉴_리스트 %}
- {{ 메뉴.이름 }}
{% endfor %}

맛집 소개 블로그 글을 써줘.
    """,
)

reply_generator = llm.ask({
    "맛집": "성심당",
    "메뉴_리스트": [
        {"이름": "튀김소보로" },
        {"이름": "튀소구마" },
        {"이름": "판타롱 부추빵" },
        {"이름": "시루 시리즈" },
        {"이름": "명란바게트" },
        {"이름": "보문산메아리" },
    ],
}, stream=True)

for reply in reply_generator:
    print(reply.text, end="", flush=True)
print()
print(reply.usage)
```

장고 템플릿 문법을 사용하기 위해, 호출 필요

파일 업로드 지원

Vision Language Model API 지정 필요 (활용 예: 이미지 설명)

```python
from pyhub.llm import OpenAILLM

llm = OpenAILLM()  # 디폴트 : gpt-4o-mini


from pathlib import Path
from typing import Union
from django.core.files import File


# pyhub.llm describe 명령과 동일
files: list[Union[str, Path, File]] = [
    # "./sample1.jpg",  # 파일경로/URL 문자열 및 Path 객체, 장고 File 객체 지원
    "https://raw.githubusercontent.com/pyhub-kr/dump-data/refs/heads/main/sample1.jpg",
]
reply = llm.ask("Explain this image in korean", files=files)
print(reply)
```

LLM마다 요청 방법이 달라요.
- 내부에서 base64 인코딩 자동 변환 등

```
이 이미지는 바다와 해안의 풍경을 보여줍니다. 앞쪽에는 콘크리트 구조물들이 보이고, 그 뒤로는 파도가 부서지는 모습이 있습니다. 수평선 너머에는 산과 건물들이 보이며, 맑고 푸른 하늘이 펼쳐져 있습니다. 전체적으로 평화롭고 아름다운 해변의 분위기를 느낄 수 있습니다.
```

임베딩 지원

Anthropic를 제외한 OpenAI, Google, Upstage, Ollama 지원.

```python
# 임베딩 지원 (단일 문자열)
#  - OpenAILLM 디폴트 임베딩 모델 : text-embeddding-3-small
embeded = llm.embed("문서 내용 1")
print(len(embeded.array), "차원")
print(embeded.array[:3], "...")
print(embeded.usage)
```

```
1536 차원
[0.008741038851439953, 0.06660456210374832, -0.004017101135104895] ...
Usage(input=6, output=0)
```

```python
# 임베딩 지원 (단일 문자열)
embeded = llm.embed(["문서 내용 2", "문서 내용 3"])
print(len(embeded.arrays), "개")
print(embeded.arrays[0][:3], "...")
print(embeded.arrays[1][:3], "...")
print(embeded.usage)
```

```
2 개
[0.018519340083003044, 0.06718852370977402, -0.0009568605455569923] ...
[-0.005201217718422413, 0.07203260064125061, -0.001449519651941955] ...
Usage(input=12, output=0)
```

임베딩 및 벡터 스토어에 저장

Postgres 데이터베이스 + pgvector 확장

supabase 서비스에서도 pgvector 확장 지원 (무료 사용 가능)

```
# 데이터베이스 생성
docker run \
    -e POSTGRES_USER=djangouser \
    -e POSTGRES_PASSWORD=djangopw \
    -e POSTGRES_DB=djangodb \
    -p 5432:5432 \
    -d \
    pgvector/pgvector:pg17
```

```
# .env 포맷
DATABASE_URL=postgresql://djangouser:djangopw@localhost:5432/djangodb
```

```
# requirements.txt

# 필수
django
pgvector
psycopg2-binary
django-pyhub-rag[all]

# 옵션
django-environ
django-extensions
```

장고 프로젝트 샘플 settings

프로젝트 생성 및 myrag 앱 생성 필요

```python
# mysite/settings.py

from pathlib import Path
from environ import Env

BASE_DIR = Path(__file__).resolve().parent.parent

env = Env()

env_path = BASE_DIR / ".env"
if env_path.is_file():
    env.read_env(env_path, overwrite=True)

# ...

INSTALLED_APPS = [
    # ...
    "pyhub.rag",  # django-pyhub-rag 앱
    "myrag",      # myrag 앱 생성 후에 추가
    "django_extensions",
]
```

```python
# ...

DATABASES = {
    "default": env.db(),  # DATABASE_URL 환경변수 파싱
}

# ...

CACHES = {
    "default": {
        "BACKEND": "django.core.cache.backends.locmem.LocMemCache",
        "LOCATION": "caches/default",
    },
    # LLM API를 위한 캐시 설정 (google, anthropic, upstage 지원)
    "openai": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": BASE_DIR / "caches/openai",  # 절대경로
        "TIMEOUT": 86400 * 30,
        "OPTIONS": {
            "MAX_ENTRIES": 5000,
            # 최대치에 도달했을 때 삭제하는 비율 : 1/5 삭제
            #  - 0 : 모두 삭제
            "CULL_FREQUENCY": 5,
        },
    },
    # "anthropic", "google", "upstage", "ollama"
}
```

pgvector-python 라이브러리에서 django 모델 지원

벡터 필드만 추가하면 기존 프로젝트/서비스에서 OK. → 운영 단순화

```python
# myrag/models.py

from django.db import models
from pgvector.django import VectorField
from pgvector.django.indexes import HnswIndex

class VectorDocument(models.Model):
    # category = models.ForeignKey("Category",
    #     on_delete=models.CASCADE)

    page_content = models.TextField()
    metadata = models.JSONField(default=dict)
    embedding = VectorField(dimensions=1536)

    class Meta:
        indexes = [
            HnswIndex(
                # unique in db
                name='myrag_vd_emb_idx',
                fields=['embedding'],
                m=16,
                ef_construction=64,
                opclasses=['vector_consine_ops']
            ),
        ]
```

```python
# myrag/migrations/파일.py

from pgvector.django import VectorExtension

class Migration(migrations.Migration):
    operations = [
        VectorExtension()  # vector 확장 활성화
        # ...
    ]
```

```
-- 수행 SQL
CREATE EXTENSION vector;
```

```
-- 생성되는 SQL
CREATE TABLE "myrag_vectordocument"
(
    "id"           bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
    "page_content" text   NOT NULL,
    "metadata"     jsonb  NOT NULL,
    "embedding"    vector(1536) NOT NULL
);
```

```
VectorExtension() 을 수행하지 않았을 경우, migrate 시에 아래 오류 발생
    return self.cursor.execute(sql)
           ~~~~~~~~~~~~~~~~~~~~~~~^^^^
django.db.utils.ProgrammingError: type "vector" does not exist
LINE 1: ... NOT NULL, "metadata" jsonb NOT NULL, "embedding" vector(153...
                                                             ^
```

```python
# 유사 문서 검색 (by 코사인 유사도)

from pgvector.django import CosineDistance


def similarity_search(query: str, k: int = 4):
    # TODO: query 문자열에 대한 임베딩 벡터 생성
    query_embedding: list[float] = [0.91231231, ...]

    qs = VectorDocument.objects.annotate(
        distance=CosineDistance("embedding", query_embedding),
    )
    qs = qs.order_by("distance")
    return qs
```

관계형 데이터와 벡터 데이터를 단일 쿼리로 처리.
- 카테고리 필터링과 벡터 유사도 검색을 단일 쿼리셋으로 처리
- 자연스러운 JOIN 연산. (select_related, prefetch_related ok)

참고
- EthicalAds에서 사용자 추적없는 광고 타겟팅
- (supabase) pgvector vs Pinecone: cost and performance

공식문서 : https://github.com/pgvector/pgvector-python?tab=readme-ov-file#django

django-pyhub-rag 활용

벡터 모델 지원 + 임베딩 지원 + LLM 지원

```python
# myrag/models.py

from django.db import models
from pyhub.rag.models.postgres import PGVectorDocument


# 디폴트로 page_content, metadata, embedding 필드 정의
#  - embeding 필드 재정의 : 임베딩 차원 수, 임베딩 모델 변경이 필요할 때
class VectorDocument(PGVectorDocument):
    # category = models.ForeignKey("Category",
    #     on_delete=models.CASCADE)

    class Meta:
        indexes = [
            PGVectorDocument.make_hnsw_index(
                "myrag_vd_emb_idx"
            ),
        ]
```

```python
# myrag/migrations/0001_initial.py

from pgvector.django import VectorExtension

class Migration(migrations.Migration):
    operations = [
        VectorExtension()  # vector 확장 활성화
        # ...
    ]
```

```python
# 유사 문서 검색 (by 코사인 유사도) - 디폴트 모델 매니저를 통한 지원

qs = VectorDocument.objects.all()  # 필요한 만큼 조건을 추가한 뒤에
docs = qs.similarity_search("유사문서 찾을 내용")
# docs = await qs.similarity_search_async("유사문서 찾을 내용")
```

관계형 데이터와 베겉 데이터를 단일 쿼리로 처리.
- 카테고리 필터링과 벡터 유사도 검색을 단일 쿼리셋으로 처리
- 자연스러운 JOIN 연산. (select_related, prefetch_related ok)

참고
- EthicalAds에서 사용자 추적없는 광고 타겟팅
- (supabase) pgvector vs Pinecone: cost and performance

공식문서 : https://github.com/pgvector/pgvector-python?tab=readme-ov-file#django

임베딩 + 벡터 스토어에 저장

+ page_content 필드가 변경되면, 자동으로 embedding 필드 업데이트 수행

방법 1) pyhub.rag 장고 관리 명령을 통한 임베딩 + 벡터 스토어 저장

```
❯ python manage.py load_jsonl myrag.VectorDocument ./hyundai.jsonl
Created final batch of 84 instances
Successfully created 84 instances in total
```

방법 2) 쿼리셋을 통한 직접 임베딩 + 벡터 스토어 저장

```python
# $ python manage.py shell_plus --print-sql

import json
from myrag.models import VectorDocument

jsonl_path = "./hyundai.jsonl"

vector_document_list = []

for line in open(jsonl_path, "rt", encoding="utf-8"):
    line = line.strip()
    if line:
        obj = json.loads(line)
        # embedding 필드 값이 없는 인스턴스에 대해서는
        # 모아서 bulk embedding API 요청
        vector_document = VectorDocument(
            page_content=obj["page_content"],
            metadata=obj["metadata"],
        )
        vector_document_list.append(vector_document)

VectorDocument.objects.bulk_create(
    vector_document_list,
    batch_size=1000,
)
```

저장된 레코드 개수 확인 및 임베딩 필드 확인

```
>>> VectorDocument.objects.first().embedding
SELECT "myrag_vectordocument"."id",
       "myrag_vectordocument"."page_content",
       "myrag_vectordocument"."metadata",
       "myrag_vectordocument"."embedding"
  FROM "myrag_vectordocument"
 ORDER BY "myrag_vectordocument"."id" ASC
 LIMIT 1

Execution time: 0.004360s [Database: default]
array([ 0.06488528,  0.03795511, -0.01112608, ...,  0.00326196,
        0.01925571, -0.00921062], shape=(1536,), dtype=float32)
>>> VectorDocument.objects.all().count()
SELECT COUNT(*) AS "__count"
  FROM "myrag_vectordocument"

Execution time: 0.013919s [Database: default]
84
```

유사 문서 검색 및 RAG 채팅

유사 문서 검색

embedding 필드에 지정된 LLM 모델을 활용하여 자동으로 쿼리 임베딩 생성한 후에, 유사 문서 검색

[django-extensions] shell_plus --print-sql 명령으로 쉘 구동

```
>>> docs = VectorDocument.objects.similarity_search("등받이 각도")
>>> for doc in docs:
...     print(doc.distance, doc.page_content[:100] + '...')
...

SELECT "myrag_vectordocument"."id",
       "myrag_vectordocument"."page_content",
       "myrag_vectordocument"."metadata",
       ("myrag_vectordocument"."embedding" <=> '[0.02761758863925934,
           -0.014913497492671013,
           0.03034409135580063,

Execution time: 0.004041s [Database: default]
```

```
0.5975217401242773 3

# 등받이 각도 조절(2)(전동식)

등받이 각도 조절 스위치(2) 윗부분을 앞으로 당기면 등받이가 앞으로 숙여지고, 뒤로 당기면 등
받이가 뒤로 젖혀집니다.

# 등받이 접...
0.6264605820178986 3

- 2: 등받이 각도 조절
- 3: 좌석 높낮이 조절(사양 적용 시)
- 4: 쿠션 각도 조절(사양 적용 시)
- 5: 다리 받침대 조절 스위치(사양 적용 시)
- 6: 릴렉...
0.6310898817884755 # 시트 및 안전 장치

등받이 각도 조절 스위치 윗부분을 앞으로 당기면 등받이가 앞으로 숙여지고, 뒤로 밀면 등받이가
뒤로 젖혀집니다.

# 쿠션 각도/높낮이 조절하기

![im...
0.6466300767627413 3

전동식

![image](p021/02-figure.jpg)
2
4

- 1 등받이 각도 조절 레버 또는 스위치(1)로 좌석 등받이를 뒤로 젖히십시오(2).
- 2 헤드레스트...
```

RAG 채팅 비교 (1/2)

django-pyhub-rag를 통해 LLM 채팅 구현

```python
# 별도 파이썬 스크립트 구동 시에 장고 프로젝트 초기 로딩
import os
from typing import Generator
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')
import django; django.setup()

# 장고 프로젝트 초기 로딩 후에 수행
from pyhub.llm import OpenAILLM, AnthropicLLM, UpstageLLM, GoogleLLM
from myrag.models import VectorDocument


class ChatLLM:
    def __init__(self):
        system_prompt = """현대 산타페 자동차 매뉴얼을 기반으로 정확하게 답변하세요.
                           매뉴얼에 없는 내용은 모른다고 대답하세요. 사실과 의견을 구별해서 대답하세요."""
#       self.llm = OpenAILLM(system_prompt=system_prompt, model="gpt-4o",
#                            temperature=0.7, max_tokens=8192)
        self.llm = UpstageLLM(system_prompt=system_prompt, model="solar-pro2-preview",
                              temperature=0.7, max_tokens=8192)

    def ask(self, user_input: str) -> Generator[str, None, None]:
        for reply in self.llm.ask(user_input, stream=True):
            yield reply.text
```

필요한 LLM/모델 활용

랭체인과 연동할 경우

```python
# 별도 파이썬 스크립트 구동 시에 장고 프로젝트 초기 로딩
import os
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'mysite.settings')
import django; django.setup()

# 장고 프로젝트 초기 로딩 후에 수행
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_core.output_parsers import StrOutputParser
from langchain_core.chat_history import InMemoryChatMessageHistory
from langchain_openai import ChatOpenAI
from langchain_google_genai import ChatGoogleGenerativeAI

from myrag.models import VectorDocument

class ChatLLM:
    def __init__(self):
        llm = ChatOpenAI(model="gpt-4o", temperature=0.7, streaming=True, max_tokens=8192)
        prompt = ChatPromptTemplate.from_messages([
            ("system", """현대 산타페 자동차 매뉴얼을 기반으로 정확하게 답변하세요.
                          매뉴얼에 없는 내용은 모른다고 대답하세요. 사실과 의견을 구별해서 대답하세요."""),
            MessagesPlaceholder(variable_name="chat_history"),
            ("human", "{input}")
        ])
        self.history = InMemoryChatMessageHistory()
        self.chain = prompt | llm

    def ask(self, user_input: str) -> Generator[str, None, None]:
        stream = self.chain.stream({
            "input": user_input,
            "chat_history": self.history.messages,
        })

        full_response = ""
        for chunk in stream:
            yield chunk.content
            full_response += chunk.content

        # Add to chat history - store original user input, not the RAG context
        self.history.add_user_message(user_input)
        self.history.add_ai_message(full_response)
```

RAG 채팅 비교 (2/2)

django-pyhub-rag를 통해 LLM 채팅 구현

```python
class RAGDecisionLLM:
    def __init__(self):
        system_prompt = """
            사용자 질문이 '현대 산타페 자동차 매뉴얼'의 내용을 기반으로 검색(RAG)이 필요한지 판단하세요.
            - 매뉴얼에 나올 법한 질문이면 "RAG"
            - 일반적인 상식이나 매뉴얼과 무관한 질문이면 "NO-RAG
        """
        self.llm = GoogleLLM(system_prompt=system_prompt, model="gemini-2.0-flash-lite")

    def should_use_rag(self, question: str) -> bool:
        reply = self.llm.ask(
            input=question,
            # choices 인자가 지정되면 강제로 temperature=0.1로 변경
            # openai llm에서는 Structured Outputs 방식으로 동작
            # https://platform.openai.com/docs/guides/structured-outputs?api-mode=chat
            choices=["RAG", "NO-RAG"],
            use_history=False,
        )
        # return reply.choice == "RAG"
        return reply.choice_index == 0
```

랭체인과 연동할 경우

```python
class RAGDecisionLLM:
    def __init__(self):
        system_prompt = """
            사용자 질문이 '현대 산타페 자동차 매뉴얼'의 내용을 기반으로 검색(RAG)이 필요한지 판단하세요.
            - 매뉴얼에 나올 법한 질문이면 "RAG"
            - 일반적인 상식이나 매뉴얼과 무관한 질문이면 "NO-RAG
        """
        llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash-lite", temperature=0.1)
        prompt = ChatPromptTemplate.from_messages([
            ("system", system_prompt),
            ("human", "Question: {question}")
        ])
        self.chain = prompt | llm | StrOutputParser()

    def should_use_rag(self, question: str) -> bool:
        result = self.chain.invoke({"question": question})
        choice = result.strip()
        return choice == "RAG"
```

공통

```python
def main():
    print("🤖 현대 산타페 AI 어시스턴트")

    chat_llm = ChatLLM()
    rag_decision_llm = RAGDecisionLLM()

    while True:
        try:
            user_input = input(">>> ").strip()
            if not user_input: continue

            is_rag = rag_decision_llm.should_use_rag(user_input)

            if is_rag:
                print("INFO: 유사 문서 검색 중 ...")
                docs = VectorDocument.objects.similarity_search(user_input)

                # 출처 정보 추출
                sources = []
                for doc in docs[:5]:  # 상위 5개 문서의 출처만 표시
                    if hasattr(doc, 'metadata') and doc.metadata:
                        source = doc.metadata.get('source', '?')
                        page = doc.metadata.get('page', '?')
                        sources.append(f"{source}/p.{page}")
                    else:
                        sources.append("?/p.?")

                human_content = f"## Context\n{docs}\n\n##Question#{user_input}"
            else:
                human_content = user_input
                sources = []

            for reply in chat_llm.ask(human_content):
                print(reply, end="", flush=True)

            # RAG 사용 시 출처 표시
            if is_rag and sources:
                print("\n\n📄 출처:")
                for source in sources:
                    print(f"  - {source}")
            print()

        except (KeyboardInterrupt, EOFError):
            print('\n\n👋 Goodbye!')
            break


if __name__ == '__main__':
    main()
```

요약

서비스 준비 단계

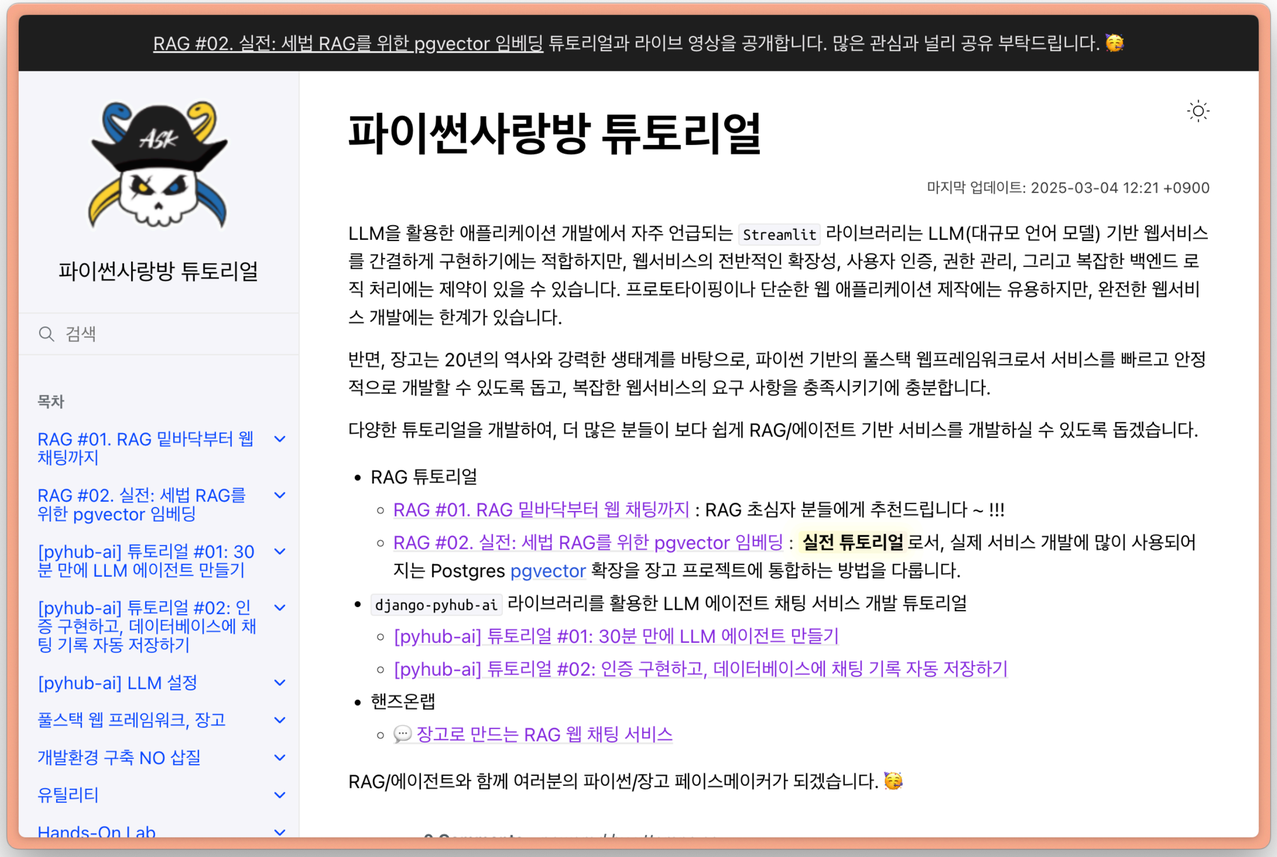
```
# 1) 라이브러리 설치
pip install -U 'django-pyhub-rag[all]'

# 2) API Key 설정 (.env)
UPSTAGE_API_KEY=...
OPENAI_API_KEY=...

# 3) Document Parse API를 통해 jsonl 파일 생성
python -m pyhub.parser upstage -i ./doc.pdf  # pptx 등

# 4) VectorDocument 모델 생성 및 마이그레이션

# 5) 지정 VectorDocument로 데이터 저장 및 임베딩
python manage.py load_jsonl myrag.VectorDocument ./output/doc.jsonl
```

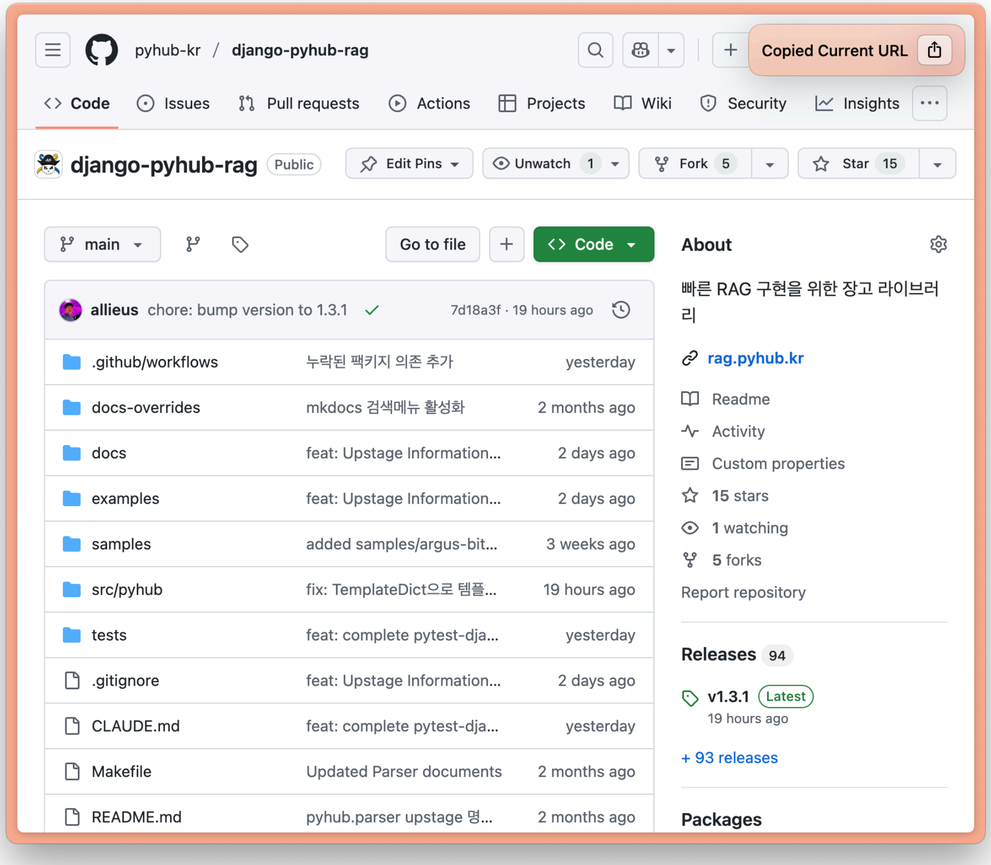
RAG 서비스

```
from myrag.models import VectorDocument

# 유사 문서 검색
user_input = "..."
docs = VectorDocument.objects.similarity_search(user_input)
human_content = f"## Context\n{docs}\n\n##Question#{user_input}"

from pyhub.llm import OpenAILLM

llm = OpenAILLM(model="gpt-4o")
reply = llm.ask(human_content)
print(reply.text)
print(reply.usage)
```

기반 지식을 익히실려면?

다양한 의견 부탁드립니다. :-)





ai.pyhub.kr

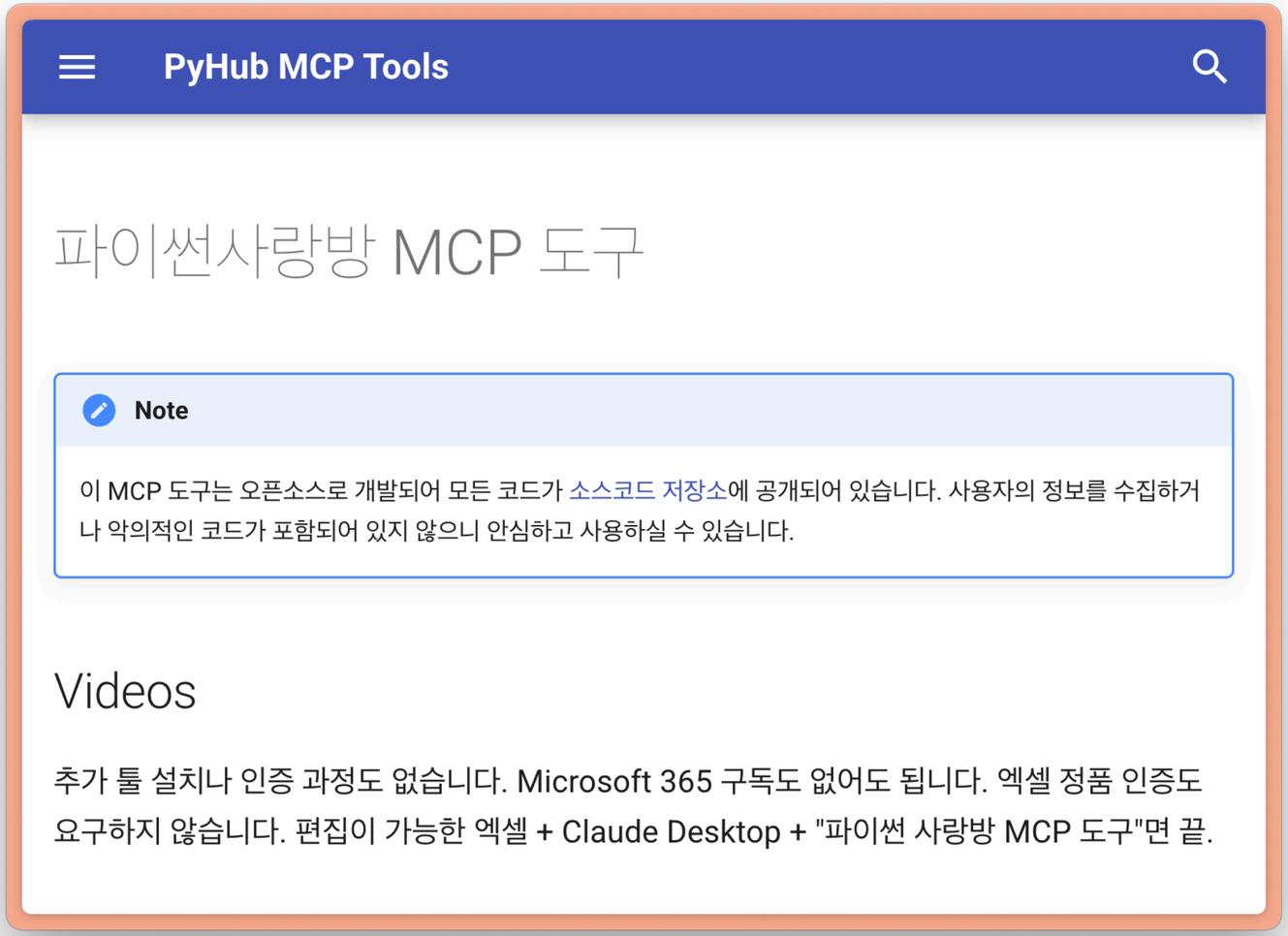
인생은 짧아요. 파이썬/장고를 쓰세요. ;-)

자매품

파이썬사랑방 MCP 도구

- 로컬 엑셀 MCP 도구
  - 엑셀을 함께 보며, LLM과 협업 가능
- 검증 단계 도구
  - 아웃룩 메일 읽기/발송 도구
  - 애플 메일/메시지/캘린더 도구
  - 감정 분석 도구
  - 네이버 지도 길찾기 도구
  - 파이썬 코드 인터프리터 도구
- 준비 중
  - Function Calling
  - MCP Client

mcp.pyhub.kr



인생은 짧아요.

파이썬/장고와 함께 시간을 아끼세요.

감사합니다. ;-)