LangChain 활용한 데이터 가져오기

LangChain 라이브러리 활용
Reference : https://python.langchain.com/docs/modules/data_connection/document_loaders/
라이브러리 설치
WebBaseLoader 활용한 데이터 가져오기
streamlit 페이지 정보 가져오기
여러 페이지의 정보를 가져오기
streamlit 여러 페이지 정보 가져오기
가져온 정보를 텍스트로 저장
streamlit페이지 전체 튜토리얼 링크 확인
결과는 다음과 같다.
WebBaseLoader을 활용한 각 링크의 정보를 가져와 Text 파일로 저장
결과
text 파일을 불러와 각 텍스트를 ChatGPT API 활용하여 요약 정리

LangChain 라이브러리 활용

Reference : https://python.langchain.com/docs/modules/data_connection/document_loade

라이브러리 설치

```
!pip install langchain
```

WebBaseLoader 활용한 데이터 가져오기

```
from langchain.document_loaders import WebBaseLoader
```

```
loader = WebBaseLoader("https://www.espn.com/")
data = loader.load()
data
```

streamlit 페이지 정보 가져오기

```
### Streamlit 페이지 가져와 보기
# https://docs.streamlit.io/library/api-reference/write-magic/st.write
loader = WebBaseLoader("https://docs.streamlit.io/library/api-reference/write-magic/st.write"
data = loader.load()
data
```

위와 같이 하나의 페이지의 정보를 가져오는 것이 가능하다.

그렇다면 링크가 많이 모여있는 상위 페이지에서 링크를 가져오는 프로그램을 짜본다.

여러 페이지의 정보를 가져오기

```
# 여러 URL을 같이 load하기
loader = WebBaseLoader(["https://ko.wikipedia.org/wiki/%EB%8C%80%ED%98%95_%EC%96%B8%EC%96%
B4_%EB%AA%A8%EB%8D%B8", "https://google.com"])
docs = loader.load()
docs
```

streamlit 여러 페이지 정보 가져오기

```
### 몇개의 Streamlit link 데이터를 다운로드 받기.
links = [ 'https://docs.streamlit.io/library/api-reference/write-magic/st.write',
          'https://docs.streamlit.io/library/api-reference/write-magic/magic' ]


for link in links:
  loader = WebBaseLoader(link)
  data = loader.load()
  content = data[0].page_content
  print( len(content) )
  print( content[0:500] )
```

가져온 정보를 텍스트로 저장

```
### 몇개의 Streamlit link 데이터를 다운로드 받기.
links = [ 'https://docs.streamlit.io/library/api-reference/write-magic/st.write',
          'https://docs.streamlit.io/library/api-reference/write-magic/magic' ]


for link in links:
  loader = WebBaseLoader(link)
  data = loader.load()
  content = data[0].page_content
  print( len(content) )
  # 파일명을 링크의 일부를 기반으로 생성
```

```
  file_name = link.split('/')[-1] + '.txt'
  # 웹페이지 데이터를 파일로 저장
  with open(file_name, 'w', encoding='utf-8') as file:
    file.write(content)
  print(f'페이지 내용을 {file_name} 파일로 저장했습니다.')
```

streamlit페이지 전체 튜토리얼 링크 확인

```
### 판다스를 이용하여 링크를 가져와 이를 이용하여 결과를 만들어낸다.
import os
import pandas as pd

# CSV 파일 경로
file_path = 'result.csv'

# CSV 파일 읽기
df = pd.read_csv(file_path)

links = df['Link'].to_list()

# 각 링크의 데이터를 다운로드하고 파일로 저장
for index, link in enumerate(links, start=1):
  print(link)
```

결과는 다음과 같다.

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

	Gubun	Link	Title	Description	Example Code
0	Write and Magic	https://docs.streamlit.io/library/api-referenc...	st.write	Write arguments to the app.	st.write("Hello **world**!")\nst.write(my_data...
1	Write and Magic	https://docs.streamlit.io/library/api-referenc...	Magic	Any time Streamlit sees either a variable or l...	st.write
2	Text elements	https://docs.streamlit.io/library/api-referenc...	Markdown	Display string formatted as Markdown.	st.markdown("Hello **world**!")\n
3	Text elements	https://docs.streamlit.io/library/api-referenc...	Title	Display text in title formatting.	st.title("The app title")\n
4	Text elements	https://docs.streamlit.io/library/api-referenc...	Header	Display text in header formatting.	st.header("This is a header")\n
5	Text elements	https://docs.streamlit.io/library/api-referenc...	Subheader	Display text in subheader formatting.	st.subheader("This is a subheader")\n
6	Text elements	https://docs.streamlit.io/library/api-referenc...	Caption	Display text in small font.	st.caption("This is written small caption text...
7	Text elements	https://docs.streamlit.io/library/api-referenc...	Code block	Display a code block with optional syntax high...	st.code("a = 1234")\n

WebBaseLoader을 활용한 각 링크의 정보를 가져와 Text 파일로 저장

```
### 판다스를 이용하여 링크를 가져와 이를 이용하여 결과를 만들어낸다.
import os
import pandas as pd

# CSV 파일 경로
file_path = 'result.csv'

# CSV 파일 읽기
```

```
df = pd.read_csv(file_path)

foldername = input("데이터저장 폴더는:")
# 데이터를 저장할 폴더 생성 (이미 존재하면 무시)
data_folder = foldername
if not os.path.exists(data_folder):
    os.mkdir(data_folder)

links = df['Link'].to_list()

# 각 링크의 데이터를 다운로드하고 파일로 저장
for index, link in enumerate(links, start=1):
  loader = WebBaseLoader(link)
  data = loader.load()
  content = data[0].page_content
  print( len(content) )
  # 파일명을 링크의 일부를 기반으로 생성
  file_name = f"{data_folder}/{index}_" + link.split('/')[-1] + ".txt"
  # 웹페이지 데이터를 파일로 저장
  with open(file_name, 'w', encoding='utf-8') as file:
    file.write(content)
  print(f'페이지 내용을 {file_name} 파일로 저장했습니다.')
```

결과

```
4529
페이지 내용을 data01/1_st.write.txt 파일로 저장했습니다.
2582
페이지 내용을 data01/2_magic.txt 파일로 저장했습니다.
3546
페이지 내용을 data01/3_st.markdown.txt 파일로 저장했습니다.
2741
페이지 내용을 data01/4_st.title.txt 파일로 저장했습니다.
3166
페이지 내용을 data01/5_st.header.txt 파일로 저장했습니다.
...
```

text 파일을 불러와 각 텍스트를 ChatGPT API 활용하여 요약 정리

최종 요약 결과는 .csv 파일로 정리.

```
!pip install openai

# 인증
import os
from openai import OpenAI

def init_api():
    with open("chatgpt.env") as env:
       for line in env:
           key, value = line.strip().split("=")
           os.environ[key] = value
```

```
init_api()
client = OpenAI(api_key  = os.environ.get("API_KEY"))

# 요약 정
## data 폴더의 파일의 모든 내용을 불러와 이를 출력하기
import os
import time

# 데이터를 저장할 폴더 설정
data_folder = 'data'

# 폴더 내의 모든 파일 목록을 가져옵니다.
file_list = os.listdir(data_folder)

# 각 파일에 대해
for file_name in file_list:
    file_path = os.path.join(data_folder, file_name)  # 파일의 전체 경로 생성

    # 파일인지 확인하고, 파일이면 내용을 읽어 출력합니다.
    if os.path.isfile(file_path):
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
            print(f'File: {file_name}\nContent:\n{content}\n{"=" * 20}\n')


## data 폴더의 파일의 모든 내용을 불러와 이를 출력하기
import os
import time

# 데이터를 저장할 폴더 설정
data_folder = 'data01'

# 폴더 내의 모든 파일 목록을 가져옵니다.
file_list = os.listdir(data_folder)

# 결과를 저장할 빈 DataFrame을 생성합니다.
df_results = pd.DataFrame(columns=['file_name', 'summary'])

# 각 파일에 대해
for file_name in file_list:
    file_path = os.path.join(data_folder, file_name)  # 파일의 전체 경로 생성

    # 파일인지 확인하고, 파일이면 내용을 읽어 출력합니다.
    if os.path.isfile(file_path):
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
            # print(f'File: {file_name}\nContent:\n{content}\n{"=" * 20}\n')

            next = client.chat.completions.create(
                    model="gpt-3.5-turbo-16k",
                    messages=[
                      {
                        "role": "system",
                        "content": "IT 가이드 설명 웹 페이지 내용인데, 이 내용을 1000자 이내로 요약을 부
                      },
                      {
                        "role": "user",
                        "content": content
```

```
                    }
                  ],
                  temperature=0,
                  max_tokens=1200,
                  top_p=1,
                  frequency_penalty=0,
                  presence_penalty=0
                )

            # 요약 결과를 DataFrame에 추가합니다.
            summary = next.choices[0].message.content
            df_results = df_results.append({'file_name': file_name, 'summary': summary}, ignc

    time.sleep(5)

# 결과 DataFrame을 CSV 파일로 저장합니다.
df_results.to_csv('summaries.csv', index=False)
```