Naver 뉴스 기사 크롤링 코드

필수 라이브러리 설치

```
# 필수 라이브러리 설치
!pip install selenium
```

```
%%shell
# Ubuntu no longer distributes chromium-browser outside of snap
#
# Proposed solution: https://askubuntu.com/questions/1204571/how-to-install-chromium-without-snap

# Add debian buster
cat > /etc/apt/sources.list.d/debian.list <<'EOF'
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster.gpg] http://deb.debian.org/debian buster main
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-buster-updates.gpg] http://deb.debian.org/debian buster-updates main
deb [arch=amd64 signed-by=/usr/share/keyrings/debian-security-buster.gpg] http://deb.debian.org/debian-security buster/updates
EOF

# Add keys
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys DCC9EFBF77E11517
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 648ACFD622F3D138
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 112695A0E562B32A

apt-key export 77E11517 | gpg --dearmour -o /usr/share/keyrings/debian-buster.gpg
apt-key export 22F3D138 | gpg --dearmour -o /usr/share/keyrings/debian-buster-updates.gpg
apt-key export E562B32A | gpg --dearmour -o /usr/share/keyrings/debian-security-buster.gpg

# Prefer debian repo for chromium* packages only
# Note the double-blank lines between entries
cat > /etc/apt/preferences.d/chromium.pref << 'EOF'
Package: *
Pin: release a=eoan
Pin-Priority: 500


Package: *
Pin: origin "deb.debian.org"
Pin-Priority: 300


Package: chromium*
Pin: origin "deb.debian.org"
Pin-Priority: 700
EOF

# Install chromium and chromium-driver
apt-get update
apt-get install chromium chromium-driver
```

라이브러리 import

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from bs4 import BeautifulSoup
import pandas as pd
import requests
import time
```

page scraping 함수

```
def safe_find_element(driver, by, value):
    try:
        return driver.find_element(by, value)
    except NoSuchElementException:
        return None
```

```
            return None

def news_scraping(news_url, driver):
    # 언론사
    press_element = safe_find_element(driver, By.XPATH, '//*[@id="ct"]/div[1]/div[1]/a/img[2]')
    press = press_element.get_attribute('title') if press_element else ""

    # 기사 제목
    title_element = safe_find_element(driver, By.ID, 'title_area')
    title = title_element.text if title_element else ""

    # 발행일자
    date_time_element = safe_find_element(driver, By.XPATH, '//*[@id="ct"]/div[1]/div[3]/div[1]/div/span')
    date_time = date_time_element.text if date_time_element else ""

    # 기자
    repoter_element = safe_find_element(driver, By.XPATH, '//*[@id="JOURNALIST_CARD_LIST"]/div[1]/div/div[1]/div/div/div[1]/a[2
    repoter = repoter_element.text if repoter_element else ""

    # 기사 본문
    article_element = safe_find_element(driver, By.ID, 'dic_area')
    article = article_element.text.replace("Wn", "").replace("Wt", "") if article_element else ""

    # 기사 반응: 쏠쏠정보
    useful_element = safe_find_element(driver, By.XPATH, '//*[@id="likeItCountViewDiv"]/ul/li[1]/a/span[2]')
    useful = useful_element.text if useful_element else ""

    # 기사 반응: 흥미진진
    wow_element = safe_find_element(driver, By.XPATH, '//*[@id="likeItCountViewDiv"]/ul/li[2]/a/span[2]')
    wow = wow_element.text if wow_element else ""

    # 기사 반응: 공감백배
    touched_element = safe_find_element(driver, By.XPATH, '//*[@id="likeItCountViewDiv"]/ul/li[3]/a/span[2]')
    touched = touched_element.text if touched_element else ""

    # 기사 반응: 분석탁월
    analytical_element = safe_find_element(driver, By.XPATH, '//*[@id="likeItCountViewDiv"]/ul/li[4]/a/span[2]')
    analytical = analytical_element.text if analytical_element else ""

    # 기사 반응: 후속강추
    recommend_element = safe_find_element(driver, By.XPATH, '//*[@id="likeItCountViewDiv"]/ul/li[5]/a/span[2]')
    recommend = recommend_element.text if recommend_element else ""

    print("뉴스:", [title, press, date_time, repoter, article, useful, wow, touched, analytical, recommend, news_url])

    return [title, press, date_time, repoter, article, useful, wow, touched, analytical, recommend, news_url]

def scraping(list_url):
    driver.implicitly_wait(3)

    news_idx = 1
    news_df = pd.DataFrame(columns = ("Title", "Press", "DateTime", "Repoter", "Article", "Useful", "Wow", "Touched", "Analytic

    for url in list_url:
        driver.get(url)
        news_df.loc[news_idx] = news_scraping(url, driver)
        news_idx += 1

    driver.close()

    return news_df
```

검색 키워드, 크롤링 페이지 설정

검색창에 검색하고자 하는 키워드 입력 하고, 수집할 페이지는 1에서 2 또는 3페이지 정도로 하시면 최신자료를 수집할 수 있습니다.

```
def make_pg_num(num):
    """Calculate the page number in the format required by the website."""
    return num if num == 1 else num+9*(num-1)

def create_url(search, page_num):
    """Create a URL with the search term and page number."""
    return f"https://search.naver.com/search.naver?where=news&sm=tab_pge&query={search}&sort=0&photo=0&field=0&pd=0&ds=&de=&clu

def make_urls(search, start_pg, end_pg):
```

```
    """Generate the URLs for the range of pages."""
    return [create_url(search, make_pg_num(i)) for i in range(start_pg, end_pg+1)]

def input_with_validation(prompt):
    """Ask for input with the given prompt, repeating until a valid integer is provided."""
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            print("Invalid input, please enter an integer.")

def main():
    search = input("검색 키워드를 입력해주세요: ")

    start_pg = input_with_validation("Wn크롤링 시작 페이지를 입력해주세요. ex)1(숫자만 입력): ")
    print(f"Wn크롤링 시작 페이지: {start_pg}페이지")

    end_pg = input_with_validation("Wn크롤링 종료 페이지를 입력해주세요. ex)1(숫자만 입력): ")
    print(f"Wn크롤링 종료 페이지: {end_pg}페이지")

    return make_urls(search, start_pg, end_pg)

if __name__ == "__main__":
    search_urls = main()
    print("생성된 URL: ", search_urls)

#Chrome drive option 설정
chrome_options = webdriver.ChromeOptions()

chrome_options.add_argument('--verbose')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
chrome_options.add_argument('--windows-size=1920, 1200')
chrome_options.add_argument('--disable-dev-shm-usage')

driver = webdriver.Chrome(options = chrome_options)
```

기사 링크 수집

```
# Initialize the list to store the links
list_url = []

# Iterate over the URLs
for url in search_urls:
    # Send GET request to the web page
    response = requests.get(url)

    # If the request is successful, extract the HTML content and create a BeautifulSoup object
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, "html.parser")
        links = soup.select("a.info, a.sub_txt")  # Select both "a.info" and "a.sub_txt" elements

        # Filter and save the links with "naver.com" in their address
        for link in links:
            href = link.get("href")
            if "naver.com" in href:
                list_url.append(href)
    else:
        print("The request failed.")

    # Sleep for 1 second
    time.sleep(2)
```

```
list_url
```

크롤링 실행

```
news_df = scraping(list_url)
```

```
news_df.to_excel("news.xlsx")
```

OpenAI API 연동

```
pip install openai
```

word를 다룰 수 있는 라이브러리 설치

```
pip install python-docx
```

GPT 기사 분석: 대화형

대화를 통해 자유롭게 묻고 답할 수 있다.
GPT에게 유능한 저널리스트이자 텍스트 분석 전문가라는 역할을 부여함.

```
import os
import openai
import pandas as pd
from docx import Document

# Assuming that you have the DataFrame 'news_df' already loaded
def ask_from_article(index):
    article = news_df['Article'][index]

    openai.api_key = "여기에 openai에서 발급받은 api key를 붙여넣기 하세요."

    # 역할 부여(유능한 기자이자, 텍스트 분석 전문가)
    messages = [
        {"role": "system", "content": "You are a very competent journalist and text analytics expert who needs to do the follow
        {"role": "user", "content": f"Here is an article: {article}"}
    ]

    while True:
        user_content = input("기사에 대한 질문을 입력하세요. : ")
        if user_content.lower() == "종료":
            break
        messages.append({"role" : "user", "content" : f"{user_content}"})

        completion = openai.ChatCompletion.create(
            model = "gpt-3.5-turbo",
            messages = messages
        )

        assistant_content = completion.choices[0].message["content"].strip()

        messages.append({"role" : "assistant", "content" : f"{assistant_content}"})

        print(f"GPT-3.5 Turbo : {assistant_content}")

    # Saving the conversation to a word file
    doc = Document()
    for message in messages:
        doc.add_paragraph(f"{message['role']} : {message['content']}")
    doc.save("대화기록.docx")
```

```
# 대화를 종료할 때는 "종료" 입력
# ask_from_article() 함수에 요약하고 싶은 기사의 번호를 괄호에 입력하고, Ctrl + Enter

ask_from_article(1)
```

GPT 기사 분석: 자동 반복

```
import os
import openai
import pandas as pd
import time
from docx import Document

def get_article_content(index):
```

```
    # Replace this with your own logic to retrieve the article content from `news_df`
    return news_df['Article'].iloc[index]

def summarize_article(index, doc):
    try:
        article = get_article_content(index)

        openai.api_key = "여기에 openai에서 발급받은 api key를 붙여넣기 하세요."

        # 역할 부여(유능한 기자이자, 텍스트 분석 전문가)
        messages = [
        {"role": "system", "content": "You are a very competent journalist and text analytics expert who needs to do the follov
        {"role": "user", "content": f"You should briefly summarize the article and write your answer in Korean. You should alsc
        ]

        completion = openai.ChatCompletion.create(
            model = "gpt-3.5-turbo",
            messages = messages
        )

        assistant_content = completion.choices[0].message["content"].strip()

        print(f"Summary of article {index}: {assistant_content}")

        # Adding the summary to the Word document
        doc.add_paragraph(f"Summary of article {index}: {assistant_content}")

    except Exception as e:
        print(f"Error processing article {index}: {str(e)}")

# Create the Word document
doc = Document()

# Call function for each article with a 20 second delay
for i in range(len(news_df['Article'])):
    summarize_article(i, doc)
    time.sleep(20)

# Save the Word document
doc.save("summaries.docx")
```

Colab 유료 제품 - 여기에서 계약 취소