# 필요한 필수 라이브러리 설치

!pip install selenium

# Colab환경에서 프로그램을 실행시키기 위한 조건 및 프로그램 설치

%%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 main

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

# 필요한 필수 라이브러리 불러오기

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

# 함수 설정

def safe\_find\_element(driver, by, value):

    try:

        return driver.find\_element(by, value)

    except NoSuchElementException:

        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]/span/span/em')

    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("\n", "").replace("\t", "") 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", "Analytical", "Recommend", "URL"))

    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

페이지 수집하실 때 원하시는 키워드를 입력하시면 됩니다. 미세먼지, 청년복지, 경기도북부특별자치도 등, 각 과에서 담당하시는 주요 키워드를 붙여넣기 하시면, 네이버 뉴스에 등록된 기사를 수집할 것입니다.

수집할 페이지 설정 후, 수집할 범위는 최신기사만 할 경우 보통은 1 ~ 2 페이지만 설정하시면 됩니다.

# 페이지 수집

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=&cluster\_rank=17&mynews=0&office\_type=0&office\_section\_code=0&news\_office\_checked=&nso=so:r,p:all,a:all&start={page\_num}"

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("\n크롤링 시작 페이지를 입력해주세요. ex)1(숫자만 입력): ")

    print(f"\n크롤링 시작 페이지: {start\_pg}페이지")

    end\_pg = input\_with\_validation("\n크롤링 종료 페이지를 입력해주세요. ex)1(숫자만 입력): ")

    print(f"\n크롤링 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)

네이버 뉴스를 검색하여 기사의 링크를 수집하는 작업을 진행합니다.

# 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)

colab의 좌측에 보시면 폴더 모양이 있습니다. 폴더를 클릭하시거나, 폴더 새로고침 버튼을 클릭하시면, news.xlsx라는 파일이 생성되어 있을 겁니다.

news\_df = scraping(list\_url)

news\_df.to\_excel("news.xlsx")

OpenAi API키를 발급받으셔야 합니다.

발급 받은 키를 “여기에 openai 에서 발급받은 api key를 붙여넣기 하세요. 에 붙여넣으시면 됩니다.

pip install openai

pip install python-docx

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 following."},

        {"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)

아래의 코드는 실행시키시면 자동으로 텍스트를 분석하고 결과를 word파일로 생성하게 됩니다.

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 following."},

        {"role": "user", "content": f"You should briefly summarize the article and write your answer in Korean. You should also create a positive/negative: and indicate positive if the article is positive, and negative if it is negative.: {article}"}

        ]

        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")