Python Programming and Practice

# Development daily news keyword summary program

Final Report

Date : 2023.12.24

Name : 지연우

ID : 224491

# 1. Introduction

## 1) Background

It's time consuming and very cumbersome to check out all the tons of news pouring out every day. But we can't completely stop paying attention to current events, so we need a more efficient way to get information. To solve this problem, a program that can easily check current events information in short sentences or keywords is needed.

## 2) Project goal

Aim to create a program that summarizes articles uploaded every day in short sentences and a few words.

## 3) Differences from existing programs

The article summary function previously provided by Naver News only summarizes each article. This, of course, helps save time, but it is difficult to easily find out various current events information. We analyze all articles to show the user a few key sentences or keywords, making it easy to get current events information.

# 2. Functional Requirement

## 1) Collect articles

- function to collect articles uploaded daily

### (1) Collect information on popular articles by media company

- Crawl the titles, body text of the top five popular articles of each media company

## 2) Summarize the articles

- Summarize each article in short sentences and words.

### (1) Create article summary

- Summarize the article in one sentence.

### (2) Extract keywords

- Extract one keyword of articles.

## 3) Show keywords

- Show keywords to user.

### (1) Sort keywords in order of frequency

- Show the keywords that appear in more articles first.

## 4) Show summary of articles by keyword

- When a user selects a keyword, it shows a summary of the article in which that keyword appears. And when a user enters article number, it shows article body text.

# 3. Implementation

## (1) Collect articles

- Input: Link to Naver News Ranking page

- Output: TSV file containing information of the articles

- Description: Crawl the titles, body text and url of the top five popular articles from each media company.

- Applied concepts:  loops, functions, string manipulation, exception handling, file write, __name__ == "__main__", package, web crawling

- Code Screenshot

```python
from datetime import datetime
import os.path

import requests
from bs4 import BeautifulSoup as bs


# 주어진 링크의 html파일을 lxml로 파싱한 BeautifulSoup 객체를 반환하는 함수
2 usages  Yeonwoo Jl
def get_soup(link):
    USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
    head = {"user-agent": USER_AGENT}
    req = requests.get(link, headers=head)
    html = req.text
    soup = bs(html, features: "lxml")
    return soup


# 링크가 주어진 기사의 정보를 리스트로 만들어 반환하는 함수
# [기사제목, 기사링크, 기사본문] 형태로 반환
1 usage  Yeonwoo Jl +1
def crawl_article(article_link):
    soup = get_soup(article_link)

    title = soup.select_one("#title_area > span").text
    body_text = soup.select_one("#dic_area").text.strip().replace( __old: "\n", __new: " ").replace( __old: '\t', __new: ' ')

    return [title, article_link, body_text]
# 네이버 뉴스의 랭킹뉴스 페이지에서 언론사별 조회수 상위 5개 기사의 정보를 수집하는 함수
# 각 기사에 대해 crawl_article 함수를 사용하여 기사 정보를 얻어오고, tsv파일에 저장
3 usages  Yeonwoo Jl +1
def crawl_rankings(file_name):
    if(os.path.isfile(file_name)):
      return
    rankings_link = "https://news.naver.com/main/ranking/popularDay.naver"
    soup = get_soup(rankings_link)
    rankingnews_boxes = soup.select(".rankingnews_box")  # 언론사별 랭킹 5위까지의 기사 정보가 담긴 박스들 선택

    with open(file_name, 'w', encoding="utf8") as fp:
        fp.write("언론사명\t기사제목\t기사링크\t기사본문\n")

        for box in rankingnews_boxes:
            company_name = box.select_one(".rankingnews_name").text
            # 영어 기사는 제외하도록 영문 신문은 수집하지 않음
            if company_name in ("코리아헤럴드", "코리아중앙데일리"):
                continue
            articles = box.select("li")

            for i, article in enumerate(articles):
                try:
                    # 링크는 https://n.news.naver.com/article/언론사코드/기사번호?ntype=RANKGIN 형태
                    article_link = article.select_one(".list_title")["href"]
                    article_data = crawl_article(article_link)
                    fp.write(company_name + "\t" + "\t".join(article_data) + "\n")
                # 집계기준에 해당하는 기사가 없어 랭킹 5위까지 기사가 존재하지 않을 때
                except:
                    print(company_name, i+1, "번째 기사 수집 실패")
```

## (2) Summarize the articles

- Input: Crawled articles data

- Output: Save TSV file containing information of the articles, summaries of the articles and keywords of the articles. And Return pandas dataframe of that file and a dictionary with a keyword extracted from the articles as a key and a list of indexes of the sentence in which the keyword appeared as a value

- Description: Summarizes articles using t5 models and calculates the frequency of keyword appearance by extracting keywords extracted by keybert.

- Applied concepts:  loops, functions, list comprehension, file read, file write, dictionary, pandas, keyword extraction, text summarize, package

- Code Screenshot

```python
import os.path

import nltk
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from konlpy.tag import Hannanum
from keybert import KeyBERT
import pandas as pd

# https://huggingface.co/eenzeenee/t5-base-korean-summarization t5기반 한국어 문서 요약 모델을 사용하여 기사들의 요약문 추출
# 기사 10개씩 나누어서 요약하도록 설정
1 usage  ± speciling
def get_summarize(articles):
    nltk.download('punkt')

    model = AutoModelForSeq2SeqLM.from_pretrained('eenzeenee/t5-base-korean-summarization').to('cuda')
    tokenizer = AutoTokenizer.from_pretrained('eenzeenee/t5-base-korean-summarization')
    result = []

    for i in range(0, len(articles), 30):
        inputs = ["summarize: " + article for article in articles[i:min(i+30, len(articles))]]
        inputs = tokenizer(inputs, max_length=512, truncation=True, return_tensors="pt", padding=True).to('cuda')
        output = model.generate(**inputs, num_beams=3, do_sample=True, min_length=10, max_length=100)
        decoded_output = tokenizer.batch_decode(output, skip_special_tokens=True)
        result += decoded_output

    return result
# 요약문에서 불용어가 아닌 명사만을 추출하여 키워드로 사용. keyword_dic에 {키워드: 포함하는 문장 번호의 리스트} 형식으로 데이터 저장
1 usage  ± speciling
def get_keywords(articles):
    #key_model = KeyBERT()
    key_model = KeyBERT('paraphrase-multilingual-MiniLM-L12-v2')
    hannanum = Hannanum()
    keyword_dic = {}
    stop_words = {"등", "씨", "것", "여성", "혐의", "경찰", "서울", "한국", "긴급", "불패", "이탈리", "10대", "20대", "30대",
                  "40대", "50대", "60대", "70대", "80대", "분노", "한", "징역", "제", "남성", "사고", "평균", "대한민국"}
    keywords = [key_model.extract_keywords(article, keyphrase_ngram_range=(1,2), top_n=2) for article in articles]
    keywords = [k[0][0] + ' ' + k[1][0] for k in keywords]
    for i, summary in enumerate(keywords):
        words = set([word for word in hannanum.nouns(summary) if word not in stop_words])
        keywords[i] = words
        for word in words:
            if word in keyword_dic:
                keyword_dic[word].append(i)
            else:
                keyword_dic[word] = [i]
    return keywords, keyword_dic
```

```python
def summerize_article(file_name, processed_file):
    if(os.path.isfile(processed_file)):
        df = pd.read_csv(processed_file, sep='\t')
        keyword_dic = {}
        for i, words in enumerate(df['키워드']):
            for word in set(words[1:-1].split(',')):
                word = word.strip().strip("'")
                if word in keyword_dic:
                    keyword_dic[word].append(i)
                else:
                    keyword_dic[word] = [i]
        return df, keyword_dic
    df = pd.read_csv(file_name, sep='\t')
    df['기사요약'] = get_summarize(df['기사본문'])
    df['키워드'], keywords_dict = get_keywords(df['기사본문'])
    df.to_csv(processed_file, index=False, sep="\t")
    return df, keywords_dict
```

### (3) Show keywords

- Input: processed article data

- Output: keywords sorted in order that appear in more articles

- Description: I defined a CLI class that organizes the screen for use in the CLI environment. The function to show keywords was implemented as a member function called print_keywords inside the class. This function sorts keywords in the order that they appear in more articles and shows them to the user 10 at a time. Users can move on to the next page or the previous page by typing on the keyboard, and each time they do so, the screen is automatically erased and redrawn. When a user selects a keyword, the print_tites function is called to view articles featuring the keyword.

- Applied concepts:  class, loops, functions, f-string, dictionary, pandas, package

- Code Screenshot

```python
class CLI():

    speciling
    def __init__(self, df, keywords_dict):
        self.df = df
        self.keywords_dict = keywords_dict
        self.system = platform.system()

    speciling
    def run(self):
        self.print_keywords()

    1 usage   speciling
    def print_keywords(self):
        keywords = sorted(self.keywords_dict.keys(), key=lambda x: -len(self.keywords_dict[x]))
        keywords_set = set(keywords)
        n = 0  # n번부터 10개의 키워드를 보여줌
        while True:
            self.clear()
            print("-----------------------------------------------------------")
            print("-----        현재 기사에서 많이 언급되는 키워드        -----")
            print("-----------------------------------------------------------")
            print("-----                이전 10개 보기: b                -----")
            print("-----                다음 10개 보기: n                -----")
            print("-----    키워드가 등장한 기사 보기: 번호 or 키워드     -----")
            print("-----          프로그램 종료: exit or ctrl+c          -----")
            print("-----------------------------------------------------------")
            print()
            for i, keyword in enumerate(keywords[n:n + 10]):
                print(f'{n + i}. {keyword}')
            print()
            while True:
                try:
                    command = input(">>")
                except:
                    break
                if command == "b":
                    if n - 10 >= 0:
                        n -= 10
                        break
                    else:
                        print("첫 페이지입니다. 다른 명령어를 입력해주세요.")
                        continue
                elif command == "n":
                    if n + 10 < len(keywords) - 1:
                        n += 10
                        break
                    else:
                        print("마지막 페이지입니다. 다른 명령어를 입력해주세요.")
                        continue
                elif command == "exit" or command == "quit":
                    return
                elif command.isdecimal() and (keyword_num := int(command)) < len(keywords):
                    self.print_titles(keywords[keyword_num])
                    break
                elif command.isalnum() and command in keywords_set:
                    self.print_titles(command)
                    break
                else:
                    print("올바르지 않은 명령어입니다. 명령어를 다시 확인해주세요.")
```

### (4) Show summary of articles by keyword

- Input: processed article data, selected keyword

- Output: information on articles featuring keywords and the body of articles selected by the user

- Description: The function to show summary of articles by keyword was implemented as a member function called print_keywords inside the CLI class. This function shows five pieces of information (article title, media company name, article summary) in an article containing the keywords you choose. Users can move on to the next page or the previous page by typing on the keyboard, and each time they do so, the screen is automatically erased and redrawn. When a user enters the article number that they want to take a closer look at, it shows the article's title, link, body and keywords in detail.

- Applied concepts:  class, loops, functions, f-string, dictionary, pandas, package

- Code Screenshot

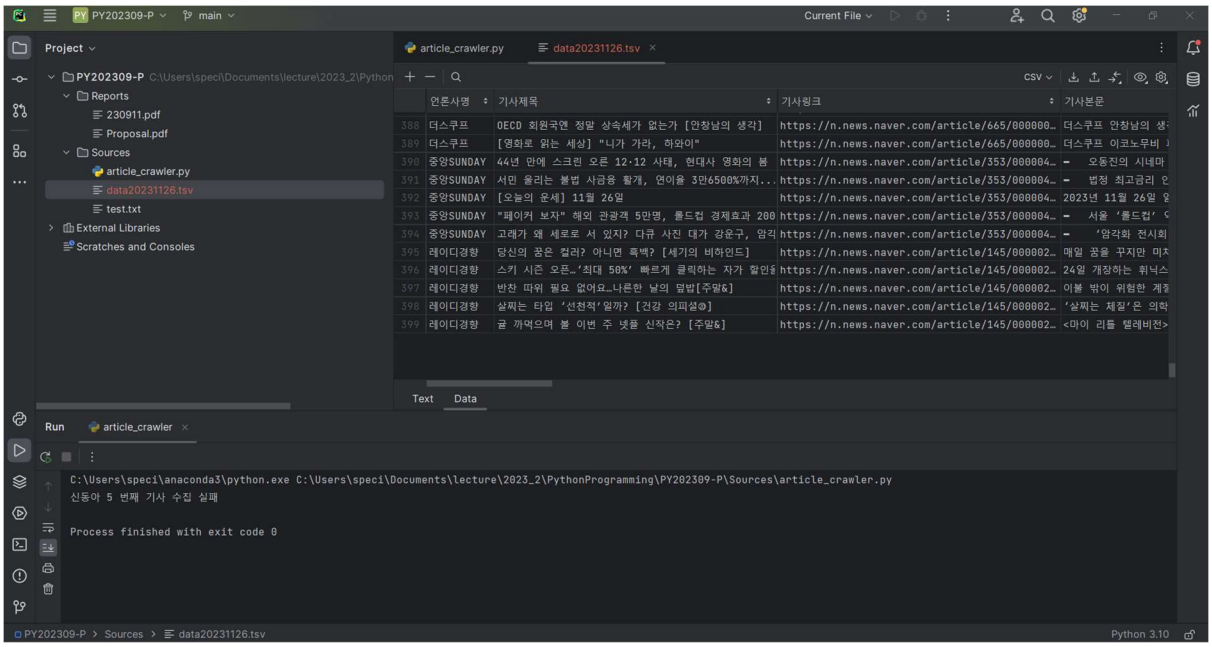
```python
def print_titles(self, keyword):
    n = 0  # n번부터 5개의 기사 정보(제목, 언론사, 요약문)를 출력해줌
    article_nums = self.keywords_dict[keyword]
    while True:
        self.clear()
        print("----------------------------------------------------------------")
        print(f"-----        키워드 '{keyword}' 가 언급된 기사         -----")
        print("----------------------------------------------------------------")
        print("-----                이전 5개 보기: b                -----")
        print("-----                다음 5개 보기: n                -----")
        print("-----          기사 전문 보기: 기사 번호 입력          -----")
        print("-----           키워드 보기로 돌아가기: exit           -----")
        print("----------------------------------------------------------------")
        print()
        for article_num in article_nums[n:n + 5]:
            article = self.df.loc[article_num]
            print(f"{article_num}번 기사")
            print(f"{article['기사제목']}\t{article['언론사명']}")
            print(f"{article['기사요약']}")
            print()
        while True:
            try:
                command = input(">>")
            except:
                break
            if command == "b":
                if n - 5 >= 0:
                    n -= 5
                    break
                else:
                    print("첫 페이지입니다. 다른 명령어를 입력해주세요.")
                    continue
            elif command == "n":
                if n + 5 < len(article_nums) - 1:
                    n += 5
                    break
                else:
                    print("마지막 페이지입니다. 다른 명령어를 입력해주세요.")
                    continue
            elif command == "exit" or command == "quit":
                return
            elif command.isdecimal() and (article_num := int(command)) in article_nums:
                self.clear()
                article = self.df.loc[article_num]
                print(f'기사제목\t{article["기사제목"]}\n')
                print(f'언론사명\t{article["언론사명"]}\n')
                print(f'기사링크\t{article["기사링크"]}\n')
                article_text = article["기사본문"].replace(".", ".\n")  # 가독성을 위해 마침표 뒤에 줄바꿈 추가
                print(f'기사본문\n{article_text}\n')
                print(f'키워드\t{article["키워드"]}\n')
                input("엔터를 눌러 나가기>>")
                break
            else:
                print("올바르지 않은 명령어입니다. 명령어를 다시 확인해주세요.")

3 usages  speciling
def clear(self):  # os별 터미널 화면 초기화 명령어 사용. 코랩에서는 정상적으로 작동하지 않음.
    if self.system == 'Windows':
        os.system("cls")
    else:
        os.system("clear")
```

# 4. Test Result

## (1) Collect articles

- Description: Successfully crawled information from 399 articles across 80 media outlets on Naver News Ranking page. For 'Shindonga' (a specific media outlet), only articles up to the 4th rank were available, resulting in a failure message when attempting to collect the 5th article.

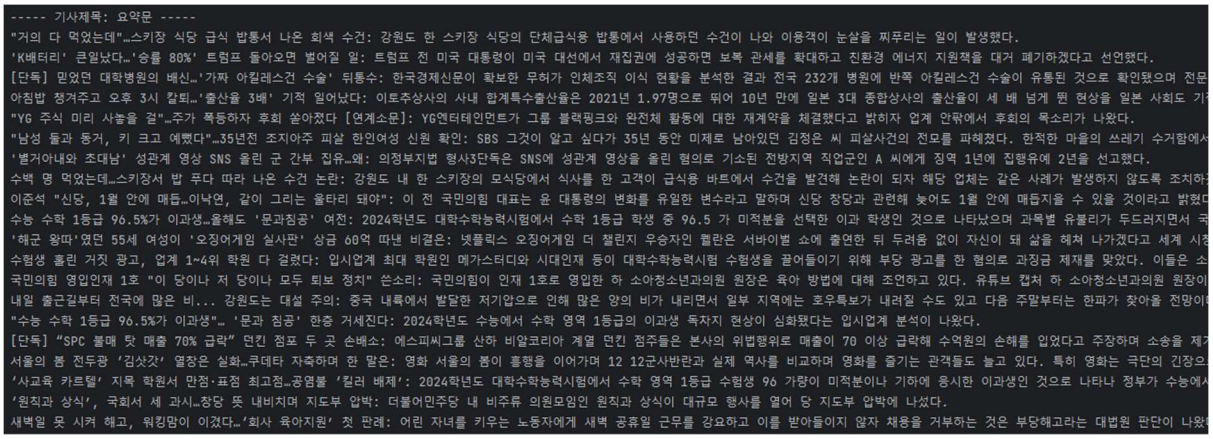
- Test Results Screenshot:

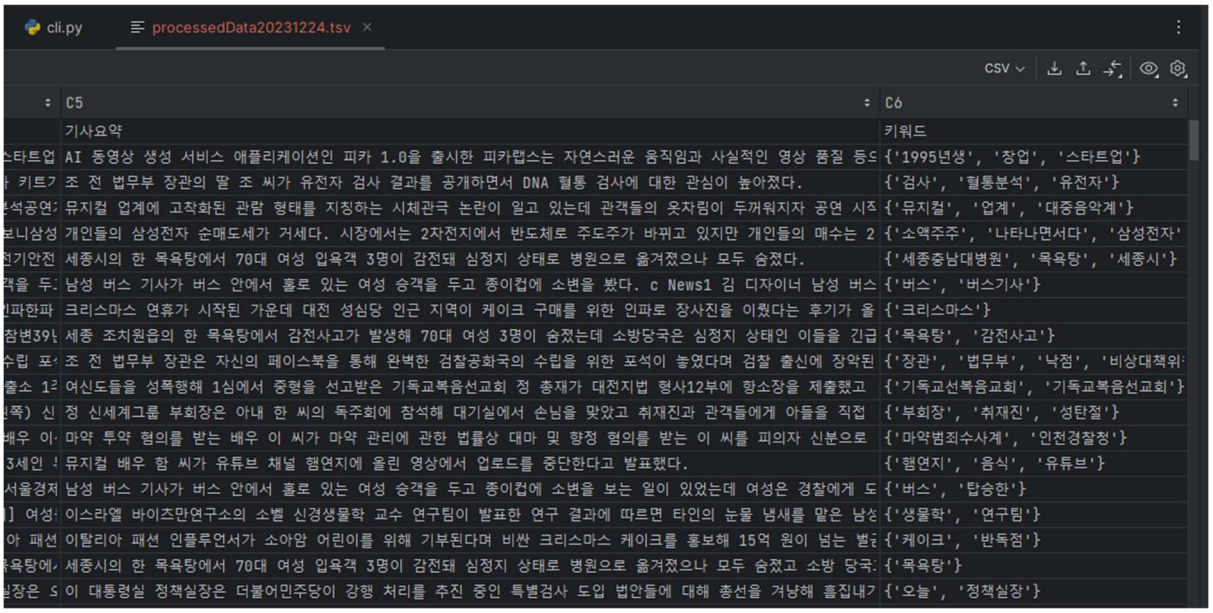



## (2) Summarize the articles

- Description: The article summary worked very well, and keyword extraction also performed not bad using keybert.

- Test Results Screenshot:




## (3) Show keywords

- Description: It was confirmed that 10 keywords appeared well in the CLI environment, and the page was moved well. If a user enters a keyword number or keyword, information on articles containing keywords will appear normally. It also works well when a user enters incorrect commands, missing keyword numbers, or missing keywords.
- Test Results Screenshot:

Anaconda Prompt - python main.py

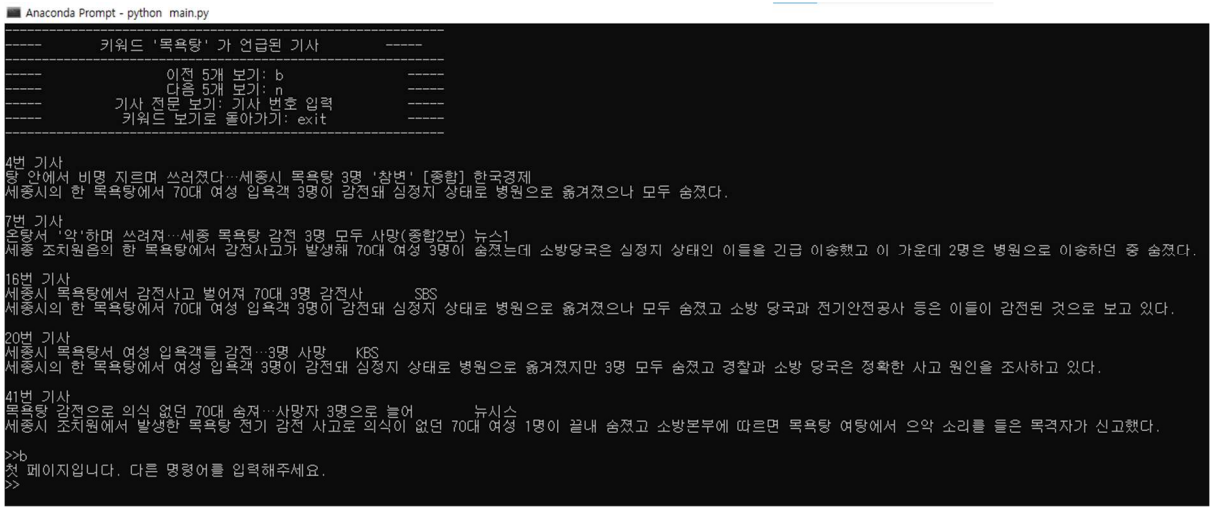
```
------------------------------------------------------------
-----        현재 기사에서 많이 언급되는 키워드          -----
------------------------------------------------------------
-----                이전 10개 보기: b                  -----
-----                다음 10개 보기: n                  -----
-----    키워드가 등장한 기사 보기: 번호 or 키워드      -----
-----          프로그램 종료: exit or ctrl+c            -----
------------------------------------------------------------

0. 목욕탕
1. 국민의힘
2. 크리스마스
3. 장관
4. 정책실장
5. 삼성전자
6. 마약범죄수사계
7. 케이크
8. 유튜브
9. 냉장고

>>b
첫 페이지입니다. 다른 명령어를 입력해주세요.
>>
```

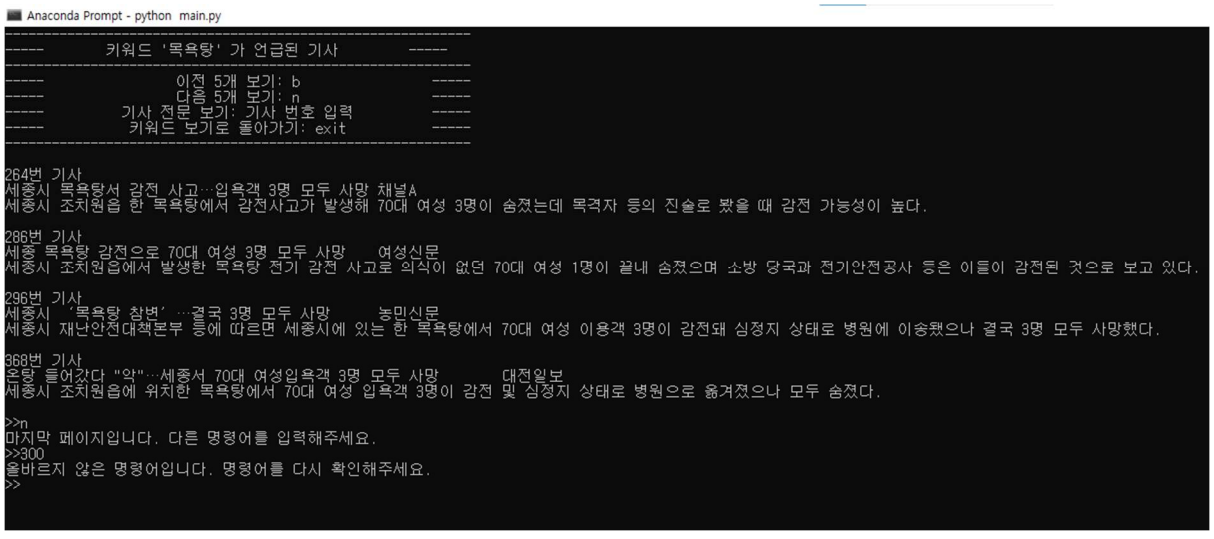
Anaconda Prompt - python main.py

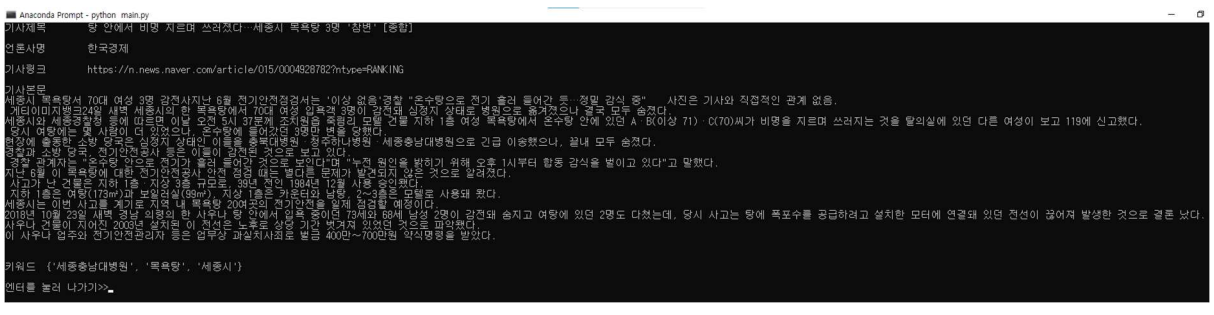
```
------------------------------------------------------------
-----        현재 기사에서 많이 언급되는 키워드          -----
------------------------------------------------------------
-----                이전 10개 보기: b                  -----
-----                다음 10개 보기: n                  -----
-----    키워드가 등장한 기사 보기: 번호 or 키워드      -----
-----          프로그램 종료: exit or ctrl+c            -----
------------------------------------------------------------

760. 축복
761. 파트1
762. 넷플릭스
763. 성
764. 결론
765. 파티
766. 기분
767. 아이돌
768. 길
769. 생명력

>>n
마지막 페이지입니다. 다른 명령어를 입력해주세요.
>>n
마지막 페이지입니다. 다른 명령어를 입력해주세요.
>>?
올바르지 않은 명령어입니다. 명령어를 다시 확인해주세요.
>>999
올바르지 않은 명령어입니다. 명령어를 다시 확인해주세요.
>>없는키워드
올바르지 않은 명령어입니다. 명령어를 다시 확인해주세요.
>>
```

## (4) Show summary of articles by keyword

- Description: Up to five pieces of article information are well displayed at once, and it was confirmed that the detailed information of the article was well displayed when the article number was entered. Like the above function, processing of incorrect user input works well.
- Test Results Screenshot:



## 5. Changes in Comparison to the Plan

### **1)** Removal of Comment Count-related Feature

- Previously: Comment count was collected in Function 1, and when showing article summaries in Function 4, they were displayed in descending order of comment count.

- Now: Comment count is no longer collected in Function 1. In Function 4, article summaries are displayed randomly.

- Reason: Comment count information was not present in the crawled HTML documents. Although considering using Selenium to collect comment counts was an option, it was decided to exclude this feature as it was not deemed crucial, and the time required for article collection became excessively long.

## 6. Lessons Learned & Feedback

I brought an external model and tried it for the first time, and I think it was a great experience to learn how to use the trained model. Thank you for your passionate instruction during the semester!