Python Programming and Practice

Development daily news keyword summary program

Progress Report : 2

Date : 2023.12.10

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 or 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.

3. Progress

1) Function Implementation

(1) Collect articles

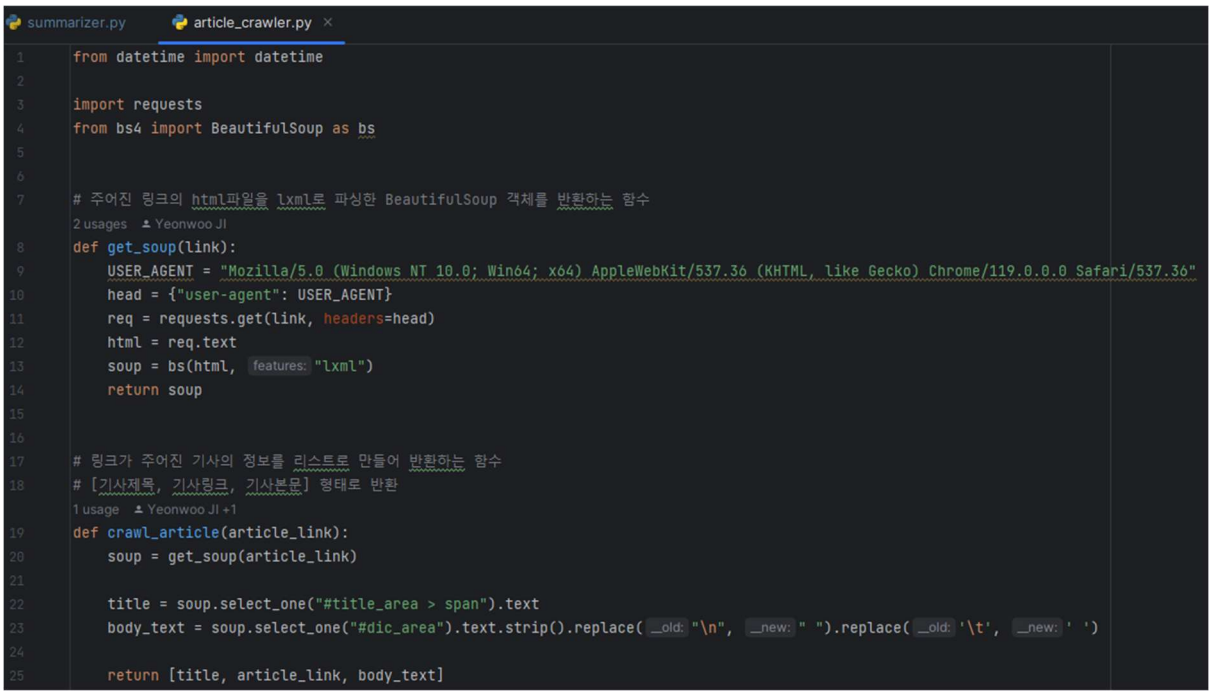
- Input: Link to Naver News Ranking page

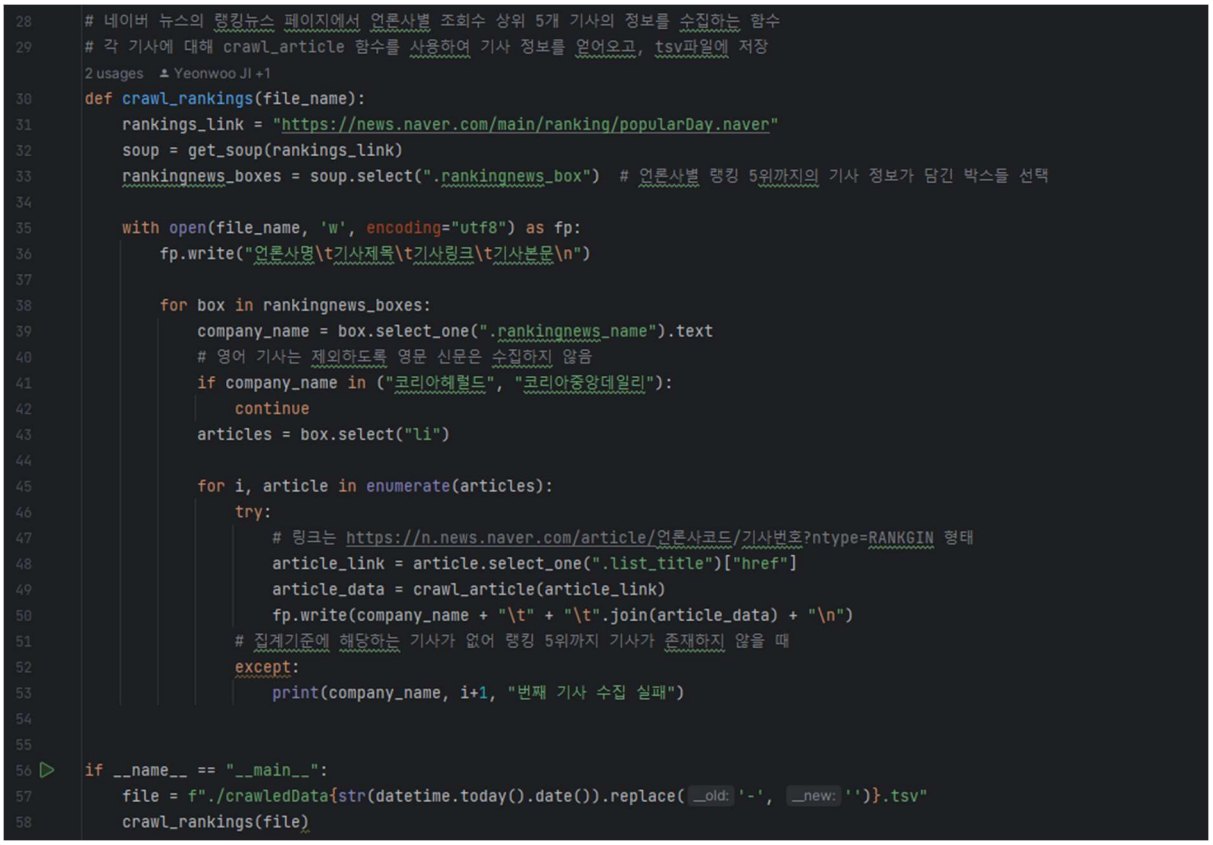
- Output: TSV file containing information of the articles

- 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__".

- Code Screenshot





```python
# 네이버 뉴스의 랭킹뉴스 페이지에서 언론사별 조회수 상위 5개 기사의 정보를 수집하는 함수
# 각 기사에 대해 crawl_article 함수를 사용하여 기사 정보를 얻어오고, tsv파일에 저장
2 usages  Yeonwoo Jl +1
def crawl_rankings(file_name):
    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")


if __name__ == "__main__":
    file = f"./crawledData{str(datetime.today().date()).replace( _old: '-',  _new: '')}.tsv"
    crawl_rankings(file)
```

(2) Summarize the articles

- Input: Crawled articles data

- Output: Summaries of the articles and a dictionary with a keyword extracted from a summary as a key and a list of indexes of the sentence in which the keyword appeared as a value

- Description: Summarizes articles using artificial intelligence models and calculates the frequency of keyword appearance by extracting nouns to be used as keywords from summary sentences

- Applied concepts: loops, functions, list comprehension, file read, file write, dictionary

- Code Screenshot

summarizer.py

```python
import nltk
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
from konlpy.tag import Hannanum

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

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

    for i in range(0, len(articles), 10):
        inputs = ["summarize: " + article for article in articles[i:min(i+10, len(articles))]]
        inputs = tokenizer(inputs, max_length=512, truncation=True, return_tensors="pt", padding=True)
        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(summaries):
    hannanum = Hannanum()
    keyword_dic = {}
    stop_words = ["등", "씨", "것"]
    for i, summary in enumerate(summaries):
        words = set([word for word in hannanum.nouns(summary) if word not in stop_words])
        for word in words:
            if word in keyword_dic:
                keyword_dic[word].append(i)
            else:
                keyword_dic[word] = [i]
    return keyword_dic
```

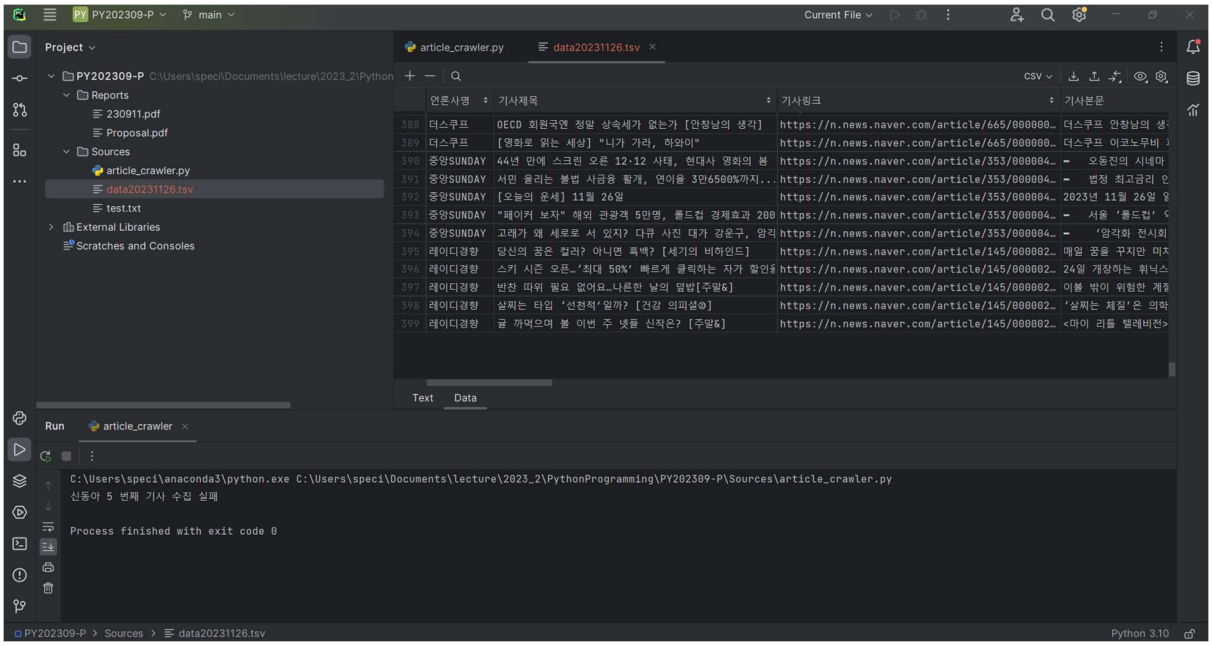
```python
2 usages  speciling *
def test_summarize(file_name):
    titles = []
    articles = []
    with open(file_name, 'r', encoding='utf8') as fp:
        for line in fp.readlines()[1:21]:
            line = line.split('\t')
            titles.append(line[1])
            articles.append(line[-1])
    summarized_sentences = summarize(articles)
    print("----- 기사제목: 요약문 -----")
    for i in range(len(titles)):
        print(titles[i] + ": " + summarized_sentences[i])
    print()
    keywords_dict = get_keywords(summarized_sentences)
    print("----- 키워드 목록 상위 10개 -----")
    top_10 = [keyword for keyword in sorted(keywords_dict.keys(), key=lambda x:-len(keywords_dict[x]))[:10]]
    for keyword in top_10:
        print(keyword + ": " + str(keywords_dict[keyword]))

```

2) Test Results

(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.

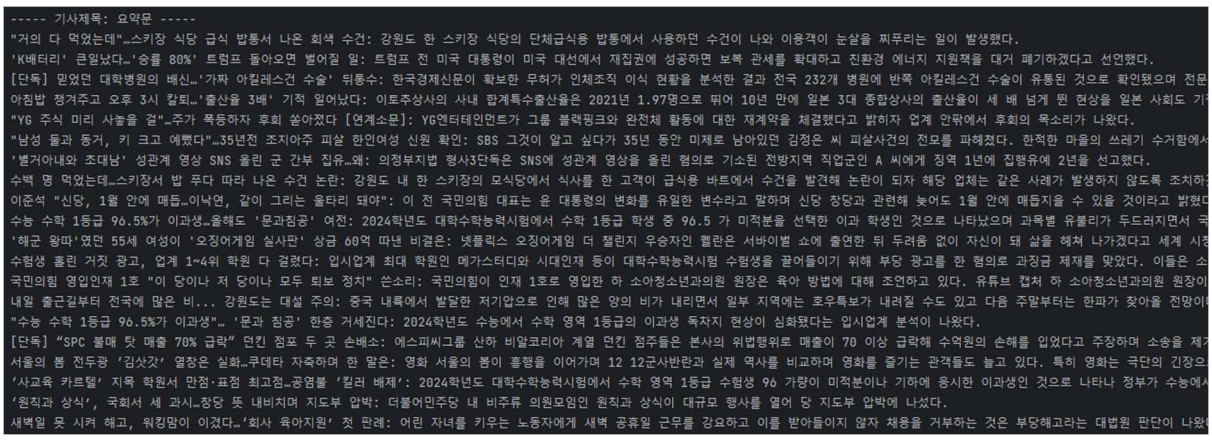
- Test Results Screenshot:



(2) Summarize the articles

- Description: The summary of the article's body works well, but in the case of keywords, contrary to expectations, many insignificant words appear. There is a need to improve the logic related to keywords.

- Test Results Screenshot:



```
----- 키워드 목록 상위 10개 -----
현상: [3, 9, 14]
수: [8, 11, 13]
1등급: [9, 14, 17]
수학: [9, 14, 17]
심화: [9, 14, 17]
이: [9, 11, 19]
대학수학능력시험: [9, 11, 17]
2024학년: [9, 14, 17]
수능: [11, 14, 17]
발생: [0, 7]
```

4. 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.

5. Schedule

- Indicating Progress

업무	11/3	11/17	12/1	12/15	12/22
Write a proposal	**Complete**				

Function 1		**Complete**			
Function 2		**In progress**			
Function 3				**Not Started**	
Function 4				**Not Started**	
Final Report					**Not Started**