과제로 만난사이

강지수, 고병표, 정해앙

과제로 만난사이 주제

네이버 금융, 경제 뉴스 기사를 입력하면 긍정, 부정을 알려주는 모델 만들기

과제로 만난사이 목차

1. 네이버 금융, 경제 뉴스 크롤링

Naive Bayse

2. 나이브 베이즈를 통한 긍정/부정 분류

3. 최종 확인

Take.1

네이버 금융, 경제 뉴스 크롤링

과제로 만난사이 네이버 금융,경제 뉴스 크롤링

```
#필요한 친구들을 불러온다
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np

#나중에 딕셔너리화를 위해 리스트를 만들어준다
news_list = []
date_list = []

for i in range(27,12,-1):  #(nn, mm, -1) 날자 설정 nn부터 mm까지
    print(i)
    base_url = 'https://news.naver.com/main/list.nhn?mode=
        LS2D&sid2=259&sid1=101&mid=shm&date=202008'
            #259 금융, 258 증권

    date_n =i
    page_n = 1
    date = base_url[-6:] + str(date_n)
```

과제로 만난사이

네이버 금융,경제 뉴스 크롤링

```
for j in range(1,24):

    page = '&page='
    URL = base_url + str(date_n) + page + str(page_n)
    print(URL)
    page_n = page_n + 1

    response = requests.get(URL)
    soup = BeautifulSoup(response.text, 'html.parser')

    news_list_1 = soup.select(
        "#main_content > div.list_body.newsflash_body >
        ul.type06_headline > li")  #네이버 뉴스 1~10
        #한페이지에 20개가 나오는 네이버 뉴스는 10개식 나누어져 있음
    news_list_2 = soup.select(
        "#main_content > div.list_body.newsflash_body >
        ul.type06 > li")  #네이버 뉴스 11~20
```

과제로 만난사이

네이버 금융,경제 뉴스 크롤링

```
        #뉴스 제목에 사진이 있는기사와 없는 기사로 나누어 진다.
        for news in news_list_1:  #news_list_1의 사진이 있는 기사
            if news.select_one('dl > dt[class=photo]'):
                a_tag = news.select_one('dl > dt:nth-child(2) > a')
                T = a_tag.text.strip()
                date_list.append(date)
                news_list.append(T)

            else: #사진이 없는 기사
                a_tag = news.select_one('dl > dt:nth-child(1) > a')
                T = a_tag.text.strip()
                date_list.append(date)
                news_list.append(T)
```

네이버 금융,경제 뉴스 크롤링

```
for news in news_list_2: #news_list_2의 사진이 있는 기사
    if news.select_one('dl > dt[class=photo]'):
        a_tag = news.select_one('dl > dt:nth-child(2) > a'))
        T = a_tag.text.strip()
        date_list.append(date)
        news_list.append(T)

    else: #사진이 없는 기사
        a_tag = news.select_one('dl > dt:nth-child(1) > a')
        T = a_tag.text.strip()
        date_list.append(date)
        news_list.append(T)
```

네이버 금융,경제 뉴스 크롤링

```
dic_data = {'date': date_list,
'title': news_list} #딕셔너리화

df = pd.DataFrame(dic_data)  #DataFrame 생성
df = df.set_index("date")  #날짜를 인덱스로 설정
df = df.drop_duplicates(['title'])  #중복값 제거

df.to_csv("test.csv",encoding='utf-8') #test.csv로 저장
```

Take.2

나이브 베이즈를 통한 긍정/부정 분류

나이브 베이즈를 통한 긍정/부정 분류

#필요한 친구들을 불러온다

```
import pandas as pd
import numpy as np
from math import log, exp
from konlpy.tag import Okt
import csv
import re
import collections
from collections import Counter


def start():
    train_datas = open_csv()
    test_data = input()
    prob = naive_bayes(train_datas, test_data, 0.5, 0.5)

    print(f'{test_data}가 부정적일 확률 : {prob[0]},
      긍정적일 확률 : {prob[1]}')
```

과제로 만난사이

나이브 베이즈를 통한 긍정/부정 분류

```
def open_csv():
    df = pd.read_csv("test.csv")

    pos_doc = []
    neg_doc = []

    for i in range(0,len(df)):

        if df.iloc[i,4] == '긍정':
            pos_doc.append(df1.iloc[i,2])
        else:
            neg_doc.append(df1.iloc[i,2])

    train_datas = [[], []]
    train_datas[0] = neg_doc
    train_datas[1] = pos_doc

    return [' '.join(train_datas[0]), ' '.join(train_datas[1])]
```

#리스트 형태는 토큰화하기가 어렵기 때문에, 전부 조인을 해서 하나의 문자열로 만들어준다

과제로 만난사이

나이브 베이즈를 통한 긍정/부정 분류

```
def start_validate():
    train_datas, test_datas = open_csv_validate()

    #트레인 데이터 (긍정, 부정)을 바탕으로 긍정적인 문장을 검사 -> 긍정적인 확률이 높으면
    prob = naive_bayes(train_datas, test_datas[1], 0.5, 0.5)
    print(f'긍정적인 데이터가 부정적일 확률 :
          {prob[0]}, 긍정적일 확률 : {prob[1]}')

    #부정인 문장을 넣고, 실제로 부정이라고 나오는지 확인
    prob = naive_bayes(train_datas, test_datas[0], 0.5, 0.5)
    print(f'부정적인 데이터가 부정적일 확률 :
          {prob[0]}, 긍정적일 확률 : {prob[1]}')
```

과제로 만난사이

나이브 베이즈를 통한 긍정/부정 분류

```
def open_csv_validate():
    df = open_csv()

    del df['date']

    pos_doc = []
    neg_doc = []

    for i in range(0,len(df)):
        if df.iloc[i,4] == '긍정':
            pos_doc.append(df.iloc[i,2])
        else:
            neg_doc.append(df.iloc[i,2])

    train_datas = [[], []]
    train_datas[0] = neg_doc[:13305]
    train_datas[1] = pos_doc[:13305]
    test_datas = [neg_doc[13305:], pos_doc[13305:]]
    return [' '.join(train_datas[0]), ' '.join(train_datas[1])],
           [' '.join(test_datas[0]), ' '.join(test_datas[1])]
```

```
#P(test_data1긍정(train_datas[1]))
#P(test_data1부정(train_datas[0]))
#동전 2개 던져서 앞면 앞면 -> 1/2 * 1/2
#긍정인 데이터셋, '최고 최고' -> 최고의 빈도수 / 전체 단어의 빈도수 * 최고의 빈도수
 / 전체 단어의 빈도수
#어떤 문장 (test_data) -> 문장에 나올 확률 = 각 단어가 등장할 확률을 다 곱한 것
```

과제로 만난사이

나이브 베이즈를 통한 긍정/부정 분류

```
def calculate_doc_prob(train_data, test_data, nowords_weight):
    sw_train_data = re.compile('[^\w]').sub(' ', train_data.lower())  #스탑워드 제거
    sw_train_token = sw_train_data.split()                            #토큰화
    train_vector = dict(Counter(sw_train_token))                      #Bow화 (단어: 빈도수 형태)

    sw_test_data = re.compile('[^\w]').sub(' ', test_data.lower())    #스탑워드 제거
    sw_test_token = sw_test_data.split()                              #토큰화
    test_vector = dict(Counter(sw_test_token))                        #Bow화 (단어: 빈도수 형태)

    total_wc = len(sw_train_token)

    log_prob = 0                                                      #log(P(test_data1긍정))
                                                                      #단어 10개로 이루어진 문장, 각 단어가 나올 확률이
                                                                      (10/500000) -> 문장이 나올 확률은 0.0~1024
                                                                      #e.1024e-50 -> 0 -> (로그함수) -50
                                                                      #확률이 소실되는 것을 방지하고, 값을 간략하게
                                                                      저장하게 위해서 로그를 취한다
    for word in test_vector:
        if word in train_vector:
            log_prob += log(train_vector[word]/total_wc)
        else:
            log_prob += log(nowords_weight/total_wc)

    return log_prob                                                   #train 데이터셋에 없는 단어가 나왔으면... 해당 단어가 나올 확률 추정불가
                                                                      #해당 단어의 빈도수
                                                                      #빈도수가 없는 단어는 우리가 빈도를 지정하주자!
```

과제로 만난사이 나이브 베이즈를 통한 긍정/부정 분류

```
def naive_bayes(train_datas, test_data, pos_prob, neg_prob):
    test_pos_prob = calculate_doc_prob(train_datas[1], test_data, 0.1)
                    + log(pos_prob)
    test_neg_prob = calculate_doc_prob(train_datas[0], test_data, 0.1)
                    + log(neg_prob)
    #P(긍정|test_data) = P(test_data|긍정) * P(긍정) / P(test_data)
    #P(부정|test_data) = P(test_data|부정) * P(부정) / P(test_data)

                                                    # 10 : 5 -> 2 : 1
    maxprob = max(test_neg_prob, test_pos_prob)     # 긍정, 부정의 상대적인 크기
    test_neg_prob -= maxprob    # test_pos_prob (로그값) -> 로그값 -50 -> 지수함수 0에 수렴
    test_pos_prob -= maxprob    # test_neg_prob (로그값) -> 로그값 -50 -> 지수함수 0에 수렴
    print(test_neg_prob, test_pos_prob)     # 로그함수 -> logex, 지수함수 -> e**x
    test_neg_prob = exp(test_neg_prob)      # -50, -52 -> (지수화) e**-50=0, e**-52=0
    test_pos_prob = exp(test_pos_prob)  # -50, -52 -> 0, -2 -> (지수화) e**0 = 1, e**-2 = 0.12
                                        # 긍정 확률 1, 부정 확률 0.12 -> 1/1+0.12, 0.12/1+0.
    #두 확률 값의 상대적인 비율
    normalized_prob = [test_neg_prob/(test_neg_prob+test_pos_prob),
                       test_pos_prob/(test_neg_prob+test_pos_prob)]

    return normalized_prob

start()
```

Take.3

최종 확인

과제로 만난사이

최종 확인

• • • 코로나 대출만기 또 연장…'부실폭탄' 우려

코로나 대출만기 또 연장…'부실폭탄' 우려
0.0 -11.559812419906514
코로나 대출만기 또 연장…'부실폭탄' 우려가 부정적일 확률 : 0.9999904581384623,
긍정적일 확률 : 9.5418615375844e-06

• • • 토스 2,000억 추가 투자유치…몸값 뛰는 핀테크

토스 2,000억 추가 투자유치…몸값 뛰는 핀테크
-4.764369793039961 0.0
토스 2,000억 추가 투자유치…몸값 뛰는 핀테크가 부정적일 확률 : 0.008456144906901 05,
긍정적일 확률 : 0.9915438550930988

• • • NH투자 자산배분 랩어카운트에 1,000억 이상 몰려

NH투자 자산배분 랩어카운트에 1,000억 이상 몰려
-0.35122423451797147 0.0
NH투자 자산배분 랩어카운트에 1,000억 이상 몰려가 부정적일 확률 : 0.4130855789154513,
긍정적일 확률 : 0.5869144210845487

수고하셨습니다.