

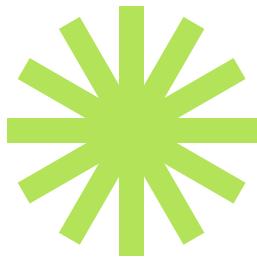
...

NAME.

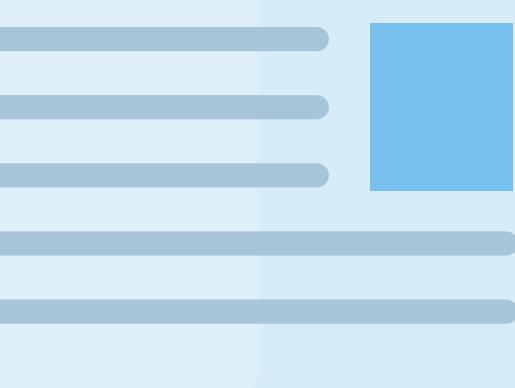
KIM MIN SEO

REALTIME

HEADLINE NEWS AND MARKET REACTION



NEWS



...

CONTENTS

1

분석 목적

2

핵심 내용

3

사용한 데이터

4

분석 코드

5

사용 방향

...

1

분석 목적

최근 몇 달간, 경제 스터디를 진행하며 매일매일 꾸준히 다양한 이슈의 뉴스 기사들을 접하게 되었습니다. 뉴스 기사들을 유심히 살펴보지 않았다면 몰랐을 많은 세계 이슈들을 자연스레 느끼게 되었고, 급격히 변화하는 세상을 빠르게 인식하는 긍정적인 변화를 맛볼 수 있었습니다.

그러나, 요즘 현대 사회는 이러한 기사들을 하나씩 꼼꼼히 읽어보기엔 너무나도 바쁘고, 그렇기에 번거로움을 느끼는 사람들도 있습니다.

따라서 저는 바쁜 현대인들을 대상으로 쉽고 빠른 세계 흐름 파악 기회를 제공하고자 '실시간 헤드라인 뉴스와 시장 반응 프로젝트'를 진행하게 되었습니다.

1

바쁘디 바쁜 현대 사회!

2

무심코 지나친 세계 이슈들

3

더욱 간편하고 쉽게
뉴스 기사 접하기!



...

2

핵심 내용



1

긍부정 분석 모델

가게 리뷰와 그에 따른 부정(0)/긍정(1)/모호함(2) 데
이터가 라벨링된 데이터를 사용하여 긍부정 분석 모델
을 만들었습니다.



2

네이버 뉴스/댓글 크롤링

실시간 네이버 헤드라인 뉴스와 뉴스에 따른 댓글들을
크롤링 하였습니다.



3

시장 반응 분석

뉴스 기사에 달린 댓글들을 기반으로, 뉴스 이슈에 대
한 시장의 반응을 판단하게 하였습니다.

...

3

사용한 데이터셋

1

```
df = pd.read_csv('/content/kr3.csv')  
df.head()
```



Rating

Review

0	1	숙성 돼지고기 전문점입니다. 건물 모양 때문에 매장 모양도 좀 특이하지만 쾌적한 편...
1	1	고기가 정말 맛있었어요! 육즙이 가득 있어서 너무 좋았아요 일하시는 분들 너무 친절...
2	1	잡내 없고 깔끔, 담백한 맛의 순댓국이 순댓국을 안 좋아하는 사람들에게도 술술 넘어...
3	1	고기 양이 푸짐해서 특 순대국밥을 시킨 기분이 듭니다 맛도 좋습니다 다만 양념장이 ...
4	1	순댓국 자체는 제가 먹어본 순대국밥집 중에서 Top5 안에는 들어요. 그러나 밥 양...

맛집 리뷰 데이터

텍스트의 긍정/부정 라벨링이 되어 있는 데이터를 사용하기 위해, 맛집 리뷰 데이터를 사용하였습니다.

3

N 뉴스 | 엔터 | 스포츠 | 날...

언론사별 정치 경제 사회

8.2(토)

네이버 뉴스 크롤링

네이버 뉴스 헤드라인 뉴스를 크롤링하여 데이터로 사용하였습니다.

분석 코드 1 (긍부정 모델 생성)

1

필요한 라이브러리 임포트

필요한 라이브러리를 임포트합니다.

```
import pandas as pd
import numpy as np
import re
from konlpy.tag import Okt
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import joblib
```

...

4

분석 코드 1 (긍부정 모델 생성)

2-1

데이터 전처리하기

tsv 파일을 불러와 csv 파일로 저장합니다.

2-2

데이터 전처리하기

데이터의 기본 정보(결측치, 비율)들을 확인합니다.

```
[ ] # 1. TSV 파일 불러오기  
df = pd.read_csv('kr3.tsv', sep='\t')  
  
# 2. CSV 파일로 저장하기  
df.to_csv('/content/kr3.csv', index=False)
```

```
[ ] df = pd.read_csv('/content/kr3.csv')  
df.head()
```



Rating

Review

0	1	숙성 돼지고기 전문점입니다. 건물 모양 때문에 매장 모양도 좀 특이하지만 쾌적한 편...
1	1	고기가 정말 맛있었어요! 육즙이 가득 있어서 너무 좋았아요 일하시는 분들 너무 친절...
2	1	잡내 없고 깔끔, 담백한 맛의 순댓국이 순댓국을 안 좋아하는 사람들에게도 술술 넘어...
3	1	고기 양이 푸짐해서 특 순대국밥을 시킨 기분이 듭니다 맛도 좋습니다 다만 양념장이 ...
4	1	순댓국 자체는 제가 먹어본 순대국밥집 중에서 Top5 안에는 들어요. 그러나 밥 양...

```
df.isna().sum()
```



0

Rating 0

Review 0

dtype: int64

```
[ ] df['Rating'].value_counts(normalize=True)
```

0은 부정, 1은 긍정, 2는 모호함.

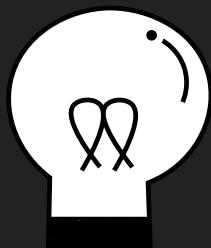


proportion

Rating

1	0.604758
2	0.284749
0	0.110493

dtype: float64



분석 코드 1 (긍부정 모델 생성)

2-3

데이터 전처리하기

`text_cleaning` 함수를 만듭니다.

이 과정에서 불용어를 제거하고, 조사/어미/구두점을 제거합니다.

```
def text_cleaning(text):
    text = re.sub(r"<.*?>|&[^;]+;", " ", text)
    text = re.sub(r"\s+", " ", text)
    text = re.sub(r"[^가-힣a-zA-Z0-9 ]", "", text)

    return text

df['Review'] = df['Review'].apply(lambda x: text_cleaning(x))
df['Review']
```

	Review
0	숙성 돼지고기 전문점입니다 건물 모양 때문에 매장 모양도 좀 특이하지만 쾌적한 편이...
1	고기가 정말 맛있었어요 육즙이 가득 있어서 너무 좋았어요 일하시는 분들 너무 친절하...
2	집내 없고 깔끔 담백한 맛의 순댓국이 순댓국을 안 좋아하는 사람들에게도 술술 넘어갈...
3	고기 양이 푸짐해서 특 순대국밥을 시킨 기분이 듭니다 맛도 좋습니다 다만 양념장이 ...
4	순댓국 자체는 제가 먹어본 순대국밥집 중에서 Top5 안에는 들어요 그러나 밥 양이...
...	...
641757	요즘 핫하게 떠오르고 있는 중국집 맥주의 여 파루 속이 안 좋지만 와봄 일명 선풍...
641758	원래 글 안 쓰는데 이거는 정말 다른 분들 위해서 써야 할 것 같네요 방금 포장 주...
641759	우리 팀 단골집 술 먹고 다음 날 가면 푸짐하게 배불리 해장할 수 있는 곳 주말도 ...
641760	원래는 평택에 있었는데 연남동에도 최근에 생겨서 방문했는데 진짜 줄이 어마어마하더라...
641761	친구들의 추천으로 가보게 된 곳 안성과 평택 몇 군데 위주로 체인점이 있는 소규모 ...

641762 rows × 1 columns

`dtype: object`

```
[ ] def get_pos(x):
    okt = Okt()
    pos = okt.pos(x)
    pos = ['{}/{}'.format(word, tag) for word, tag in pos if tag not in ['Josa', 'Eomi', 'Punctuation']]
    return pos

result = get_pos(df['Review'][0])
print(result)
```

['숙성/Noun', '돼지고기/Noun', '전문점/Noun', '입니다/Adjective', '건물/Noun', '모양/Noun', '때문/Noun', '매장/Noun', ...]

...

4

분석 코드 1 (긍부정 모델 생성)

3

모델 학습하기

TfidfVectorizer를 사용하여
모델을 학습합니다.

4

모델 평가하기

LogisticRegression을 사용하여
모델을 평가합니다.

tqdm.pandas()

```
# 형태소 분석 진행상황 보기  
df['pos'] = df['Review'].progress_apply(get_pos)
```

```
# 이미 토큰화된 데이터를 그대로 사용하기 위한 설정  
def identity_tokenizer(x):  
    return x
```

```
def identity_preprocessor(x):  
    return x
```

```
# 벡터라이저 생성  
tfidvect = TfidfVectorizer(  
    tokenizer=identity_tokenizer,  
    preprocessor=identity_preprocessor,  
    token_pattern=None  
)  
X = tfidvect.fit_transform(df['pos'])
```

```
# 라벨 추출  
y = df['Rating']
```

```
# 학습/테스트 분할  
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)  
  
print("X_train shape:", x_train.shape)  
print("X_test shape:", x_test.shape)
```

```
100%|██████████| 641762/641762 [3:26:50<00:00, 51.71it/s]  
X_train shape: (513409, 247724)  
X_test shape: (128353, 247724)
```

lr = LogisticRegression(random_state=0, class_weight='balanced', max_iter=1000)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.63	0.84	0.72	14182
1	0.84	0.73	0.78	77623
2	0.53	0.61	0.57	36548
accuracy			0.71	128353
macro avg	0.67	0.73	0.69	128353
weighted avg	0.73	0.71	0.71	128353

...

4

분석 코드 1 (긍부정 모델 생성)

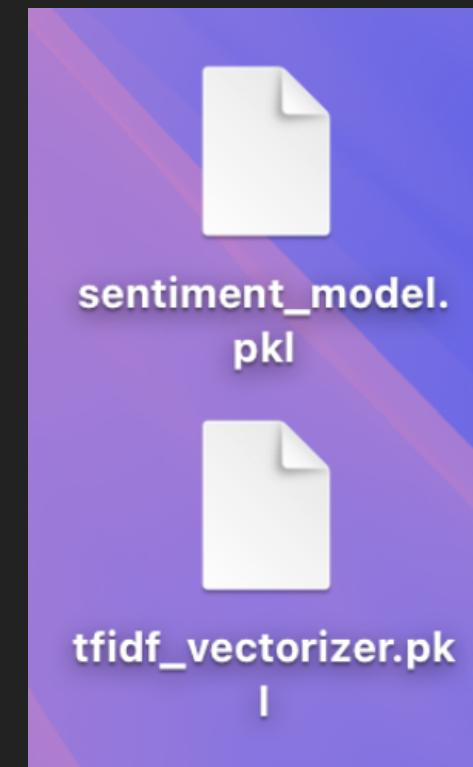
5

모델과 벡터 저장하기

모델과 벡터를 저장합니다.

```
# 모델과 벡터 저장
joblib.dump(lr, 'sentiment_model.pkl')
joblib.dump(tfidvect, 'tfidf_vectorizer.pkl')

['tfidf_vectorizer.pkl']
```



분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

1

필요한 라이브러리 임포트

필요한 라이브러리를 임포트합니다.

```
import requests
from bs4 import BeautifulSoup
from transformers import pipeline
from konlpy.tag import Okt
import pandas as pd
import numpy as np
import re
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
import joblib
import json
from IPython.display import Image, display, Markdown
```

분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

2

네이버뉴스 크롤링

영역별로 네이버뉴스 헤드라인 뉴스를 크롤링하고,
new_dic이라는 딕셔너리에 주요 내용을 저장합니다.

```
def get_soup_obj(news_link):
    headers = {'User-Agent': 'Mozilla/5.0'}
    response = requests.get(news_link, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')

    return soup

def get_top3_news_info(sec, sid):
    default_img = "https://search.naver.com/search.naver?where=image$sm=tab_jum$query=naver#"
    sec_url = "https://news.naver.com/section/%s" % sid
    print("section url : ", sec_url)

    news_list3 = []
    soup = get_soup_obj(sec_url)

    lis3 = soup.find_all("li", class_="sa_item _SECTION_HEADLINE", limit=3)
    for li in lis3:
        title_tag = li.find('strong', class_="sa_text_strong")
        a_tag = li.select_one('a.sa_text_title')
        media_tag = li.find('div', class_='sa_text_info_left')
        img_tag = li.find('img')

        news_info = {
            'title': title_tag.get_text(strip=True) if title_tag else None,
            'media_com': media_tag.get_text(strip=True) if media_tag else None,
            'news_link': a_tag['href'] if a_tag and 'href' in a_tag.attrs else None,
            'img_src': (img_tag.get('data-src') or img_tag.get('src')) if img_tag else None
        }
        news_list3.append(news_info)

    return news_list3
```

```
summarizer = pipeline("summarization", model="csebuetnlp/mT5_multilingual_XLSum")

def naver_news_top3():
    news_dic = dict()
    sections = ['pol', 'eco', 'soc', 'cul', 'wor', 'sci']
    section_ids = ['100', '101', '102', '103', '104', '105']

    for sec, sid in zip(sections, section_ids):
        news_info = get_top3_news_info(sec, sid)

        for news in news_info:
            news_link = news['news_link']
            news_contents = get_article_contents(news_link)
            news['news_contents'] = news_contents

            try:
                summary_output = summarizer(news_info['news_contents'], max_length=128, min_length=30, do_sample=False)
                snews_contents = summary_output[0]['summary_text']
            except Exception as e:
                sentences = news_contents.split('.')
                if len(sentences) > 3:
                    snews_contents = '.'.join(sentences[:3])+'.'
                else:
                    snews_contents = news_contents

            news['snews_contents'] = snews_contents

        news_dic[sec] = news_info

    return news_dic
```

그 과정에서 뉴스 전체 내용을 요약하는 코드도 작성하여, 뉴스 요약본 또한 딕셔너리에 추가합니다

분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

3

댓글 스크래핑

뉴스 기사 별로 댓글을 스크래핑하여,
데이터셋을 구축하는 함수를 작성합니다.

```
def extract_press_article_id(news_link):
    match = re.search(r'oid=(\d+)&aid=(\d+)', news_link)
    if match:
        return match.groups()

    match = re.search(r'/article/(\d{3})/(\d+)', news_link)

    if match:
        return match.groups()

    return None, None

def get_comment_count(press_id, article_id):
    url = "https://apis.naver.com/commentBox/cbox/web_naver_list_jsonp.json"
    params = {
        'ticket': 'news',
        'templateId': 'default',
        'pool': 'cbox5',
        'lang': 'ko',
        'country': 'KR',
        'objectId': f"news{press_id},{article_id}",
        'pageSize': 1,
        'indexSize': 10
    }
    headers = {
        'Referer': f"https://news.naver.com/main/read.naver?oid={press_id}&aid={article_id}",
        'User-Agent': 'Mozilla/5.0'
    }
    response = requests.get(url, headers=headers, params=params)

    if response.status_code == 200:
        try:
            json_str = response.text[response.text.find('(')+1:-2]
            data = json.loads(json_str)
            return data['result']['count']['total']
        except Exception as e:
            print(f"[X] JSON 파싱 오류] {press_id}, {article_id}: {e}")
            return None
    else:
        print(f"[X] 요청 실패] status_code={response.status_code}, press_id={press_id}, article_id={article_id}")
        return None
```

```
def count_comments(news_dic, sec):
    for article in articles:
        news_link = article.get('news_link')
        press_id, article_id = extract_press_article_id(news_link)
        if press_id and article_id:
            count_comment = get_comment_count(press_id, article_id)
        else:
            count_comment = None
    return count_comment
```

댓글 수를 파악하는 함수도 생성합니다.

```
def get_comments_only(news_link):
    press_id, article_id = extract_press_article_id(news_link)
    if not press_id or not article_id:
        return []

    object_id = f"news{press_id},{article_id}"
    referer_url = f"https://news.naver.com/main/read.naver?oid={press_id}&aid={article_id}"
    url = "https://apis.naver.com/commentBox/cbox/web_naver_list_jsonp.json"
    headers = {
        'Referer': referer_url,
        'User-Agent': 'Mozilla/5.0'
    }

    comments = []
    page = 1
    while True:
        params = {
            'ticket': 'news',
            'templateId': 'default',
            'pool': 'cbox5',
            'lang': 'ko',
            'country': 'KR',
            'objectId': object_id,
            'pageSize': 100,
            'indexSize': 10,
            'page': page
        }

        response = requests.get(url, headers=headers, params=params)
        if response.status_code != 200:
            break

        try:
            json_str = response.text[response.text.find('(')+1:-2]
            data = json.loads(json_str)
            comment_list = data['result'].get('commentList', [])
            if not comment_list:
                break
            for c in comment_list:
                comments.append(c['contents'])
            page += 1
            time.sleep(0.2)
        except:
            break

    return comments
```

분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

4

긍부정 모델 적용

앞서 만들어둔 긍부정 모델을 적용하여,
댓글의 긍부정 정도를 파악합니다.

```
def collect_all_comments_only(news_dic, sec):
    all_comments_list = []
    for article in articles:
        news_link = article.get('news_link')
        comments = get_comments_only(news_link)
        all_comments_list.extend(comments)
    return pd.DataFrame({'comment': all_comments_list})

def comment_emotion(df_all_comments, model, tfidvect):
    def text_cleaning(text):
        text = re.sub(r"*.*/>|&[^;]+;", " ", text)
        text = re.sub(r"\s+", " ", text)
        text = re.sub(r"^[가-힣a-zA-Z0-9 ]", "", text)
        return text

    def get_pos(x):
        okt = Okt()
        pos = okt.pos(x)
        return ['{}/{}'.format(word, tag) for word, tag in pos if tag not in ['Josa', 'Eomi', 'Punctuation']]

    df_all_comments['cleaned'] = df_all_comments['comment'].apply(text_cleaning)
    df_all_comments['pos'] = df_all_comments['cleaned'].apply(get_pos)

    X = tfidvect.transform(df_all_comments['pos'])

    preds = model.predict(X)

    df_all_comments['pred'] = preds
    df_all_comments['emotion'] = df_all_comments['pred'].apply(
        lambda x: '긍정' if x==1 else('부정' if x==0 else '모호함')
    )

    return df_all_comments[['comment', 'emotion']]

def emotion_ratio(result_df):
    total = len(result_df)

    emotion_ratio = (
        result_df['emotion']
        .value_counts(normalize=True)
        .mul(100)
        .round(2)
        .to_dict()
    )

    return emotion_ratio
```

분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

5

정리해보기

앞선 내용들을 정리하여 출력해봅니다.

```

def identity_tokenizer(x): return x
def identity_preprocessor(x): return x
model = joblib.load('sentiment_model.pkl')
tfidvect = joblib.load('tfidf_vectorizer.pkl')

sections = ['pol', 'eco', 'soc', 'cul', 'wor', 'sci']

for sec in sections:
    print(f"\n■ 섹션: {sec}")
    articles = news_dic.get(sec, [])

    for i, article in enumerate(articles):
        news_link = article.get('news_link')
        news_title = article.get('title')
        print(f" ■ 기사 {i+1}: {news_title}")

        comments = get_comments_only(news_link)
        comment_count = len(comments)

        if comment_count >= 5:
            print(f" ● 댓글 수가 {comment_count}개로 충분합니다. 댓글 분석을 시작합니다.")

            df_all_comments = collect_all_comments_only(news_dic, sec)
            result_df = comment_emotion(df_all_comments, model, tfidvect)

            emotion_ratios = emotion_ratio(result_df)
            for emotion, ratio in emotion_ratios.items():
                print(f"해당 기사에 대해 {emotion} 반응은 {ratio:.2f}%입니다.")

        else:
            print(f" ● 댓글 수가 {comment_count}개로 부족하여, 시장 반응을 확인하기 어렵습니다.")
    print('\n')

```

■ 섹션: soc

■ 기사 1: 약속한 듯 같은 날 3번째, 용인·부산 이어 세종서도...택시 상가 돌진, 1명 부상

● 댓글 수가 56개로 충분합니다. 댓글 분석을 시작합니다.

해당 기사에 대해 모호함 반응은 52.63%입니다.

해당 기사에 대해 부정 반응은 41.05%입니다.

해당 기사에 대해 긍정 반응은 6.32%입니다.

■ 기사 2: 명태균 재소환 "성실히 특검에서 조사하고 수사하는 부분 협조" [현장영상+]

● 댓글 수가 19개로 충분합니다. 댓글 분석을 시작합니다.

해당 기사에 대해 모호함 반응은 52.63%입니다.

해당 기사에 대해 부정 반응은 41.05%입니다.

해당 기사에 대해 긍정 반응은 6.32%입니다.

■ 섹션: cul

■ 기사 1: 한때 '낮 최고' 37도까지...내륙 곳곳엔 소나기 [오늘날씨]

● 댓글 수가 4개로 부족하여, 시장 반응을 확인하기 어렵습니다.

■ 기사 2: 세계유산 홍보·연구 강화...‘반구천세계암각화센터’ 만든다

● 댓글 수가 1개로 부족하여, 시장 반응을 확인하기 어렵습니다.

■ 기사 3: 생의 끝까지 전쟁 기록한 우크라이나 작가...‘여성과 전쟁’

● 댓글 수가 0개로 부족하여, 시장 반응을 확인하기 어렵습니다.

■ 섹션: wor

■ 기사 1: 쿠글러 연준 이사, 깜짝 사임...국채시장 발작 2년물 26bp '뚝'(상보)

● 댓글 수가 0개로 부족하여, 시장 반응을 확인하기 어렵습니다.

■ 기사 2: 美무역대표 “무역 대상국과 추가 합의, 몇 건 안 될 수도”

● 댓글 수가 0개로 부족하여, 시장 반응을 확인하기 어렵습니다.

■ 기사 3: “실패한 대통령”... 러 명목상 ‘2인자’ 겨냥한 트럼프의 조롱

● 댓글 수가 0개로 부족하여, 시장 반응을 확인하기 어렵습니다.

분석 코드 2 (네이버 뉴스 크롤링 및 시장반응)

6

섹션 선택 및 헤드라인 뉴스

섹션을 선택하면, 헤드라인 뉴스 관련 정보들이 제시
되게 합니다.

```
sections = ['pol', 'eco', 'soc', 'cul', 'wor', 'sci']
print("사용 가능한 섹션:", sections)

selected_sec = input("검색할 섹션을 입력하세요: ").strip()

if selected_sec not in sections:
    print("X 유효하지 않은 섹션입니다. 다시 실행해주세요.")
else:
    print(f"선택된 섹션: {selected_sec}")
    articles = news_dic.get(selected_sec, [])

    for i, article in enumerate(articles):
        news_link = article.get('news_link')
        news_title = article.get('title')
        img_url = article['img_src']
        summary = article['snews_contents'].replace('\n', '\n\n')
        print(f"■ 기사 {i+1}")
        display(Image(url=img_url))
        display(Markdown(f"### ■ {article['title']}\n\n{summary}"))

    comments = get_comments_only(news_link)
    comment_count = len(comments)

    if comment_count >= 5:
        print(f" 댓글 수가 {comment_count}개로 충분합니다. 댓글 분석을 시작합니다.")

        df_all_comments = collect_all_comments_only(news_dic, sec)
        result_df = comment_emotion(df_all_comments, model, tfidvect)

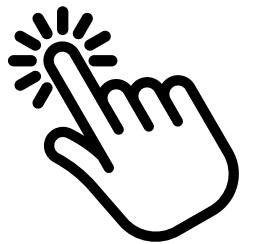
        emotion_ratios = emotion_ratio(result_df)
        for emotion, ratio in emotion_ratios.items():
            print(f"해당 기사에 대해 {emotion} 반응은 {ratio:.2f}%입니다.")

    else:
        print(f" 댓글 수가 {comment_count}개로 부족하여, 시장 반응을 확인하기 어렵습니다.")

print('\n')
```



사용 방향



1 쉽고 빠른 세계 흐름 파악



매일 그리고 실시간으로 변하는 분야별 헤드라인 뉴스와 요약본을 토대로 쉽고 빠르게 세계 흐름 파악이 가능하다.

2 시장 반응 파악



나의 생각 뿐만 아니라, 타인의 생각들을 살펴봄으로써 해당 이슈에 대한 시장의 반응은 어떠한지 한 눈에 파악이 가능하다.

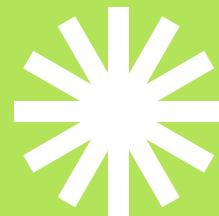
3 더 자세히 알고 싶다면?



요약본을 보고 더 자세히 알아보고 싶은 뉴스 기사가 생긴다면, 제시된 뉴스 링크를 타고 들어가 확인이 가능하다.

...

THANK YOU!



이상으로 프로젝트 설명을 마치겠습니다.

PROJECT