

기업 뉴스 수집 및 감성 분석

강의 목표

- ✓ 기업 관련 뉴스 데이터를 자동으로 수집하는 방법 습득
- ✓ 뉴스 데이터를 이용해 시장 심리 및 평판 분석 방법 학습
- ✓ Python & 오픈소스를 활용한 실전 분석 파이프라인 경험
- ✓ 수집 → 분석 → 저장 → 시각화까지 전 과정 실습

왜 기업 뉴스 분석이 중요한가?

포인트	설명
⚠️ 리스크 관리	기업 위기 및 부정적 이슈 조기 파악 (리콜, 사고 등)
💰 투자 의사결정	시장 심리 파악 → 투자 전략 수립 보조
🔍 경쟁사 분석	경쟁사 동향 모니터링 및 시장 트렌드 파악
🌱 ESG 트렌드	환경/사회/지배구조 관련 긍정적 변화 감지
📊 데이터 기반 인사이트	비정형 텍스트 → 정량화된 지표로 변환

“기업은 수많은 뉴스로 평가받습니다. 데이터를 빠르게 정리하고 이해하는 능력은 경쟁력입니다.”

뉴스 데이터 수집 방법





방법	장단점
 웹 크롤링 (BeautifulSoup)	간편하지만 사이트 구조 변화에 민감
 동적 크롤링 (Selenium)	Javascript 렌더링 페이지 대응 가능
 API 활용 (NewsAPI 등)	구조화된 데이터 제공, 속도 빠름

- 주요 소스
- 네이버 뉴스 (국내 주요 언론)
 - 구글 뉴스 (글로벌 기사)
 - 언론사 RSS / Open API

감성 분석이란?

- 텍스트에서 감정이나 태도(Positive / Negative / Neutral) 를 추출하는 기술
- 인간의 감정이나 의견을 기계가 이해하도록 변환
- 비정형 데이터 → 정량적 지표로 변환

활용 사례

-  기사 분석: 기업/산업 이슈 파악
-  소비자 리뷰 분석: 상품 개선
-  소셜 미디어 분석: 트렌드 예측
-  금융: 주가 예측 모델의 보조 지표

감성 분석 방법론

방법	설명	예
사전 기반 (Lexicon)	긍/부정 단어 리스트 기반	호재', '성장' → Positive
머신러닝 기반	데이터 학습을 통해 감정 분류	SVM, RandomForest
딥러닝 기반	문맥 이해 및 복합 감정 분석	BERT, KoBERT 등

실습 포인트

- 긍/부정 키워드 사전 활용
- 실시간으로 긍/부정/중립 결과 확인
- 감정 분포 시각화

분석 결과의 비즈니스 활용

활용 방안	설명
 시장 반응 모니터링	긍부정 트렌드 변화 감지
 리스크 알림	부정 키워드 급증 시 알림
 투자 전략 보조	긍정 뉴스 증가 시 매수 전략
 마케팅 전략 수립	소비자 반응 및 이슈 대응

"데이터는 비즈니스 의사결정의 근거로 연결됩니다."

실습 로드맵

- 데이터 수집
 - 네이버 뉴스 키워드 기반 크롤링
- 데이터 분석
 - 긍/부정 키워드 감성 분석
- 데이터 시각화
 - 워드클라우드 / 감성 분포 차트
- 데이터 저장 및 활용
 - DuckDB 저장 및 향후 확장

Step 1: 네이버 뉴스에서 기업 뉴스 수집하기

- BeautifulSoup를 사용해 네이버 뉴스 검색 결과에서 뉴스 제목과 URL을 수집합니다.
- 검색어와 페이지 수를 지정할 수 있어 다양한 기업의 뉴스 데이터를 수집할 수 있습니다.

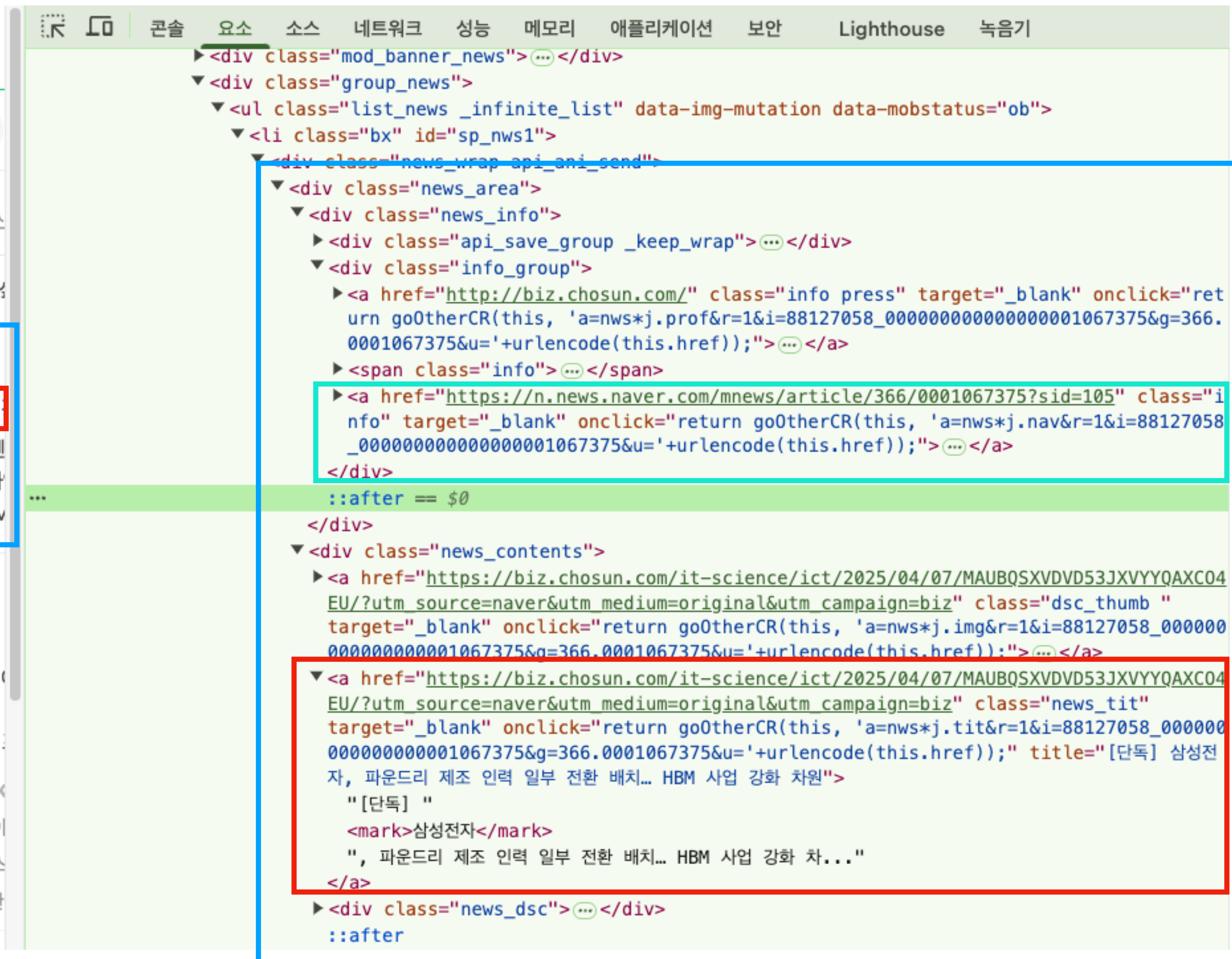
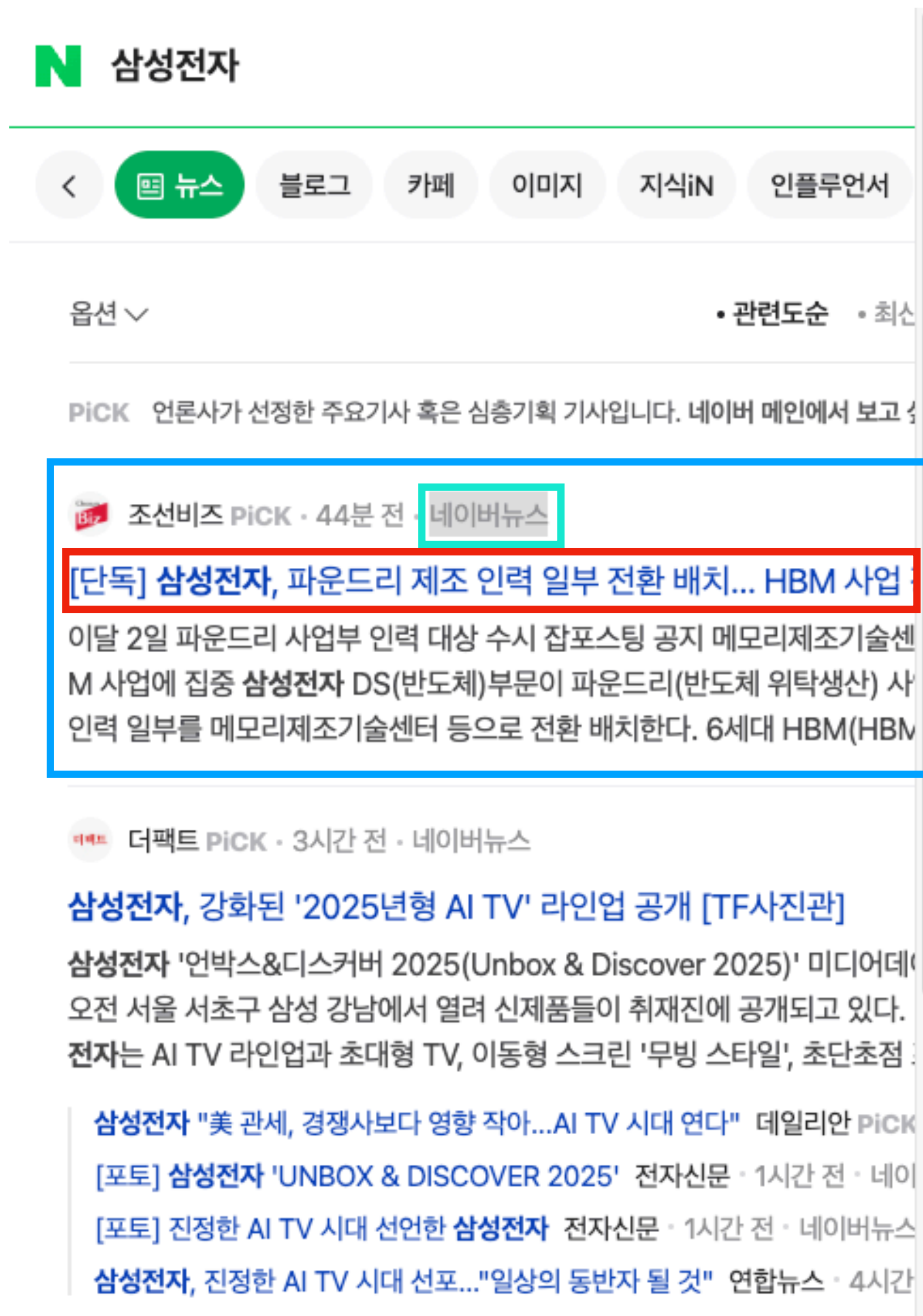
```
# 필요한 라이브러리 импорт
import requests
from bs4 import BeautifulSoup
import pandas as pd

print("[INFO] 필요한 라이브러리 импорт 완료!")
```

Step 1: 네이버 뉴스에서 기업 뉴스 수집하기

```
# 뉴스 수집 함수 정의
def get_naver_news(query, pages=1):
    print(f"[INFO] 뉴스 수집 시작: 검색어='{query}', 페이지 수={pages}")
    results = []
    for page in range(1, pages * 10, 10):
        url = f"https://search.naver.com/search.naver?where=news&query={query}&start={page}"
        print(f"[DEBUG] 요청 URL: {url}")
        response = requests.get(url, headers={'User-Agent': 'Mozilla/5.0'})
        if response.status_code != 200:
            print(f"[ERROR] 요청 실패: 상태 코드 {response.status_code}")
            continue
        soup = BeautifulSoup(response.text, "html.parser")
        articles = soup.select("div.news_area")
        print(f"[DEBUG] 페이지 {page//10 + 1} - 수집된 기사 수: {len(articles)}")
        for article in articles:
            title_tag = article.select_one("h2")
            info_group = article.select("div.info_group a")
            if len(info_group) >= 2:
                url = info_group[1]['href'] # 두 번째 a 태그의 href 속성
            else:
                url = title_tag['href'] # 예비: 만약 두 번째 a 태그 없으면 기존 방식 사용
            title = title_tag.get_text()
            print(f"[DEBUG] 기사 제목: {title} | 원문 URL: {url}")
            content = get_article_content(url)
            results.append({"title": title, "url": url, "content": content})
    print(f"[INFO] 뉴스 수집 완료! 총 수집 건수: {len(results)}")
    return pd.DataFrame(results)
```


Step 1: 네이버 뉴스에서 기업 뉴스 수집하기



Step 1: 네이버 뉴스에서 기업 뉴스 수집하기

```
# 본문 내용 크롤링 함수 정의
def get_article_content(article_url):
    try:
        response = requests.get(article_url, headers={'User-Agent': 'Mozilla/5.0'})
        if response.status_code != 200:
            print(f"[ERROR] 기사 본문 요청 실패: {response.status_code} - {article_url}")
            return ""
        soup = BeautifulSoup(response.text, "html.parser")
        content = soup.select_one(" ")
        if content:
            text = content.get_text(strip=True)
            print(f"[DEBUG] 기사 본문 길이: {len(text)}")
            return text
        else:
            print("[WARNING] 본문 선택자에서 내용이 비어 있음")
            return ""
    except Exception as e:
        print(f"[ERROR] 본문 크롤링 중 예외 발생: {e}")
        return ""
```


Step 1: 네이버 뉴스에서 기업 뉴스 수집하기

```
# 뉴스 수집 실행
news_df = get_naver_news("삼성전자", pages=2)
print("[INFO] 수집된 뉴스 데이터프레임 미리보기:")
print(news_df.head())
```

Step 2: 감성 분석 준비

- 긍정과 부정 키워드를 정의합니다.
- 뉴스 제목과 본문에서 키워드를 찾아 감성 점수를 계산하는 함수를 준비합니다.

```
# 긍정/부정 키워드 정의
positive_words = ["호재", "성장", "기대", "강세", "최고"]
negative_words = ["리스크", "하락", "위기", "악재", "최저"]
print("[INFO] 감성 분석 키워드 정의 완료!")
```

Step 2: 감성 분석 준비

```
# 감성 분석 함수 정의
def sentiment_score(text):
    pos = sum(word in text for word in positive_words)
    neg = sum(word in text for word in negative_words)
    score = "Positive" if pos > neg else "Negative" if neg > pos else "Neutral"
    print(f"[DEBUG] 텍스트 길이: {len(text)} | 긍정: {pos}, 부정: {neg}, 결과: {score}")
    return score
```

Step 3: 뉴스 감성 분석 적용

- 뉴스 제목 + 본문을 활용하여 감성 점수를 계산합니다.

```
print("[INFO] 감성 분석 적용 시작!")
news_df['sentiment'] = news_df.apply(lambda row: sentiment_score(
    row['title'] + row['body'], analyzer), axis=1)
print("[INFO] 감성 분석 적용 완료!")
print(news_df.head())
```

뉴스 감성

Step 4: 데이터 시각화

- 워드클라우드를 통해 자주 등장하는 키워드를 시각화합니다.
- 감성 분포를 파이 차트로 나타내어 긍/부정 비율을 시각적으로 확인합니다.

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

print("[INFO] 데이터 시각화 시작!")

# Colab 환경에서 한글 폰트 설치
!apt-get update -qq
!apt-get install -y fonts-nanum
import matplotlib.pyplot as plt
plt.rc('font', family='NanumGothic')
font_path =  '/usr/share/fonts/truetype/nanum/NanumGothic.ttf' 
```

Step 4: 데이터 시각화

```
# 워드클라우드 생성
text = ' '.join(news_df['title'])
wordcloud = WordCloud(font_path=font_path, width=800, height=400).generate(text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("뉴스 제목 워드클라우드")
plt.show()

print("[INFO] 워드클라우드 시각화 완료!")
```

Step 5: 분석 결과 DuckDB에 저장

- DuckDB를 사용하여 로컬 또는 클라우드 환경에서도 SQL 기반으로 데이터를 다룰 수 있습니다.
- 분석 결과를 DuckDB 데이터베이스 파일로 저장합니다.
- 저장된 데이터는 추후 SQL 쿼리로 분석하거나 외부 시스템과 연동할 수 있습니다.

```
print("[INFO] DuckDB에 저장 시작!")

# DuckDB 설치 및 импорт
!pip install duckdb
import duckdb

# DuckDB에 데이터 저장
con = duckdb.connect(database='news_sentiment.duckdb', read_only=False)
print("[INFO] DuckDB 연결 성공!")
con.execute("CREATE TABLE IF NOT EXISTS news_sentiment AS SELECT * FROM news_df")
print("[INFO] 데이터 저장 완료!")
con.close()
print("[INFO] DuckDB 연결 종료!")
```

Step 6: Streamlit 웹 서비스 실행

- Streamlit을 사용하여 분석된 데이터를 웹 브라우저에서 시각화합니다.
- DuckDB에 저장된 데이터를 불러와 데이터 테이블, 감성 분포 차트, 뉴스 원문 링크를 제공합니다.
- ngrok을 사용하여 외부에서 접근 가능한 URL을 생성합니다.

```
print("[INFO] Streamlit 웹 서비스 준비 시작!")  
  
# 필요한 라이브러리 설치  
!pip install streamlit pyngrok --quiet
```

Step 6: Streamlit 웹 서비스 실행

- app.py 파일 생성
- 서비스 코드 작성



The screenshot shows a code editor interface with a file explorer on the left and a code editor on the right. A blue arrow points to the file explorer, and another blue arrow points to the '새 파일' (New File) option in the context menu.

File Explorer (Left):

- Files: Gemini가 작성한 코드로 파일 분석, 업로드
- Directories: sample_data
- Files: news_sentiment.duckdb

Code Editor (Right):

```
app.py X
1 import streamlit as st
2 import duckdb
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 st.title("📊 기업 뉴스 감성 분석 결과")
7 st.markdown("네이버 뉴스 기반으로 수집한 데이터를 분석한 결과입니다.")
8
9 # DuckDB에서 데이터 읽기
10 con = duckdb.connect(database='news_sentiment.duckdb', read_only=True)
11 df = con.execute("SELECT * FROM news_sentiment").df()
12
13 # 데이터 미리보기
14 st.subheader("뉴스 데이터 미리보기")
15 st.dataframe(df)
16
17 # 감성 분포 차트
18 st.subheader("감성 분포 차트")
19 sentiment_counts = df['sentiment'].value_counts()
20 fig, ax = plt.subplots()
21 sentiment_counts.plot(kind='pie', autopct='%1.1f%%', colors=['lightgreen', 'lightcoral', 'lightgrey'], ax=ax)
22 ax.set_ylabel('')
23 st.pyplot(fig)
24
25 # 원문 링크 추가 표시
26 st.subheader("뉴스 원문 링크")
27 for idx, row in df.iterrows():
28     st.markdown(f"- [{row['title']}]({row['url']})")
29
30 con.close()
```

Step 6: Streamlit 웹 서비스 실행

```
# ngrok으로 공개 URL 생성 (Auth token 필요)
from pyngrok import ngrok

# 여기에 발급받은 ngrok auth token 입력하세요!
NGROK_AUTH_TOKEN = "여기에 발급받은 ngrok auth token 입력하세요!"
```


Step 6: Streamlit 웹 서비스 실행


```
ngrok.set_auth_token(NGROK_AUTH_TOKEN)

print("[INFO] ngrok 터널 생성 중...")
public_url = ngrok.connect(addr="8501", proto="http")
print(f"[INFO] Streamlit 앱 Public URL: {public_url}")

# Streamlit 앱 실행
!streamlit run app.py --server.port 8501 --server.enableCORS false

print("[INFO] Streamlit 웹 서비스가 실행되었습니다. 위의 Public URL을 클릭하여 접속하세요!")
```

Step 6: Streamlit 웹 서비스 실행



```
[INFO] ngrok 터널 생성 중...
[INFO] Streamlit 앱 Public URL: NgrokTunnel: "https://d043-35-224-228-240.ngrok-free.app" -> "http://localhost:8501"
2025-04-07 05:23:38.191
Warning: the config option 'server.enableCORS=false' is not compatible with 'server.enableXsrfProtection=true'
As a result, 'server.enableCORS' is being overridden to 'true'.

More information:
In order to protect against CSRF attacks, we send a cookie with each request.
To do so, we must specify allowable origins, which places a restriction on
cross-origin resource sharing.

If cross origin resource sharing is required, please disable server.enableXsrfProtection.

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.28.0.12:8501
External URL: http://35.224.228.240:8501

Stopping...
Stopping...
[INFO] Streamlit 웹 서비스가 실행되었습니다. 위의 Public URL을 클릭하여 접속하세요!
```