SNS Text를 통한 감정 분석

인공지능

컴퓨터공학과-2020108269 -정현석

2024.11.00

목차

서론

SNS Text 감정 분석을 선택한 이유

- SNS는 사람들이 감정을 표현하고 의견을 나누는 중요한 공간이다. 여기 쌓인 텍스트 데이터를 통해 사람들의 감정과 사회적 반응을 분석하면, 여론 파악이나 고객 요구 이해 등 여러 분야에 활용할 수 있다. 이 프로젝트는 SNS 텍스트 감정 분석을 통해 사람들의 감정을 구조적으로 이해하고, 다양한 응용 가능성을 탐구하기 위해 진행하게 되었다.

서론

필요성

- **실시간 여론 파악:** sns 텍스트는 실시간으로 업데이트되기 때문에 빠르게 변화하는 여론이나 사회적 반응을 즉각적으로 파악할 수 있어서, 기업이나 기관은 고객의 요구와 사회적 반응에 민첩하게 대응이 가능하다
- **정책 및 마케팅 전략 수립:** 감정 분석을 통해 특정 이슈에 대한 사람들의 긍정적 또는 부정적 반응을 예측하고, 이를 바탕으로 정책이나 마케팅 전략을 효과적으로 설계가 가능하다.
- **정서 및 심리 분석 활용:** SNS 데이터를 활용해 개인 또는 집단의 감정 변화를 분석함으로써 정신 건강과 관련된 연구나 심리 상태 모니터링에도 기여가 가능하다.

관련 사례


금융 · 증권 / 은행

고객상담에 AI 감정분석 도입…"보이스피싱 의심 감지"

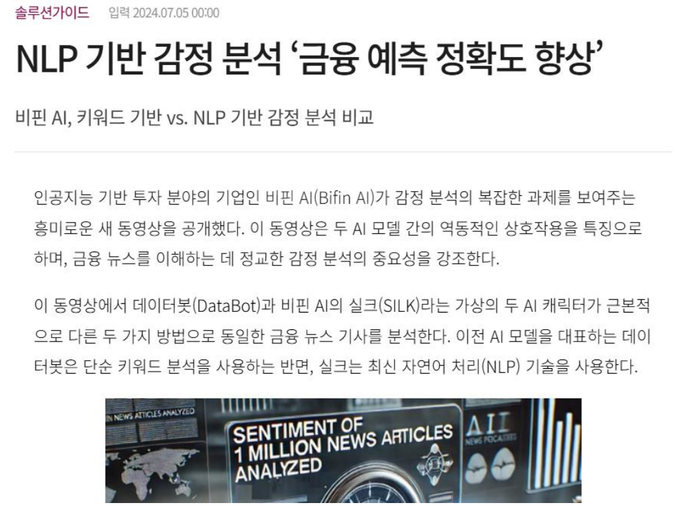
박동해 기자

2024.08.08 오전 10:21

파이썬 초음파 검사,
몽 시나리오를 폭로

많이 본 뉴스

1 "저승사자 보인
수미 별세 후 제


솔루션가이드 입력 2024.07.05 00:00

NLP 기반 감정 분석 '금융 예측 정확도 향상'

비핀 AI, 키워드 기반 vs. NLP 기반 감정 분석 비교

인공지능 기반 투자 분야의 기업인 비핀 AI(Bifin AI)가 감정 분석의 복잡한 과제를 보여주는 흥미로운 새 동영상을 공개했다. 이 동영상은 두 AI 모델 간의 역동적인 상호작용을 특징으로 하며, 금융 뉴스를 이해하는 데 정교한 감정 분석의 중요성을 강조한다.

이 동영상에서 데이터봇(DataBot)과 비핀 AI의 실크(SILK)라는 가상의 두 AI 캐릭터가 근본적으로 다른 두 가지 방법으로 동일한 금융 뉴스 기사를 분석한다. 이전 AI 모델을 대표하는 데이터봇은 단순 키워드 분석을 사용하는 반면, 실크는 최신 자연어 처리(NLP) 기술을 사용한다.

감정분석을 사용한 사례

- **[1]고객상담에 AI 감정분석 도입:** AI의 감정분석을 통하여 보이스피싱을 예방하고 방지한다.
- **[2]NLP 기반 감정 분석 '금융 예측 정확도 향상':** 감정분석을 통해 금융 주식에 관한 예측도를 측정하고 정확도를 향상시킨다.

출처:[1] https://www.ajunews.com/view/20240808093929241

[2]https://www.gttkorea.com/news/articleView.html?idxno=11891

관련 연구/내용

심리학적 감정과 소셜 웹 자료를 이용한 감성의 실증적 분류

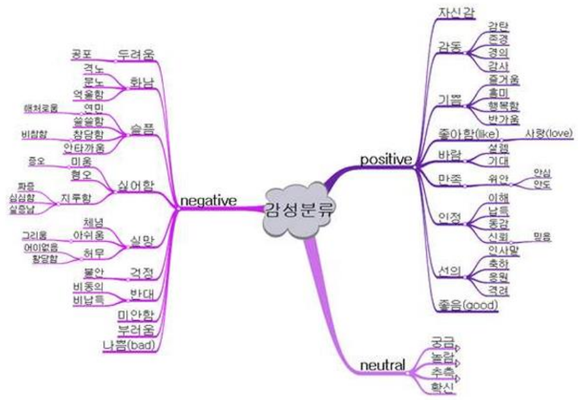


그림 4. 재분류된 감성 카테고리
Fig. 4. Re-classified Sentiment Categories

국내 연구로는 플러칙의 감정 바퀴를 대립 감정 형식으로 24개의 감정으로 분류한 윤여선 연구[13]이 있다. 표 1은 윤여선 연구에서 제안하는 감정 명세로서 대립 형식에 맞추기 위하여 플러칙의 감정 중 일부를 수정하고 있다. 이 연구에서는 대립되는 감정으로 분류를 하고 있으나 극성에 따른 분류는 제시되지 않고 있다. 예를 들어 "화남(anger) "과 "두려움(fear)"는 대립되는 감정으로 표시되고 있지만 감성의 극성으로 보면 둘 모두 "부정"으로 분류된다. 인문학에서 다루는 감정 분류를 오피니언 마이닝에서 사용하기 위해서는 극성에 따라 감정을 재분류할 필요가 있다.

표 1. 대립 감정 명세
Table 1. Opposition Emotion Specification

	감정명	대립 감정명
기본감정	JOY_HAPPYNESS	SADNESS
	ANGER	FEAR
	SURPRISE	DESIRE_ANTICIPATION
	TRUST	DISGUST
조합감정	LOVE	REMORSE
	DISAPPOINTMENT	PRIDE
	CONTEMPT	SHAME
	AGGRESSION	FATALISM
	OPTIMISM	PESSIMISM
	GUILT	GRATEFULNESS
	CURIOSITY	CYNISM
	ENVY	SENTIMENTALITY

3. 제안하는 감성 분류

감정이나 감성에 대한 연구는 언제나 연구자들 간에 이견이 있고 합의된 형태는 존재하지 않는다. 감정에 대한 기존 연구[11][14]에서 제시하는 기본 감정들에 대해서 극성으로 나누어 보면 표 2와 같이 대체로 유사하지만 분류 방식에 따라 감정의 군집이 달라진다. 여기서 중립은 긍정과 부정으로 분류되지 않고 다른 문장 요소나 상황에 따라 극

표 2. 기본 감정들의 극성 분류
Table 2. Polarity Classification of Basic Emotions

	부정적	긍정적	중립
Plutchik	Anger, disgust, fear, sadness,	Joy, acceptance, anticipation	Surprise
Ekman	Anger, disgust, fear, sadness	Joy	Surprise
Frijda	Sorrow	Happiness, interest, desire	Surprise, wonder
Tomkins	Anger, disgust, fear, contempt, shame, distress	Joy, interest	surprise
Watson	Fear, rage	love	
Weiner and Graham	Sadness	Happiness	

앞에서 언급한 것처럼 감성의 종류에 대한 분류는 관점에 따라 다르고, 또한 분류의 목적에 따라 달라진다. 본 논문에서는 이슈가 쉽게 드러나는 제품이나 국가 정책 등을 주제로 하는 소셜 웹의 텍스트를 대상으로 주요 감성을 분류하고자 한다.

상품의 경우에는 상품평에서 호감의 의견이 "감동"에서 출발한 것인지 "기대"에서 출발한 것인지에 따라 관련 회사의 대응 전략이 달라질 수 있다. 따라서 앞으로의 오피니언 마이닝에서는 의견에 대한 단순한 감성의 강도뿐만 아니라 그 의견의 근거가 되는 감성의 종류도 요구될 수 있다. 이를 위하여 본 논문에서는 합리적인 감성의 기준으로 심리학적 기본 감정들을 감성 분류 기준에 맞춰 기본 감성으로 정의하고자 한다. 그리고 실제로 소셜 웹에서 표현되고 있는 감성들의 분포를 반영하기 위하여 소셜 웹의 텍스트를 분석하여 그로부터 기본 감성 카테고리를 재분류하고자 한다. 본 논문에서는 이러한 목적을 위하여 감성 카테고리를 구축

SNS Text 감정분석 참고 논문

- **심리학적 감정 사용:** 플러칙의 감정 바퀴 등 심리학적 감정 모델을 기반으로 감정을 정의하여 감정 분류의 정확도를 높였다. 따라서 이러한 감정 분류는 감정의 세부적 종류를 제시할 수 있다.
- **감정 정보 추출 및 프레임 구성:** 감정 정보 프레임을 구성하여 감정의 강도와 극성 을 기준으로 감정을 재분류 했다. 여기에서는 기쁨 ,신뢰 ,슬픔 ,분노 등 23개의감정 카테고리가 제시되었다.

출처: 장문수-심리학적 감정과 소셜 웹 자료를 이용한 감성의 실증적 분류 2012-10

데이터 전처리

위의 논문과 관련된 전처리 형식

- **불용어 제거:** 의미 없는 단어를 제거하여 분석에 중요한 단어만 남김.
- **소문자 변환 및 특수 문자 제거:** 텍스트 데이터를 정규화 하여 데이터 일관성을 높임.
- **어간 추출 및 표제어 추출:** 단어의 기본형을 남겨 의미를 단순화.
- **벡터화:** 텍스트 데이터를 수치형 벡터로 변환하여 모델이 처리할 수 있게 구축.

그림 출처: https://www.megazonesoft.com/blog-machine-learning/

데이터 전처리

전처리 예시

I am happy 문장을 전처리하는 과정

1. **소문자 변환:** 모든 텍스트를 소문자로 변환하여, I am happy가 i am happy로 바뀐다. 이렇게 바꾸면 대소문자 차이로 인한 데이터 불일치가 방지가 된다.

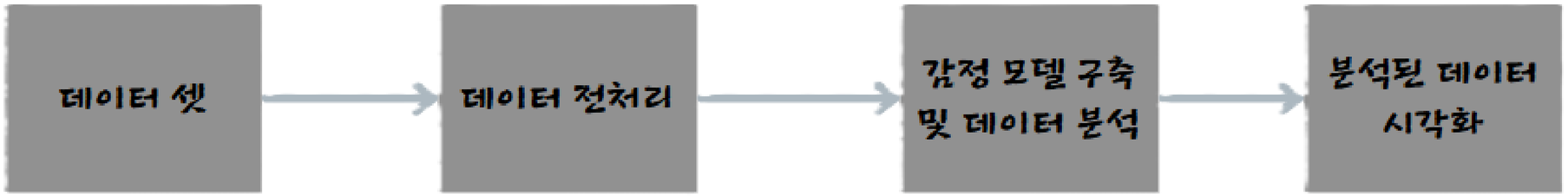
2. **불용어 제거:** 감정에 크게 기여하지 않는 불용어를 제거한다. 여기서는 I 와 am 이 불용어로 간주가 되어 제거되고, happy만 남는다.

3. **어간 추출 또는 표제어 추출:** happy는 기본형이기 때문에 변화가 없다 하지만 다른단어 running이라면 run으로 변환된다.

4. **벡터화:** 최종 전처리된 단어인 happy를 벡터화 단계에서 수치형 데이터로 변환시킨다. 예를들면 happy라는 단어의 빈도가 1로 기록된 벡터 형태로 변환 시킬 수 있다.

전체 개요

SNS Text 감정분석 구축하여 사람들의 감정을 분석



전체 개요

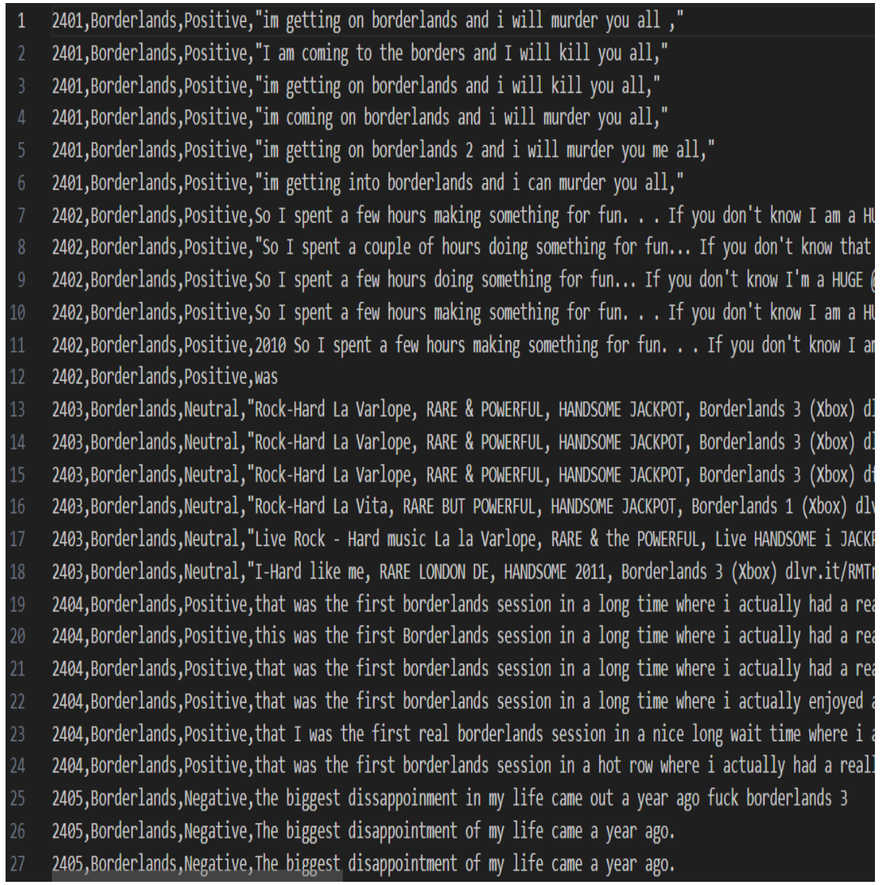
감정 분석 사용하는 이유

- 감정 분석 AI를 활용하여 온라인 마켓에 있는 상품의 글들을 판별하여 실질적인 사람들의 리뷰글들을 추출

그림 출처:[1] http://word.tta.or.kr/dictionary/dictionaryView.do?subject=%EA%B0%90%EC%84%B1%20%EB%B6%84%EC%84%9D

[2] https://r2bit.com/book_tm/sentiment.html

데이터 셋



```
2401,Borderlands,Positive,"im getting on borderlands and i will murder you all ,"
2401,Borderlands,Positive,"I am coming to the borders and I will kill you all,"
2401,Borderlands,Positive,"im getting on borderlands and i will kill you all,"
2401,Borderlands,Positive,"im coming on borderlands and i will murder you all,"
2401,Borderlands,Positive,"im getting on borderlands 2 and i will murder you me all,"
2401,Borderlands,Positive,"im getting into borderlands and i can murder you all,"
2402,Borderlands,Positive,So I spent a few hours making something for fun. . . If you don't know I am a HU
2402,Borderlands,Positive,"So I spent a couple of hours doing something for fun... If you don't know that
2402,Borderlands,Positive,So I spent a few hours doing something for fun... If you don't know I'm a HUGE (
2402,Borderlands,Positive,So I spent a few hours making something for fun. . . If you don't know I am a HU
2402,Borderlands,Positive,2010 So I spent a few hours making something for fun. . . If you don't know I am
2402,Borderlands,Positive,was
2403,Borderlands,Neutral,"Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dl
2403,Borderlands,Neutral,"Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dl
2403,Borderlands,Neutral,"Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dl
2403,Borderlands,Neutral,"Rock-Hard La Vita, RARE BUT POWERFUL, HANDSOME JACKPOT, Borderlands 1 (Xbox) dlv
2403,Borderlands,Neutral,"Live Rock - Hard music La la Varlope, RARE & the POWERFUL, Live HANDSOME i JACKP
2403,Borderlands,Neutral,"I-Hard like me, RARE LONDON DE, HANDSOME 2011, Borderlands 3 (Xbox) dlvr.it/RMTm
2404,Borderlands,Positive,that was the first borderlands session in a long time where i actually had a rea
2404,Borderlands,Positive,this was the first Borderlands session in a long time where i actually had a rea
2404,Borderlands,Positive,that was the first borderlands session in a long time where i actually had a rea
2404,Borderlands,Positive,that was the first borderlands session in a long time where i actually enjoyed a
2404,Borderlands,Positive,that I was the first real borderlands session in a nice long wait time where i a
2404,Borderlands,Positive,that was the first borderlands session in a hot row where i actually had a reall
2405,Borderlands,Negative,the biggest dissappoinment in my life came out a year ago fuck borderlands 3
2405,Borderlands,Negative,The biggest disappointment of my life came a year ago.
2405,Borderlands,Negative,The biggest disappointment of my life came a year ago.
```

SNS 감정 분석에 사용될 데이터 셋

- 트위터의 보더랜드라는 게임과 관련
- 총 74682개의 트위터 댓글들을 학습

데이터 셋 출처: https://www.kaggle.com/models/ruhul20/twitter-sentiment-analysis?select=twitter_training.csv

사용될 모델

모델 비교

모델명	모델 개요	모델 특징
나이브 베이즈 (Naive Bayes)	확률 기반 모델로, 각 단어가 특정 감정(긍정, 부정등)에 속할 확률을 계산하여 전체 텍스트의 감정을 예측	• **빠르고 효율적:** 계산이 빠르고, 대규모 텍스트 데이터를 처리할 때 성능이 좋음 • **텍스트 분류에 유리** • **독립 가정:** 각 단어가 독립적으로 가정 하기 때문에, 단어 간 상관관계를 반영하기 어려움
랜덤 포레스트 (Random Forest)	여러 개의 결정트리(Decision Treses)를 조합한 앙상블 모델로, 각 트리의 예측을 다수결로 결합하여 최종 예측을도출한다.	• **높은 정확도:** 여러 트리를 결합하여 예측하기 때문에, 단일 트리보다 높은 정확도를 가짐 • **과적합 방지:** 다양한 샘플링과 무작위서을 통해 과접합을 방지하며, 일반화 성능이 뛰어남
결정 트리 (Decision Tree)	데이터를 트리 구조로 분할하여 분류 또는 회귀를 수행하는 모델이다. 각 노드는 특정 조건에 따라 데이터를 분할하고, 리프 노드에서 최종 클래스를 예측한다.	• **직관적이고 해석가능:** 트리 구조가 직관적이라 이해하기 쉬움 • **간단한 문제에 적합** • **복잡한 패턴을 포작하는데 한계가 있음**

사용될 모델

GPT 와 다른 모델 비교

CHAT GPT

정확도: 높음
속도: 느림
감정 표현 처리: 비정형적, 미묘한 감정까지 처리 가능

복잡한 문맥과 미묘한 감정을 다뤄야 할 때 사용, 감정 알고리즘을 사용할 것이고 대규모 데이터 활용을 하지않아 적합하지 않음

나이브 베이즈

정확도: 중간(단순한 텍스트나 명확한 패턴에서 효과적)
속도: 빠름
감정 표현 처리: 정형적이고 명확한 감정 표현에서만 효과적

빠르고 간단한 감정 분석이 필요할 때 사용,
감정 알고리즘을 사용할 것이고 빠른 처리 속도를 가지고 있어 이번 감정 분석 활용에 탁월함

데이터 테스트

```python
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

라이브러리 코드

- **Pandas:** 데이터 처리 및 분석 라이브러리
- **Numpy:** 수치계산 라이브러리
- **Seaborn:** 통계 데이터 시각화 라이브러리
- **Warnings:** 경고 메시지 제어 라이브러리
- **Matplotlib:** 데이터 시각화 라이브러리

데이터 테스트

```
columns_nm = ['Id','Entry','target','text']
```

```
df = pd.read_csv('twitter_training.csv', encoding = 'unicode_escape',names=columns_nm)
```

+ Code + Markdown

```
df
```

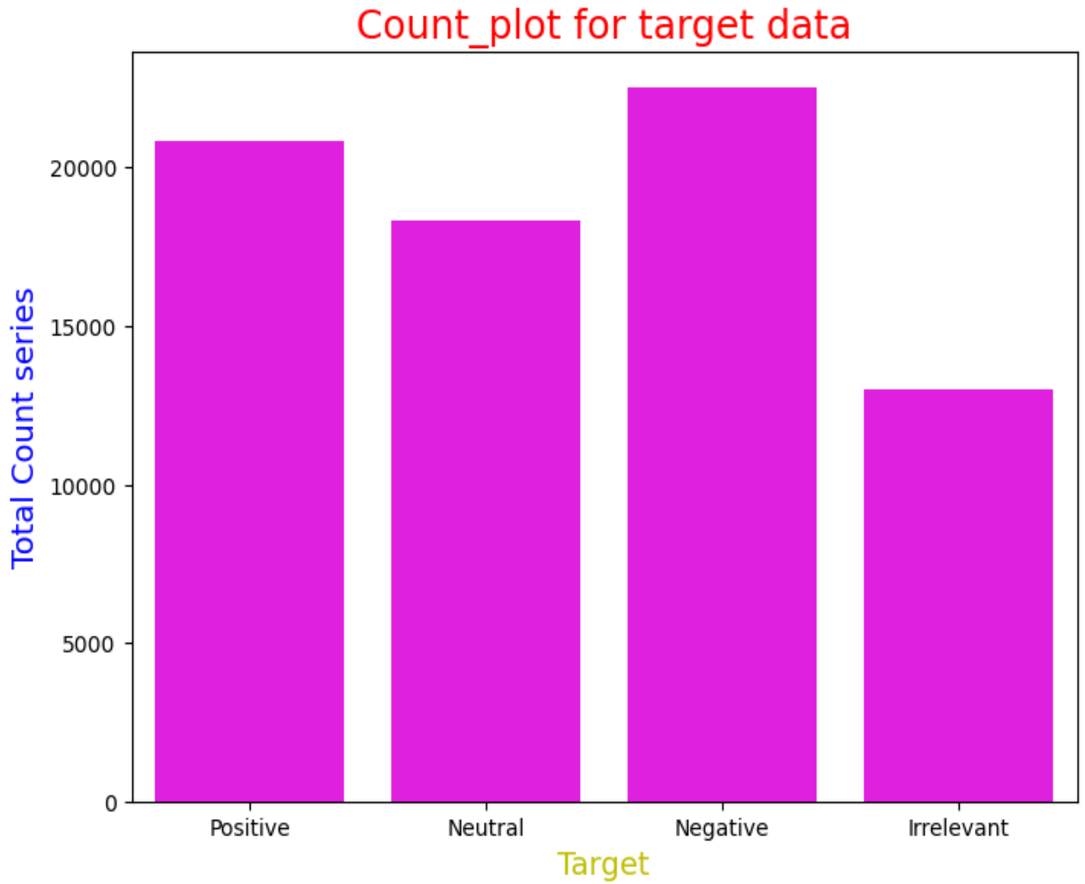
	Id	Entry	target	text
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
...	...	...	...	...
74677	9200	Nvidia	Positive	Just realized that the Windows partition of my...
74678	9200	Nvidia	Positive	Just realized that my Mac window partition is ...
74679	9200	Nvidia	Positive	Just realized the windows partition of my Mac ...
74680	9200	Nvidia	Positive	Just realized between the windows partition of...
74681	9200	Nvidia	Positive	Just like the windows partition of my Mac is l...

74682 rows × 4 columns

Csv 데이터 리스트 생성

- **Id:** 각 데이터 항목을 고유하게 식별
- **Entry:** 주요 타이틀 게임 이름 또는 기업 이름이 있다
- **Target:** 텍스트 감정값이 들어감
- **Text:** 사용자가 작성한 메시지 또는 댓글 내용

데이터 테스트



학습할 데이터 그래프 분석

- **Negative**는 가장 빈도가 높으며 약 22,542개 대다수 차지
- **Positive**는 두 번째로 빈도가 높으며 약 20,832개
- **Neutral**은 약 18,318개로, 중간 수준의 빈도수 차지
- **Irrelevant**는 약 12,990개로 가장 낮은 빈도수 차지

데이터 테스트

```python
def processword(text):
    # smaller convert
    text = text.lower()

    # remove puntuations
    text = re.sub(r"[^\w\s]", "", text)

    # remove numbers
    text = re.sub(r"\d+", "", text)

    #split text into tokens
    tokens = word_tokenize(text)

    #remove stopwords
    stop_wprds = set(stopwords.words("English"))
    tokens = [word for word in tokens if word not in stop_wprds]

    # Stemmer
    stemmer = PorterStemmer()
    stemmer_token = [stemmer.stem(token) for token in tokens]

    ## Lemmatizer
    lematizer = WordNetLemmatizer()
    lemmatize_token = [lematizer.lemmatize(token) for token in stemmer_token]

    return " ".join(lemmatize_token)
df['process_text'] = df['text'].apply(processword)
```

데이터 전처리

1. 함수 정의
2. 텍스트 전처리
    - 소문자로 변화
    - 구두점 제거 (예시: ,.? 제거)
    - 숫자 제거
    - 토큰화 (텍스트 단어화)
    - 불용어 제거(분석의 필요없는 is ,a, the, and 제거)
    - 어간 추출(running-> run)
    - 원형 복원(better -> good, children->child)
    - 단어 결합(리스트 형태로 나눈 단어를 하나로 합침)

데이터 테스트

```
from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer(max_features=5000)
feature = v.fit_transform(df['process_text'])
```

```
feature
```

```
<74682x5000 sparse matrix of type '<class 'numpy.int64'>'
        with 659718 stored elements in Compressed Sparse Row format>
```

```
feature_cv = feature.toarray()
```

```
feature_cv
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
feature_cv.shape
```

텍스트 데이터를 숫자로 변환하기

- **CountVectorizer** 사용하여 텍스트 데이터를 숫자 벡터로 전환(모델이 텍스트 데이터를 이해할 수 있게)
- **Max_features = 5000:** 텍스트 데이터에서 가장 자주 등장하는 5,000개의 단어만 사용, 데이터 크기를 줄이고, 모델 학습 속도 증진
- **v.fit_transform(df['process_text']):** 전처리 된 데이터를 CountVectorizer 사용해 벡터화
- **X=feature_cv[:30000]:** 전처리된 텍스트 데이터를 30000개 샘플로 제안하여 입력데이터로 사용
- **Y=df.target[:30000]:** 대응하는 타겟 데이터(감정 레이블, positive,Negative)를 30000개로 제한, 이는 모델이 학습할 출력 값이다

데이터 테스트

```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train, y_test = train_test_split(x,y, random_state=32, test_size=0.2)
```

```
X_train.shape
```

```
(24000, 5000)
```

+ Code + Markdown

```
X_test.shape
```

```
(6000, 5000)
```

```
X_train
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

데이터 셋을 학습용과 테스트용으로 분류

- **X:** 입력 데이터(텍스트 데이터가 벡터화된 배열)
- **Y:** 타겟 데이터(감정(Positive,Negative 등))
- **매개변수: test_size=0.2**(데이터 20%를 테스트용으로 사용

나머지 80%는 학습용으로 사용)

- **Random_state=32:** 데이터를 섞는 난수 시드를 설정(데이터 섞는 방식을 고정시킴)
- 결과

X_train:학습데이터(입력)

X_test: 테스트 데이터(입력)

Y_train: 학습데이터(타겟)

Y_test: 테스트 데이터(타겟)

데이터 테스트

성능 분석 테이블 분류

- **Precision (정밀도)**:
  모델이 특정 클래스로 예측한 데이터 중, 실제로 해당 클래스인 비율
  즉, "맞았다"고 한 것 중 실제로 맞은 비율

- **Recall (재현율)**:
  실제 특정 클래스에 속하는 데이터 중, 모델이 제대로 예측한 비율
  즉, "전체 정답 중 얼마나 맞췄는가"를 나타냄

- **F1-score**:
  Precision과 Recall의 조화 평균. Precision과 Recall 간 균형을 측정

- **Support**:
  각 클래스의 실제 데이터 개수

데이터 테스트

```
from sklearn.metrics import classification_report
y_pred = nb.predict(X_test)
print(f"Report : \n{classification_report(y_test,y_pred)}")
```

```
Report :
              precision    recall  f1-score   support

  Irrelevant       0.68      0.52      0.59      1011
    Negative       0.73      0.75      0.74      1519
     Neutral       0.73      0.60      0.66      1509
    Positive       0.66      0.82      0.73      1961

    accuracy                           0.70      6000
   macro avg       0.70      0.67      0.68      6000
weighted avg       0.70      0.70      0.69      6000
```

나이브 베이즈 분석

- **Positive 클래스:** 가장 높은 재현율(82%)로 긍정 데이터를 잘 찾아냄 하지만 정밀도(66%)이 낮아, 긍정으로 잘못 예측한 비율도 있음

- **Irrelevant 클래스:** 정밀도(68%)은 중간 수준이지만,재현율(52%)이 낮음 즉, 무관한 데이터를 잘 예측하지 못하는 경향이 있음

- **성능 수준:**

보통 수준: 70% 정확도는 나쁘지 않지만 Irrelevant 클래스의 재현율(52%), neutral 성능 개선이 중요

데이터 테스트

```
y_pred = tree_model.predict(X_test)
tree_report = classification_report(y_test,y_pred)
print(f"DecisionTree Report : \n{tree_report}")
```

```
DecisionTree Report :
              precision    recall  f1-score   support

  Irrelevant       0.79      0.70      0.74      1011
    Negative       0.84      0.83      0.84      1519
     Neutral       0.73      0.83      0.78      1509
    Positive       0.85      0.82      0.84      1961

    accuracy                           0.81      6000
   macro avg       0.80      0.80      0.80      6000
weighted avg       0.81      0.81      0.81      6000
```

결정 트리

- **Accuracy(정확도):81%**
- 테스트 데이터 전체 중 81%를 올바르게 예측
- **Macro Avg(평균):80%**
- 각 클래스의 Precision, Recall, F1-score의 단순 평균
- **Weighted Avg(가중 평균) :88%**
- 각 클래스의 데이터 개수(Support)를 고려한 평균

- **Decision Tree 모델 성능:**
- 81% 정확도로 준수한 성능을 보임 Positive와 Negative 클래스에서 높은 F1-score(0.84)로 긍정적, 부정적 데이터 잘 분류 하지만 Irrelevant 클래스에서 재현율(0.7)낮아 일부 무관한 데이터를 놓침. Neutral 클래스의 정밀도(0.73)이 낮아 다른 클래스를 중립으로 잘못 예측하는 경우가 있음

데이터 테스트

```
y_pdict = model.predict(X_test)
rn_report = classification_report(y_test,y_pdict)
print(f"Report : \n{rn_report}")
```

```
Report :
              precision    recall  f1-score   support

  Irrelevant       0.93      0.80      0.86      1011
    Negative       0.90      0.91      0.91      1519
     Neutral       0.82      0.90      0.86      1509
    Positive       0.90      0.90      0.90      1961

    accuracy                           0.88      6000
   macro avg       0.89      0.88      0.88      6000
weighted avg       0.89      0.88      0.88      6000
```

랜덤 포레스트

- **Accuracy(정확도):88%**
- 테스트 데이터 전체 중 88%를 올바르게 예측
- **Macro Avg(평균):88%**
- 모든 클래스에서 성능을 동일하게 반영한 평균
- **Weighted Avg(가중 평균):88%**
- 각 클래스의 데이터 개수(Support)를 고려한 평균
- **Random Forest 모델 성능:**
- 88% 정확도로, 높은 수준의 분류 성능을 보임 정밀도과 재현율의 균형이 좋아, F1-score가 전반적으로 높음

데이터 테스트

랜덤 포레스트 개선 방향

- **Irrelevant Recall(0.80) 개선**
- 무관한 데이터를 더 잘 잡아내기 위해 데이터 증강이나 더 복잡한 모델 사용
- **Neutral Precision(0.82)**
- 중립 데이터로 잘못 예측된 사례를 줄이기 위해 추가 전처리나 하이퍼 파라미터 튜닝(모델 학습 전 직접 설정해야 하는 값들) 시도

  N_estimators: 사용할 트리의 개수

  Max_features: 각 트리가 선택할 최대 특성 수

  Bootstrap: 데이터를 부트스트랩 방식으로 샘플링할지 여부

  부트스트랩 방식: 샘플 데이터를 재 샘플링 하는 기법이다. 주어진 데이터에서 중복을 허용하면서 랜덤하게 새로운 샘플을 생성하는 방식

데이터 테스트

```
predict_sentiment("i love this film")
```

```
Predicted sentiment: ['Positive']
```

```
predict_sentiment("i hate this film")
```

```
Predicted sentiment: ['Negative']
```

랜덤 포레스트

```
predict_sentiment("i love this film")
```

```
Predicted sentiment: ['Positive']
```

```
predict_sentiment("i hate this film")
```

```
Predicted sentiment: ['Irrelevant']
```

나이브 베이즈

나이브 베이즈와 랜덤 포레스트 비교

- 두 모델의 긍정적인 부분은 같지만, 부정적인 부분에서 갈린다

결론/소감

3개의 모델중 랜덤포레스트가 가장 뛰어남

- 개선을 한다면 랜덤포레스트 모델에서 하이퍼파라미터 튜닝을 사용

프로젝트 진행 소감

- 프로젝트를 진행하고 나서 AI학습과정들을 세세하게 알 수 있었고 여러가지 모델들을 학습시키고 실행 경험들을 할 수 있어서 좋았다

감사합니다