데이터과학과 인공지능

12주 소셜 데이터의 감성 예측 모델

(Sentiment Prediction of Korean Text)

유 길 상

개요 1

감성 분류 모델 구축과 예측

학습 내용

- 컴퓨터에게 한국어 영화 감상평을 주고 긍정과 부정 댓글을 학습시킨 후, 새로운 댓글에 대하여 감성을 예측할 수 있는 모델구축
- 과정: 데이터 준비 – 전처리 – 벡터화 – 로지스틱 회귀 분석 – 분석 모델 평가 - 활용

필요한 모듈

- KoNLPy 설치
- **from** konlpy.tag **import** Okt
- import re # 정규표현식(regular expression)
- from sklearn.feature_extraction.text import TfidfVectorizer
- from sklearn.linear_model import LogisticRegression

감성 분류 모델 구축과 예측

핵심 개념

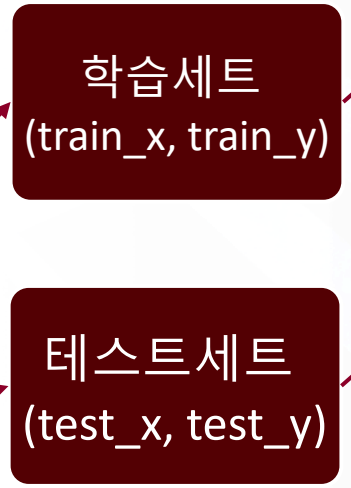
- 텍스트 전처리 → 특성 벡터화 → 머신러닝 모델 구축 및 학습/평가 프로세스 수행
- 텍스트 전처리에는 토큰화, 불용어 처리, 형태소 분석 등의 작업이 포함
- 벡터화: 텍스트 데이터를 TF-IDF 방식으로 벡터화하여 각 단어의 중요도를 수치화한 형태로 변환

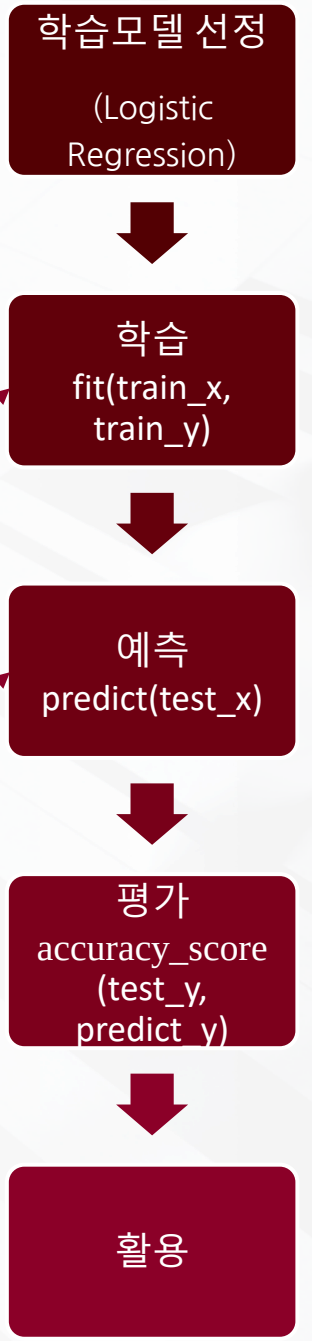
데이터 수집	특성 벡터화	감성 분석 모델 구축	분석 모델 평가	감성 예측
• 네이버 영화 댓글 수집 • 파일처리 • 결측 값 제거 • 분석 불가능 문자 제거 • 학습용, 테스트용 데이터 분리	• 형태소 기반 토큰화 • TF-IDF기반 벡터 생성	• 로지스틱 회귀 분석 모델 생성 • 최적 매개변수 도출	• 평가 데이터 벡터화 • 감성 예측 • 모델 정확도 확인	• 텍스트 입력 • 전처리 • 특성 벡터화 • 결과 예측

감성 분류 모델 구축과 예측

■ **분류 모델 만들기**

<학습 데이터>
train_x : 문제지 데이터
train_y : 문제지에 대한 정답 데이터.
<테스트 데이터>
test_x : 시험지 데이터.
test_y : 시험지에 대한 정답 데이터.





단어를 숫자로 바꾸는 과정이 필요

데이터 벡터화

텍스트는 문자라서 머신러닝 모델이 바로 이해하지 못하므로 숫자(벡터화)로 바꿔야 함

Bag of Words (BoW)

- 문장을 구성하는 단어들의 등장 횟수(빈도) 만 숫자로 표현하는 방식
- 단어의 순서나 문맥은 고려하지 않음
- 단순하지만 효과가 좋고 가장 자주 쓰이는 방식

Word2Vec

- 단어가 어떤 단어들과 함께 자주 등장하는지(문맥) 을 학습해서 단어를 벡터로 표현하는 방법
- BoW와 달리 의미적 유사성을 벡터에 반영할 수 있음
- 예: “왕 - 남자 + 여자 = 여왕” 같은 연산이 가능할 정도로 의미 정보 포함

예) Puppy cute lovely

예) You are so beautiful, you are not bad girl

데이터 벡터화

■ **카운트 기반 벡터화 Vs. TF-IDF기반 벡터화**

빈도 기반 숫자화

(2) 지방법원 및 고등법원 판결문 데이터 전처리

	body
0	1. 형사소송의 경과의정부지
1	피고인은 화성시 토지의 소
2	피고인은 창원지방법원에
3	피고인은 서울 등대문구에
4	피고인은 2020. 1. 조경 택
5	성명불상자는 전화금융사기
6	피해자 A는 베트남 출신의
7	1. 산지관리법위반, 자연공
8	피고인들은 부천시에 있는
9	피고인은 경 울산 이하 불상
10	피고인은 양산시 대로에 있
11	피고인은 경 피고인이 종업
12	피고인은 울산 울주군 6길
13	피고인은 울산지방법원에
14	피고인은 울산 남구 ○○로
15	피고인은 울산 동구 ○○로
16	피고인은 울산 울주군 ○○
17	피고인은 모닝 차량의 운전
18	피고인 박선주는 울산지방
19	피고인 임사원은 ○○ 주식
20	자동차 소유자는 국토교통
21	피고인은 ○○화재에 홈인
22	의 지위 등] 울주군시설관리
23	피고인은 울산지방법원에
24	피고인은 양산시 하북면에
25	피고인은 ○○종합건설 주식
26	피고인은 피해자 양피해, 피
27	피고인 김피고와 피해자 최
28	피고인 장회장은 민주노총

TF-IDF를 통해 빈도가 적은 명사위주로 본문 내용 전처리

가중 처벌 형법 제조 부분 위헌법률심판 경과 정부 지방 검찰청 기소 적용 누범 이유 의정부 법원 절도 미수죄 절도죄 수절 의정부교도소 집행 현관문 유의 바이저 상자 징역형 위헌 제청 대상 재판 전제 제호 무기 조의 호의 총칙 법령 특별 금고 이상 종료 면제 정한 처단형 상한 결론 헌법 죄형 법정주의 원칙 재판소 형벌 체계 균형 명확성 국회 조항 개정 결정 공통 구법 법정형 인용 방식 구체 명시 유기형 상향 하한 확대 문제 고려 반영 때문 구성 요건 표지 법률 추가 실무 소정 일반 배제 혼선 대법원 최근 과거 판례 비판력 한국 형사법 학회 논란 조문 의미 규정 조정 법관 국민 수록 문구 구성요건 법정 비례 자의 예측 곤란 과정 형성 여기 평가 재량 종전 제시 우려 안정 개별 책임 부합 판단 요청 방위 절도범 이하라 소법 직권 심판

빈도 기반 + 중요도 가중치

데이터 벡터화

카운트 기반 벡터화

- 단어에 숫자형 값을 할당할 때 각 문서에서 해당 단어가 등장하는 횟수(단어 빈도)를 부여하는 벡터화 방식
- 문서 별 단어의 빈도를 정리하여 문서 단어 행렬(DTM)을 구성하는 데 단어 출현 빈도가 높을수록 중요한 단어로 다루어 짐
- 문서 d에 등장한 단어 t의 횟수는 tf(t,d)로 표현

	따라서	데이터	분석	…	이다	하였다
문장1	3	21	15	…	4	8
문장2	13	4	5	…	7	9

데이터 벡터화

TF-IDF 기반 벡터화 (Term Frequency-Inverse Document Frequency)

- **특정 단어가 특정 문서에서 얼마나 중요한지를 나타내는지 수치화**
- **TF: 특정 문서에 많이 나타나는 단어는** 해당 문서의 단어 벡터에 **가중치를 높임**
- **IDF: 모든 문서에 많이 나타나는 단어는** 범용적으로 사용하는 단어로 취급하여 **가중치를 낮추는 방식**
- d에 등장한 단어 t의 TF-IDF
- TfidfVectorizer 클래스 $tf-idf(t,\ d) = tf(t,d) \times idf(t,d)$

  특정 단어가 문서 내에서 등장하는 빈도 ✖ 특정 단어가 전체 문서 집합에서 가지는 희소성

	따라서	데이터	분석	…	이다	하였다
문장1	0.14	0.54	0.43	…	0.16	0.21
문장2	0.12	0.49	0.65	…	0.15	0.20

감성 분류 모델

■ 감성 분석(오피니언 마이닝)

- 머신러닝 기반의 감성 분석이 늘어나고 있음(제품 평판, 정당 성향 등)
- 텍스트에서 사용자의 주관적인 의견이나 감성, 태도를 분석하는 텍스트 마이닝의 핵심 분석 기법 중 하나
- 텍스트에서 감성을 나타내는 단어를 기반으로 **긍정 또는 부정의 감성을 결정**
- 감성 사전 기반의 감성 분석은 **감성 단어에 대한 사전(관련분야의 말뭉치 구축)을 가진 상태에서** 단어를 검색하여 점수를 계산

■ 로지스틱 회귀(Logistic Regression)분석

- 일상 속 풀고자 하는 많은 문제 중에서는 두 개의 선택지 중에서 정답을 고르는 문제가 많음
- 감성분류에서는 **긍정과 부정을 결정하는 문제에 활용**
- 둘 중 하나를 결정하는 문제를 이진 분류(Binary Classification)라고 하며, 대표적인 알고리즘으로 로지스틱 회귀를 활용

전체 모델링 과정

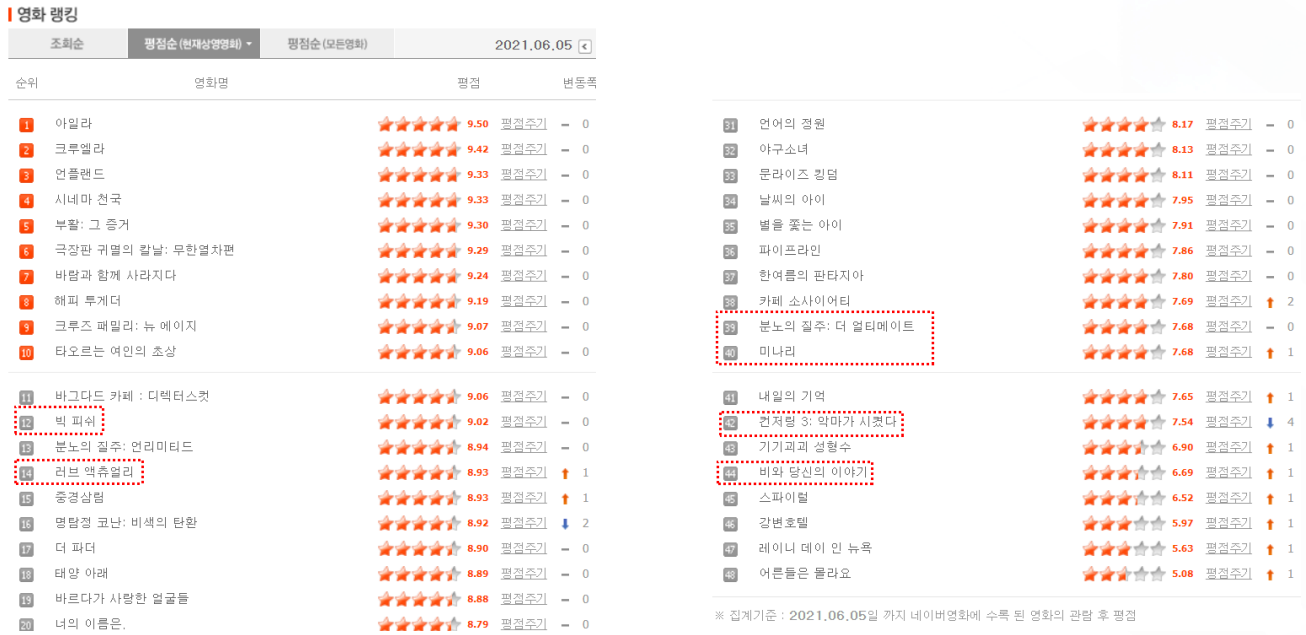
2

데이터 수집

■ 네이버 영화 댓글 크롤링

- 네이버 영화 페이지에서 리뷰를 크롤링하여 수집
- 장르별, 평이 좋은 영화와 좋지않은 영화를 골고루 선택하여 200,000개 댓글을 수집(다양한 영화 권장)

https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=cur&date=20210605



데이터 수집

긍정/부정 분류

- 파일 내용은 3개의 컬럼 (no, score, text)으로 구성
- score 컬럼은 감성 분류 클래스 값, text 는 댓글
- 긍정/부정 값은 1~10점의 평점 중에서 중립적인 평점인 5~8점은 제외하고 1~4점을 부정 감성 0으로, 9~10점을 긍정 감성 1로 표시

네이버 영화평점랭킹 선정방법

집계기준 전국기준 현재 상영되고 있는 영화 중 평점 응답자가 **300명** 이상인 경우

집계기간 전일까지의 누적 평점

표본오차 95% 신뢰수준에서 ±5.65

구분	N	Range	Minimum	Maximum	Mean	Std.Deviation
평점	26	4.39	4.75	9.14	7.59	7.59

데이터 수집

■ 수집된 데이터 확인

분노의질주.csv ×

1 to 10 of 75 entries

	score	text
0	1	뇌를 비우고 즐기면 된다
1	1	시리즈 팬들에게 주는 선물 같았던 엔딩..크.... 아련아련 ㅜㅜ
2	1	역시 분노의 질주! 이런 블록버스터는 역시 극장에서 봐야지
3	1	와 오랜만에 극장에서 봤는데 스케일 도랐 액션 개쩜
4	1	티라노사우루스는 팔이 짧아 분노의 질주 꿀잼 액션을 보고도 박수를 못쳐서 멸종됐다.
5	1	전작까지는 우주빼고 안가본데가 없었는데 이제 우주까지 나가네요..
6	1	졸라 신난다 진짜ㅋㅋㅋ 아 아침 일찍 보고 왔는데 보람있군
7	1	저스틴 린 감독이 진짜 오리지널 감독이긴 하네. 진짜 <분노의 질주> 시리즈 보는 것 같음!!!!
8	1	진짜 상상이상의 액션ㅋㅋㅋㅋ 아직 안본 분들 정말 상상도 못하셨을 겁니다
9	1	너무 재미있었어요~~ 기다린 보람이 있었어요 다음편두 더 기대할게요

Show 10 per page

1 2 3 4 5 6

수집된 Data Load

- **패키지 설치**

```
1  !pip install konlpy
```

- **데이터 불러오기**

```
2  import pandas as pd
   from konlpy.tag import Okt
   okt = Okt()
   data = pd.read_excel('movies_modify.xlsx')
```

- 데이터 확인

	score	text
0	1	굳 ㅋ
1	0	GDNTOPCLASSINTHECLUB
2	0	뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아
3	0	지루하지는 않은데 완전 막장임... 돈주고 보기에는....
4	0	3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??
...	...	...
199995	0	인간이 문제지.. 소는 뭔죄인가..
199996	1	평점이 너무 낮아서...
199997	0	이게 뭐요? 한국인은 거들먹거리고 필리핀 혼혈은 착하다?
199998	1	청춘 영화의 최고봉.방황과 우울했던 날들의 자화상
199999	0	한국 영화 최초로 수간하는 내용이 담긴 영화

200000 rows × 2 columns

데이터 정제하기 - 의미 없는 문자 제거

- **정규화를 통해 한글(자음, 모음, 한글단어 외 모두 공백으로 변환)**

 # 'ㄱ ~ '힣'까지의 문자를 제외한 나머지는 공백으로 치환, 영문: a-z| A-Z

^ : 문자열의 시작을 의미.
+ : 하나 이상의 연속된 공백을 의미

3

```
# 한글과 공백을 제외하고 모두 제거
data['text'] = data['text'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","") # 정규 표현식 수행
data['text'] = data['text'].str.replace('^ +', "") # 공백은 empty 값으로 변경
data
```

	score	text
0	1	굳 ㅋ
1	0	
2	0	뭐야 이 평점들은 나쁘진 않지만 점 짜리는 더더욱 아니잖아
3	0	지루하지는 않은데 완전 막장임 돈주고 보기에는
4	0	만 아니었어도 별 다섯 개 줬을텐데 왜 로 나와서 제 심기를 불편하게 하죠

데이터 정제하기 - 결측 값 처리

■ 비어 있는 댓글(column) 삭제 → 댓글이 남아있는 항목만 담기

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   score   200000 non-null  int64
 1   text    199834 non-null  object
dtypes: int64(1), object(1)
```

5

```
# document 열의 중복 제거
data.drop_duplicates(subset=['text'], inplace=True)
data.info()
```

```
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   score   189886 non-null  int64
 1   text    189886 non-null  object
dtypes: int64(1), object(1)
```

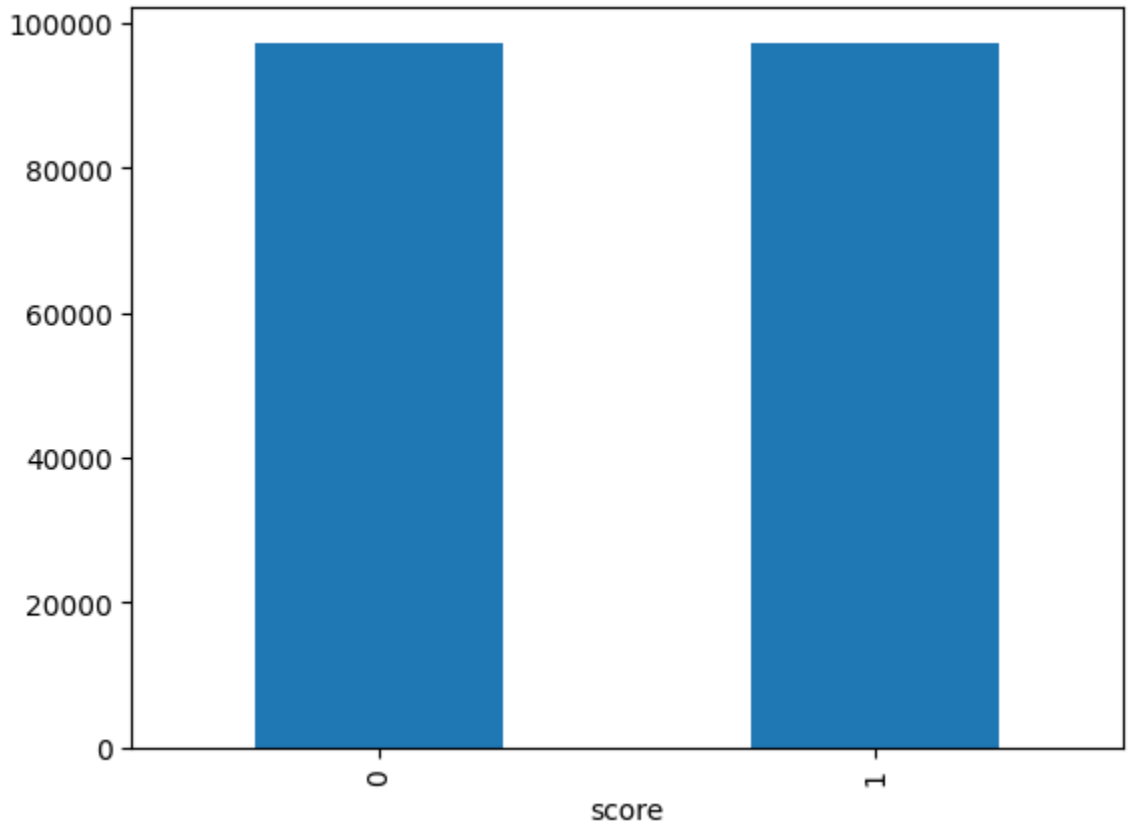
4 text가 비어 있거나(공백), NaN이거나, 문자열 길이가 0인 경우 해당 행을 삭제

```
data['text'] = data['text'].fillna('') # NaN → 빈 문자열로 변경
data['text'] = data['text'].str.strip() # 양쪽 공백 제거
data = data[data['text'] != ''] # 길이가 0인 행 삭제
```

데이터 정제하기 - 결측 값 처리

긍정 /부정 데이터 비율 확인

```
data['score'].value_counts().plot(kind = 'bar')
```



데이터 셋 분리

6 데이터를 시리즈 객체로 나누어 저장

```
exam = data['text'] # 시리즈 객체로 저장
ox = data['score']
```

<학습 데이터>
train_x : 학습용 문제지 데이터
train_y : train_x 에 대한 정답 데이터

<테스트 데이터>
test_x : 시험지 데이터.
test_y : test_x 에 대한 정답 데이터.

7 Train용 데이터셋과 Test용 데이터 셋 분리

- 예측력을 정확히 알아보기 위해 수집된 데이터를 학습용과 테스트 용으로 분리하여 진행
- 보통 20%를 테스트용으로 분리해 두고 테스트: test_size

```
from sklearn.model_selection import train_test_split

train_x, test_x, train_y, test_y = train_test_split(exam, ox , test_size=0.2, random_state=0)

print(len(train_x), len(train_y), len(test_x), len(test_y))
```

토큰화

토큰화(tokenization)는 주어진 코퍼스(corpus)에서 토큰(token)이라 불리는 단위로 나눔

■ 토큰화에서 고려해야 할 사항

- 구두점이나 특수문자를 제외해도 무방한지 확인 (ex, Ph.D, AT&T, $2,000, 11/22/1990, 45.34)
- 줄임말과 단어 내 띄어쓰기 (we’re, thx, 고대, 영끌,)
- 시대에 따라 변화는 문법과 혼란 (ex, 경윳값, 노담)
- 한국어는 띄어쓰기가 어렵고, 지켜지지 않고 있음
- 불용어 처리

■ 한국어의 토큰화

- 한글은 영어처럼 띄어쓰기 기준으로 토큰화를 할 수 없으므로, 형태소 분석기를 사용
- 한국어는 어절 독립단어가 아니므로 형태소(자립, 의존) 분해가 필요

Tokenizer - Open Korean Text 모듈

워드 클라우드 분석에 가장 적합한 모듈

```
from konlpy.tag import Okt  # okt 모듈
t = Okt()
my_text = '한국어 분석을 시작합니다 재미있어요~~'
print(t.morphs(my_text))
```

```
['한국어', '분석', '을', '시작', '합니다', '재미있어요', '~~']
```

8

```
# 한글 tokenizer
from konlpy.tag import Okt
okt = Okt()

def okt_tokenizer(text):
    return okt.morphs(text)
```

벡터화

토큰(token)의 벡터화: 컴퓨터가 텍스트를 숫자로 처리할 수 있도록 데이터의 정수화

9 TF-IDF 벡터화(단어 빈도-역 문서 빈도: Term Frequency-Inverse Document Frequency)

```
# 벡터화: 텍스트 데이터를 수치 벡터로 변환하여 머신러닝 모델에 입력할 수 있도록 준비
stopwords = ['의','가]
from sklearn.feature_extraction.text import TfidfVectorizer

tfv = TfidfVectorizer(tokenizer=okt_tokenizer,
                      ngram_range=(1,2),
                      stop_words=stopwords)
tfv.fit(train_x) # 벡터를 계산하고 단어의 중요도를 분석하여 내부 어휘 사전을 구성

# 학습 후 입력 데이터를 벡터로 변환
tfv_train_x = tfv.transform(train_x)
```

```
(0, 95759)    0.3667604743684077 5
(0, 95722)    0.2056600914393815
(0, 80096)    0.1120973735290840 7
(0, 68773)    0.2685901236921903
(0, 68430)    0.0955106207335841 4
(0, 63085)    0.3311461218137470 3
```

벡터화

토큰(token)의 벡터화: 컴퓨터가 텍스트를 숫자로 처리할 수 있도록 데이터의 정수화

■ n-gram

딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평가된 영화 인듯..

1 gram 딱히 대단한 재미도 감동도 없는데 너무 과대 평가된 영화 => Positive

2 gram 딱히 대단한 재미도 감동도 없는데 너무 과대 평가된 영화 => Negative

3 gram 딱히 대단한 재미도 감동도 없는데 너무 과대 평가된 영화 => Negative

벡터화

■ n-gram 비교

2 gram

이 수업은 내 인생에 나쁘지 않는 도움이 될것 같다
예측 결과: 긍정 감성 (확률: 67.38%)

딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나
예측 결과: 부정 감성 (확률: 85.82%)

영화 감성모델의 테스트 정확도 : 87.01%
precision: 0.8716
recall: 0.8700
f1-score: 0.8708

3 gram

이 수업은 내 인생에 나쁘지 않는 도움이 될것 같다
예측 결과: 부정 감성 (확률: 69.12%)

딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된
예측 결과: 긍정 감성 (확률: 77.17%)

영화 감성모델의 테스트 정확도 : 82.16%
precision: 0.8399
recall: 0.7973
f1-score: 0.8181

토큰화 및 TF-IDF 벡터화

■ 단어 사전 확인

```
csr_matrix: tfv_train_x

csr_matrix with shape (152778, 193679)
tfv_train_x = tfv.transform(train_x)
```

```
1  #tfv.vocabulary_ # 단어 사전 확인
```

```
{'시간': 96001,
 '내내': 29469,
 '특별한': 177273,
 '장면': 147907,
 '없이': 113517,
 '대화': 41456,
 '로만': 53737,
 '풀어가는게': 180588,
 '이렇게': 137330,
 '지루한지': 164451,
 '몰랐다': 64205,
 '사람': 85162,
 '간의': 3680,
 '심리전': 98785,
 '이나': 135342,
 '쫓고': 168250,
 '쫓기는': 168252,
 '스릴': 94295,
 '감은': 5112,
 '나름': 25620,
 '표현': 180372,
 '했다지만': 188906,
 '처음': 170297,
 '루즈': 54152,
 '함': 186819,
 '그대로': 17362,
 '끝': 24128,
 '까지': 22231,
```

감성 분석 모델 구축

■ 로지스틱 회귀(Logistic Regression)

- 데이터가 어떤 카테고리에 속하는지 분류하기 위한 통계적 방법
- 결과(종속 변수)가 범주형(예: 예/아니오, 1/0)일 때 사용

  - 이메일이 스팸인지 아닌지(1/0) 분류.

  - 환자가 질병이 있는지 없는지(1/0) 예측.
- 로지스틱 함수(시그모이드 함수)를 사용하여 확률로 결과를 출력

  - 선형 회귀의 결과값을 로지스틱 함수(sigmoid function)를 넣어 결과를 0에서 1 사이의 확률로 나타냄

  - 확률이 0.5 이상이면 "1", 0.5 미만이면 "0"으로 분류.

감성 분석 모델 구축

10 로지스틱 회귀(Logistic Regression) 모델(default) 학습

```
from sklearn.linear_model import LogisticRegression # 이진 분류용 모델
from sklearn.metrics import accuracy_score
model = LogisticRegression(random_state=0)
model.fit(tfv_train_x, train_y)    # 학습하기: 기출문제와 정답 제공
# 테스트 데이터셋으로 예측
tfv_test_x = tfv.transform(test_x)
test_predict = model.predict(tfv_test_x) # 주어진 문제 풀기
# 평가: 정확도 계산, 제출된 답안(text_predict)과 정답(text_y)을 채점
print('영화 감성모델의 테스트 정확도 : {:3.2f}%'.format(accuracy_score(test_y,
test_predict)*100))
```

```
영화 감성모델의 테스트 정확도 : 86.06%
```

분석 모델 평가

11 정답 없이 문제풀어보고 채점하기

```
# 모델 평가
tfv_test_x = tfv.transform(test_x)
test_predict = model.predict(tfv_test_x) # 주어진 문제 풀기
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
# 제출된 답안(text_predict)과 정답(text_y)을 채점
print('영화 감성모델의 테스트 정확도 :
{:3.2f}%'.format(accuracy_score(test_y, test_predict)*100))
# 정밀도, 재현율, f1-score
print('precision: {:.4f}'.format(precision_score(test_y, test_predict)))
print('recall: {:.4f}'.format(recall_score(test_y, test_predict)))
print('f1-score: {:.4f}'.format(f1_score(test_y, test_predict)))
```

```
영화 감성모델의 테스트 정확도 : 85.41%
precision: 0.8566
recall: 0.8502
f1-score: 0.8534
```

감성 예측 활용

12

```
import re
input_text = '딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나'
print(input_text)
#입력 텍스트에 대한 전처리 수행
input_text = re.compile(r'[ㄱ-ㅣ가-힣]+').findall(input_text)
input_text = [" ".join(input_text)]
# 입력 텍스트의 피처 벡터화
st_tfidf = tfv.transform(input_text)
# 최적 감성 분석 모델에 적용하여 감성 분석 평가
st_predict = model.predict(st_tfidf)
# 확률 계산
st_predict_proba = model.predict_proba(st_tfidf)
#예측 결과 출력
if st_predict == 1:
    print('예측 결과: 긍정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][1] * 100))
else:
    print('예측 결과: 부정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][0] * 100))
```

```
오랜만에 심쿵한 영화
예측 결과: ->> 긍정 감성

['돈주고 본게 아깝다']
예측 결과: ->> 부정 감성

['이걸 왜 만든거야']
예측 결과: ->> 부정 감성

['정말 오랜 만에 재미난 영화를 본 듯']
예측 결과: ->> 긍정 감성
```

```
딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나
예측 결과: 부정 감성 (확률: 53.17%)
```

하이퍼파라미터(Hyperparameter)적용

3

감성 분석 모델 구축

■ 하이퍼파라미터(Hyperparameter)

파라미터	설명	권장값
C	penalty 강도	0.1~100
penalty 방식	"l1" : 중요하지 않은 feature를 제거하는 효과 (희소 모델) "l2" : 기본값, 전체적으로 부드러운 규제 "elasticnet" : L1 + L2 혼합	l1, l2, elasticnet
solver	최적화 알고리즘(penalty에 따라 사용 가능한 solver가 다름) "liblinear" : 작은 데이터셋에 강함, l1/l2 지원 "saga" : 큰 데이터 및 sparse matrix에 강함, l1/l2/elasticnet 모두 지원	liblinear, saga
max_iter	반복 횟수 텍스트 벡터화 → feature 수가 많아지면 기본값 100은 부족할 수 있으므로 증설	200~500

감성 분석 모델 구축

■ 텍스트 데이터(LogisticRegression + TF-IDF 최적 조합

model = LogisticRegression(C=1, penalty='l1', solver='liblinear')

- 필요없는 단어(특징)를 0으로 만들어주는 효과 → feature selection
- 텍스트처럼 feature 수가 매우 많을 때, 모델의 해석력이 좋아짐 (중요 단어만 남김)

model = LogisticRegression(C=10, penalty='l2', solver='liblinear')

- 가장 안정적이고 **기본적으로 추천**되는 조합
- 대부분의 텍스트 데이터에서 성능 고르게 좋음

model = LogisticRegression(C=1, penalty='elasticnet', solver='saga', l1_ratio=0.5)

- L1 + L2의 장점 혼합 → 매우 큰 텍스트 데이터에서 효과적
- L1의 특성 선택 + L2의 안정성

감성 분석 모델 구축

10-1 하이퍼파라미터(Hyperparameter) 적용 후 학습

```
from sklearn.linear_model import LogisticRegression # 이진 분류용 모델
from sklearn.metrics import accuracy_score
model = LogisticRegression(
    random_state=0,
    C=10,
    penalty='l2',
    solver='liblinear'
)
model.fit(tfv_train_x, train_y)    # 학습하기: 기출문제와 정답 제공
# 테스트 데이터셋으로 예측
tfv_test_x = tfv.transform(test_x)
test_predict = model.predict(tfv_test_x) # 주어진 문제 풀기
# 평가: 정확도 계산, 제출된 답안(text_predict)과 정답(text_y)을 채점
print('영화 감성모델의 테스트 정확도 : {:3.2f}%'.format(accuracy_score(test_y,
test_predict)*100))
```

```
영화 감성모델의 테스트 정확도 : 86.98%
```

분석 모델 평가

11 정답 없이 문제풀어보고 채점하기

```
tfv_test_x = tfv.transform(test_x)

test_predict = model.predict(tfv_test_x) # 주어진 문제 풀기
from sklearn.metrics import accuracy_score , precision_score, recall_score, f1_score
# 제출된 답안(text_predict)과 정답(text_y)을 채점
print('영화 감성모델의 테스트 정확도 : {:3.2f}%'.format(accuracy_score(test_y, test_predict)*100))
print('precision: {:.4f}'.format(precision_score(test_y, test_predict)))
print('recall: {:.4f}'.format(recall_score(test_y, test_predict)))
print('f1-score: {:.4f}'.format(f1_score(test_y, test_predict)))
```

```
영화 감성모델의 테스트 정확도 : 86.98%
precision: 0.8720
recall: 0.8687
f1-score: 0.8704
```

감성 예측 활용

12

```
import re
input_text = '딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나'
print(input_text)
#입력 텍스트에 대한 전처리 수행
input_text = re.compile(r'[ㄱ-ㅣ가-힣]+').findall(input_text)
input_text = [" ".join(input_text)]
# 입력 텍스트의 피처 벡터화
st_tfidf = tfv.transform(input_text)
# 최적 감성 분석 모델에 적용하여 감성 분석 평가
st_predict = model.predict(st_tfidf)
# 확률 계산
st_predict_proba = model.predict_proba(st_tfidf)
#예측 결과 출력
if st_predict == 1:
    print('예측 결과: 긍정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][1] * 100))
else:
    print('예측 결과: 부정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][0] * 100))
```

```
오랜만에 심쿵한 영화
예측 결과: ->> 긍정 감성

['돈주고 본게 아깝다']
예측 결과: ->> 부정 감성

['이걸 왜 만든거야']
예측 결과: ->> 부정 감성

['정말 오랜 만에 재미난 영화를 본 듯']
예측 결과: ->> 긍정 감성
```

```
딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나
예측 결과: 부정 감성 (확률: 74.80%)
```

성능향상 - GridSearchCV 적용

4

감성 분석 모델 구축

10-2 하이퍼파라미터(Hyperparameter)튜닝 - GridSearchCV

```
# 참고: 12시간 이상 소요되므로, 가장 영향력이 큰 파라미터만 선택적으로 적용하는 것을 권장
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV  # 하이퍼파라미터 방법(그리드서치)
model = LogisticRegression(random_state=0)
params = {
    'C': [1, 5, 10, 14, 20],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 200, 500, 1000],
}
grid_cv = GridSearchCV(model, param_grid=params, cv=5, scoring='accuracy',
verbose=True)
grid_cv.fit(tfv_train_x, train_y)    # 학습하기: 기출문제와 정답 제공
print(grid_cv.best_params_, grid_cv.best_score_)# 적합한 파라메터, 최고 정확도 확인
```

```
params = {'C': [14]}
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
{'C': 14} 0.8636465884285187
```

분석 모델 평가

11-2 정답 없이 문제풀어보고 채점하기

```
tfv_test_x = tfv.transform(test_x)

test_predict = model.predict(tfv_test_x) # 주어진 문제 풀기
from sklearn.metrics import accuracy_score , precision_score, recall_score, f1_score
# 제출된 답안(text_predict)과 정답(text_y)을 채점
print('영화 감성모델의 테스트 정확도 : {:3.2f}%'.format(accuracy_score(test_y, test_predict)*100))
print('precision: {:.4f}'.format(precision_score(test_y, test_predict)))
print('recall: {:.4f}'.format(recall_score(test_y, test_predict)))
print('f1-score: {:.4f}'.format(f1_score(test_y, test_predict)))
```

```
영화 감성모델의 테스트 정확도 : 87.01%
precision: 0.8716
recall: 0.8700
f1-score: 0.8708
```

감성 예측 활용

12-2

```
import re
input_text = '딱히 대단한 재미도 감동도 없는데 ~! 너무 과대 평과된 영화 중 하나'
print(input_text)
#입력 텍스트에 대한 전처리 수행
input_text = re.compile(r'[ㄱ-ㅣ가-힣]+').findall(input_text)
input_text = [" ".join(input_text)]
# 입력 텍스트의 피처 벡터화
st_tfidf = tfv.transform(input_text)
# 최적 감성 분석 모델에 적용하여 감성 분석 평가
st_predict = grid_cv.best_estimator_.predict(st_tfidf)
#  확률  계산
st_predict_proba = grid_cv.best_estimator_.predict_proba(st_tfidf)
#예측  결과  출력
if st_predict == 1:
    print('예측 결과: 긍정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][1] * 100))
else:
    print('예측 결과: 부정 감성 (확률: {:.2f}%)'.format(st_predict_proba[0][0] * 100))
```

```
•••  이 수업을 수강한게 너무 좋다
     예측 결과: 긍정 감성 (확률: 96.42%)
```

Pickle활용(학습 모델 저장)

5

벡터화 결과 및 모델 저장

벡터화 결과 저장해 두고 사용하기

1 Pickle 모듈

- pickle 모듈을 사용하여 tfv 데이터와 학습모델을 pickle 파일에 저장하기 → 재 학습 시간 단축
- 'wb'는 write binary(바이너리 쓰기 모드)
- pickle.dump(tfv_train_x, f): tfv 내용을 f 파일에 저장

```
import pickle
save_objects = {'tfv': tfv,'model': model} # 저장할 객체 묶기

# pickle 저장
with open('sentiment_model_Okt_TFv_ft.pickle', 'wb') as f:
    pickle.dump(save_objects, f)
```

벡터화 결과 및 모델 저장

벡터화 결과 저장해 두고 사용하기

2 Pickle 불러오기 (load)

```
# 모델 + 벡터 불러오기
import pickle
from konlpy.tag import Okt
okt = Okt()

def okt_tokenizer(text):
    return okt.morphs(text)

with open('sentiment_model_Okt_TFv_L2model.pickle', 'rb') as f:
    loaded = pickle.load(f)

tfv = loaded['tfv']
model = loaded['model']
```

12 감성 예측하기 → 12(Hyperparameter), 12-2(grid_cv)


Thank you