SnS 크롤링을 통한 햄버거 브랜드별 평가 감성분석

김란
김민지
김명섭
박태준
한유진

CONTENTS

01
주제선정

02
데이터 개요

03
연구방법

04
데이터 수집 / 전처리

05
예측 모델 구현

06
모델링 예측결과

주제 선정

01

주제 선정 이유

리뷰 뒤에 숨겨진 긍정적 또는 부정적인 감정을 예측할 수 있는 모델을 만들 수 있을까? 만들 수 있다면, 이러한 예측 모델을 이용하여 부정적 요소와 긍정적 요소를 마케팅에 활용할 수 있자 않을까 생각하여 주제를 선정하였다.

가장 친숙하고 동네마다 있으며, 대표적인 브랜드들이 생각이 나는 햄버거를 주제로 선정하였고, 2023 버거 프랜차이즈 트렌드 리포트에서 버거 프랜차이즈 이용 순위와 한 번 이라도 취식 경험이 있는지 순위를 보았을 때, 다른 점을 발견하여 실제 이용해 본 사람들의 리뷰를 데이터로 선정하였다.

출처 : 버거 프랜차이즈 트렌드 리포트 2023

〈그래픽=채민선 기자〉 ©중기이코노미

데이터 개요

02

인스타그램

#롯데리아 / #맥도날드 / #버거킹
검색하였을 때 가장 최근 게시물 10000개 수집

네이버 리뷰

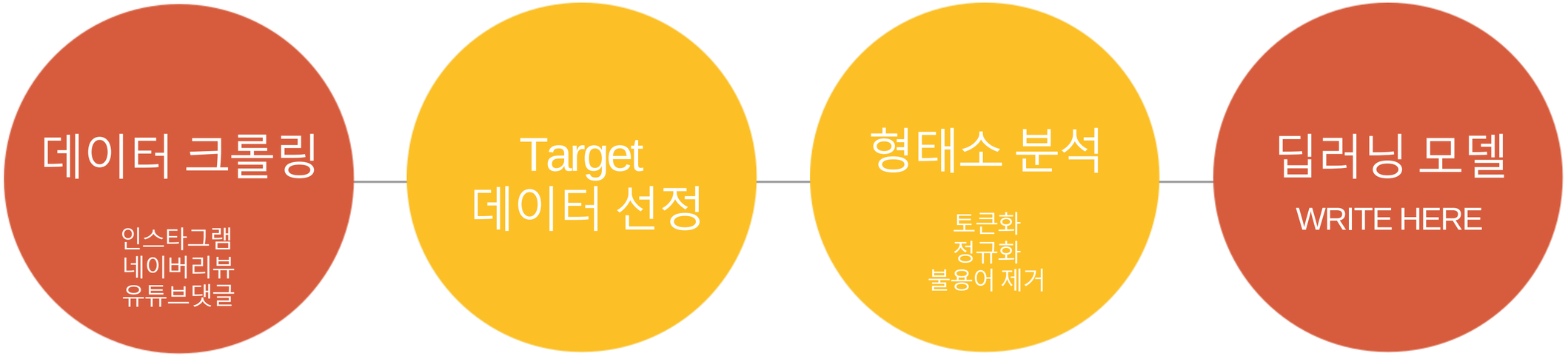
각 브랜드 3사별 DT매장
50개 매장의 최근 리뷰 200개씩
총 10000개 수집

유튜브 댓글

각 브랜드 3사별 평가 댓글과
위생 관련 뉴스 댓글
time.sleep(1000) 기준으로
댓글 수집

연구 방법

03



크롤링	라벨링	전처리	딥러닝
BeautifulSoup , Selenium	모든 데이터의 30% 긍정 1 / 부정 0	Konlpy(코엔엘파이) Okt / Mecab	Tensorflow GRU

데이터 수집 / 전처리

04

데이터 수집 (Crawling)

롯데리아 역삼점 햄버거

★4.22/5 · 방문자리뷰 3,484 · 블로그리뷰 129

출발 도착

저장 거리뷰 공유

배달

홈 메뉴 리뷰 사진

#맥도날드

게시물

94.1만

팔로우

인기 게시물

신메뉴가 나왔다는 소식에 바로 가서 먹어본
진도대파크림크로켓버거!

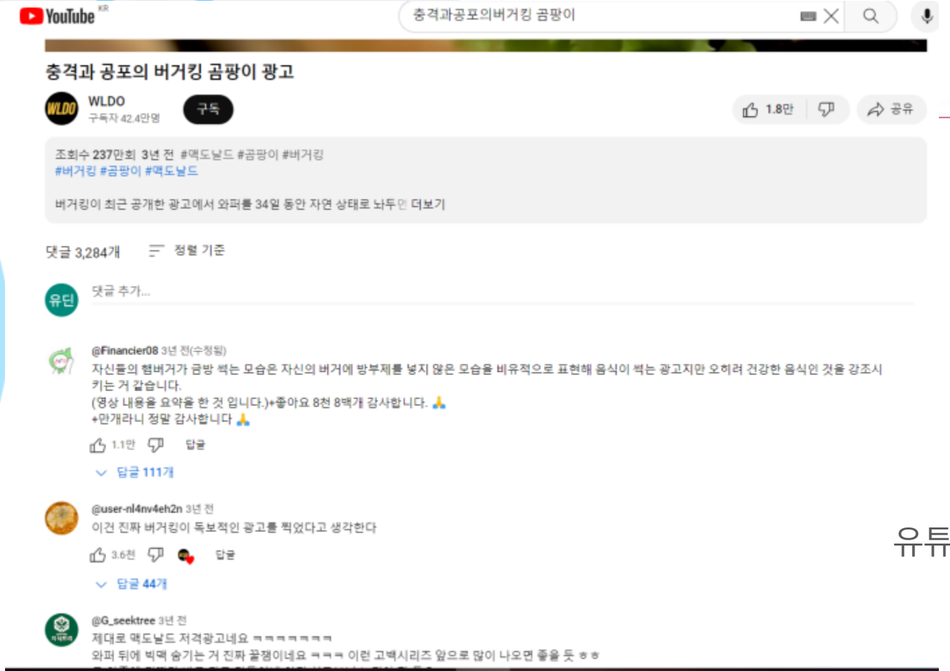
고기패티와 크로켓 두가지 들어가서
튀김맛과 패티맛이 다 느껴지고
대파의 향기가 강하게 나서 좋았어요!!

소스도 두가지 소스인 것 같은데
진짜 한번 가서 먹어보는 것을 추천합니다 !!

크로켓은 크림과 감자와 야채로 만들어서
겉은 바삭하고 속은 메쉬드 포테이토
죽 느낌나서 호불호 나뉠 것 같아요 !

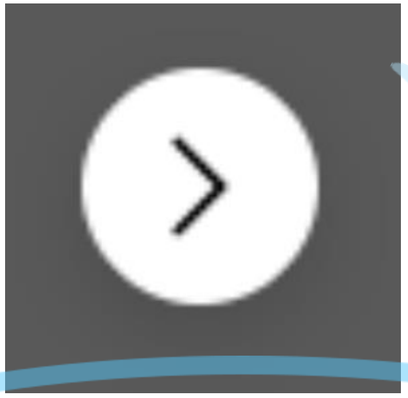
56분

인스타그램



유튜브

04



데이터 수집 - 인스타그램

```python
# 검색어 조건에 따른 url 생성
def insta_searching(word):
    url = "https://www.instagram.com/explore/tags/" + str(word)
    return url
```

```python
# 열린 페이지에서 첫 번째 게시물 클릭 + sleep 메소드 통하여 시차 두기
def select_first(driver):
    first = driver.find_element(By.CSS_SELECTOR, '._aagu')
    first.click()
    time.sleep(3)
```

```python
# 본문 내용, 해시태그 가져오기
def get_content(driver):
    html = driver.page_source
    soup = BeautifulSoup(html, 'lxml')
    # 본문 내용
    try:
        content = soup.select('div._a9zs')[0].text
    except:
        content = ''
    # 해시태그
    tags = re.findall(r'#[^\s#,\\]+', content)
    data = [content, tags]
    return data
```

```python
# 첫 번째 게시물 클릭 후 다음 게시물 클릭
def move_next(driver):
    right = driver.find_element(By.CSS_SELECTOR, "svg[aria-label='다음']")
    right.click()
    time.sleep(3)
```

```python
# 크롬 브라우저 열기
driver = webdriver.Chrome()
driver.get('https://www.instagram.com')
time.sleep(3)
```

```python
# 인스타그램 로그인
email = '' # 아이디 입력
input_id = driver.find_element(By.XPATH, '//*[@id="loginForm"]/div/div[1]/div/label/input')
input_id.clear()
input_id.send_keys(email)
password = '' # 비밀번호 입력
input_pw =driver.find_element(By.XPATH, '//*[@id="loginForm"]/div/div[2]/div/label/input')
input_pw.clear()
input_pw.send_keys(password)
input_pw.submit()
time.sleep(5)
```

```python
# 게시물 조회할 검색 키워드 입력 요청
word = input("검색어를 입력하세요 : ")
word = str(word)
url = insta_searching(word)
```

검색어를 입력하세요 : 맥도날드

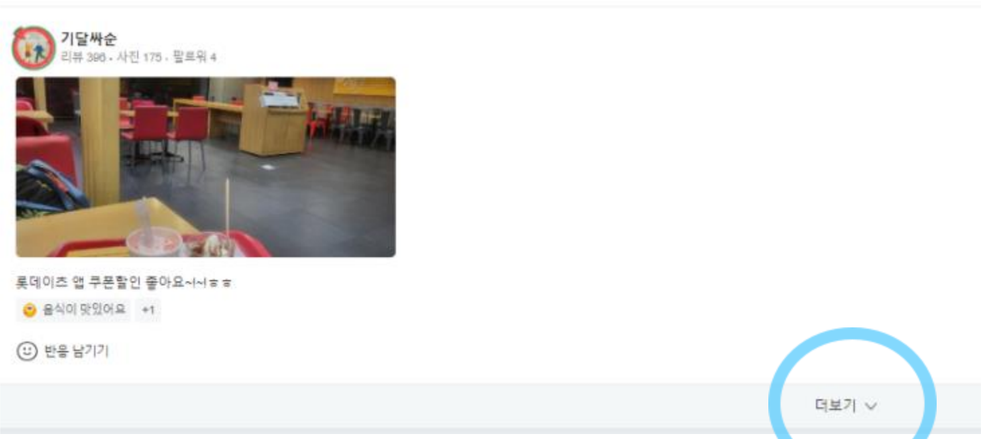
```python
# 검색 결과 페이지 열기
driver.get(url)
time.sleep(8)
```

```python
# 첫번째 게시물 클릭
select_first(driver)
```

```python
# 게시물 수집 시작
results = []
target = 10000
for i in range(target):
    try:
        data = get_content(driver)
        results.append(data)
        move_next(driver)
    except:
        time.sleep(2)
        move_next(driver)
```

04

데이터 수집 - 네이버지도 리뷰




```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import re
from bs4 import BeautifulSoup
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
```

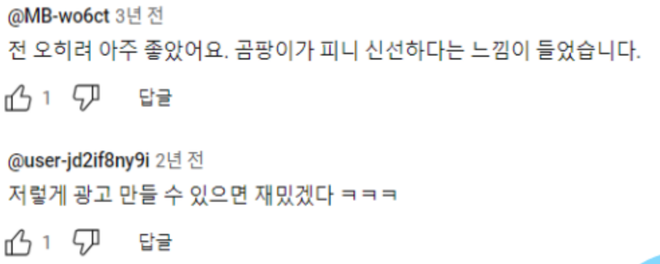
```
driver = webdriver.Chrome()
```

```
results = []
for i in place :
    naver_map_search_url = f'https://map.naver.com/v5/search/{i}/place' # 검색 url 만들기
    driver.get(naver_map_search_url) # 검색 url 접속 = 검색하기
    time.sleep(17)
    cu = driver.current_url
    res_code = re.findall(r"place/(₩d+)", cu)
    final_url = f'https://pcmap.place.naver.com/restaurant/{res_code[0]}/review/visitor/'
    driver.get(final_url)
    time.sleep(5)
    for j in range(19) :
        next = driver.find_element(By.CSS_SELECTOR, ".lfH3O")
        next.click()
        time.sleep(3)

        html = driver.page_source
        soup = BeautifulSoup(html, 'lxml')
        time.sleep(1)

    one_review = soup.find_all('div', attrs = {'class':'ZZ4OK'})
    for k in range(len(one_review)):
        try:
            review_content = one_review[k].find('span', attrs = {'class':'zPfVt'}).text
        except: # 리뷰가 없다면
            pass
        results.append(review_content)
```

04



데이터 수집 - 유튜브

```
driver = webdriver.Chrome()
url = 'https://www.youtube.com/watch?v=-omxjG2sl04'
driver.get(url)
last_page_height = driver.execute_script("return document.documentElement.scrollHeight")
while True:
    driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")
    time.sleep(3.0)
    new_page_height = driver.execute_script("return document.documentElement.scrollHeight")
    if new_page_height == last_page_height:
        break
    last_page_height = new_page_height
time.sleep(1000)
html_source = driver.page_source
driver.close()
soup = BeautifulSoup(html_source, 'lxml')

mains = soup.select('div#main')
youtube_user_IDs = []
youtube_comments = []
for main in mains :
    youtube_user_IDs.append(main.select('div#header div#header-author > h3 > a#author-text > span'))
    youtube_comments.append(main.select('div#comment-content'))
```

```
str_youtube_userIDs = []
str_youtube_comments = []

for i in range(len(youtube_user_IDs)):
    try:
        str_tmp = str(youtube_user_IDs[i][0].text)
    #     print(str_tmp)
        str_tmp = str_tmp.replace('₩n', '')
        str_tmp = str_tmp.replace('₩t', '')
        str_tmp = str_tmp.replace('                    ','')
        str_youtube_userIDs.append(str_tmp)

        str_tmp = str(youtube_comments[i][0].text)
        str_tmp = str_tmp.replace('₩n', '')
        str_tmp = str_tmp.replace('₩t', '')
        str_tmp = str_tmp.replace('                    ', '')

        str_youtube_comments.append(str_tmp)

    except Exception as e:
        print(e)

for i in range(len(str_youtube_userIDs)):
    print(str_youtube_userIDs[i], str_youtube_comments[i])
```

04

데이터 라벨링

인스타그램 + 네이버지도 리뷰 + 유튜브 댓글
각 브랜드 별 파일 합친 뒤

약 22000개의 리뷰 중 30%인 7000개의 리뷰를
target 데이터로 선정하여
긍정을 1, 부정을 0으로 라벨링 진행

	content	target
13396	새벽에 드라이브 쓰루로 방문하기 좋아오!	1
13397	안정적인 맛	1
13398	궁금한게 있는데 맥도날드 알바 왜 인상쓰고 일함? 가는 곳마다 말 잘못걸면 한대 처 맞을거 같아	0
13399	굿굿굿 다음에또갈께요~	1
13400	앱할인 굿	1
13401	패티를 레어로 익혔네 ㅋㅋ	0
13402	조아용 마싯어요	1
13403	잘 먹었습니다	1
13404	맥스파이시 크리미 어니언 맛있어요. 상하이보다 풍미가 더 있어요. 크리미하면서 더 매콤해요. 소스랑 베이컨 더 들어간거같아요. 그냥 맥스파이시 상하이보다 천원 이상 더 주고 먹을거 같진 않아요. 맥도날드도 이제 비싸네요.	0
13405	오늘의 점심#맥도날드#상하이스파이시버거세트	
13406	나 호주 갔을때 햄버거 하나 시켰는데 세입만에 다먹어서 어이가 없었다 매뉴판에서는 개크게 그려놓고 나오니까 개아담하네 사기날드 다신안가 버거킹만 가자~	0
13407	주차공간이 항상 차요. 그래도 드라이브 스루는 가능	1
13408	여러가지 주문후에 함께 담아달라했더니 종이가방을 던져주듯이 주더라~ 바빠서 그러면 이해하겠지만 자기들끼리 떠들면서 말이다~패스트푸드라서 다행이랄까, 식당이였으면 다시는 안가고 싶다는ㅜ	0
13409	맥도날드는 역시 치즈버거!!	1
13410	리뉴얼 되고 처음 방문했는데너무 깨끗해요늦은 시간 커피 마시기 딱 좋구요^^	1

04

데이터 라벨링 - 각 브랜드 3사별로 나눈 이유

```
loaded_model = load_model('best_model.h5')
print("₩n 테스트 정확도: %.4f" % (loaded_model.evaluate(X_test, y_test)[1]))
```

맥도날드(train) 롯데리아, 버거킹 (test)

```
218/218 [==============================] - 22s 100ms/step - loss: 0.6153 - acc: 0.7890
롯데리아 테스트 정확도: 0.7890

219/219 [==============================] - 22s 101ms/step - loss: 0.5888 - acc: 0.7781
버거킹 테스트 정확도: 0.7781
```

롯데리아(train) 맥도날드, 버거킹 (test)

```
219/219 [==============================] - 13s 57ms/step - loss: 0.4359 - acc: 0.8393
맥도날드 테스트 정확도: 0.8393

219/219 [==============================] - 12s 57ms/step - loss: 0.4057 - acc: 0.8592
버거킹 테스트 정확도: 0.8592
```

버거킹(train) 롯데리아, 맥도날드 (test)

```
218/218 [==============================] - 23s 103ms/step - loss: 0.4752 - acc: 0.7876

 테스트 정확도: 0.7876

219/219 [==============================] - 34s 151ms/step - loss: 0.3646 - acc: 0.8574

 테스트 정확도: 0.8574
```

04

7000개의 라벨링 한 리뷰들을 train / test로 나누는 비율을
7 : 3 , 8 : 2, 9 : 1 로 차례로 진행.
train의 비율을 높일수록 정확도가 높아짐을 알 수 있다.
따라서 test set을 따로 나누지 않고,
라벨링한 모든 데이터를 train set으로 정하고 모델 구현하였다.

데이터 전처리

```
loaded_model = load_model('best_model.h5')
print("₩n 테스트 정확도: %.4f" % (loaded_model.evaluate(X_test, y_test)[1]))
```

맥도날드

7 : 3

```
66/66 [==============================] - 6s 86ms/step - loss: 0.2994 - acc: 0.8893

 테스트 정확도: 0.8893
```

8 : 2

```
44/44 [==============================] - 4s 79ms/step - loss: 0.2903 - acc: 0.8955
테스트 정확도: 0.8955
```

9 : 1

```
22/22 [==============================] - 2s 85ms/step - loss: 0.2657 - acc: 0.9083
테스트 정확도: 0.9083
```

롯데리아

7 : 3

```
66/66 [==============================] - 8s 110ms/step - loss: 0.2917 - acc: 0.8885
테스트 정확도: 0.8885
```

8 : 2

```
44/44 [==============================] - 3s 62ms/step - loss: 0.2454 - acc: 0.9046
테스트 정확도: 0.9046
```

9 : 1

```
22/22 [==============================] - 3s 91ms/step - loss: 0.2478 - acc: 0.9127
테스트 정확도: 0.9127
```

버거킹

7 : 3

```
66/66 [==============================] - 7s 95ms/step - loss: 0.4232 - acc: 0.8706

 테스트 정확도: 0.8706
```

8 : 2

```
44/44 [==============================] - 10s 207ms/step - loss: 0.2847 - acc: 0.8831

 테스트 정확도: 0.8831
```

9 : 1

```
22/22 [==============================] - 7s 282ms/step - loss: 0.2908 - acc: 0.8926

 테스트 정확도: 0.8926
```

04

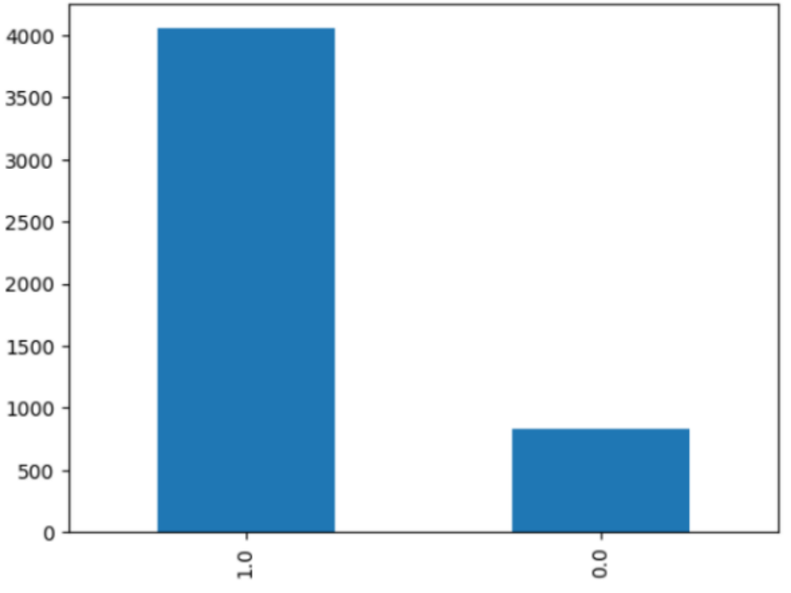
데이터 전처리

```
train_data, test_data = train_test_split(data1, test_size = 0.3, random_state = 42)
print('훈련용 리뷰의 개수 :', len(train_data))
print('테스트용 리뷰의 개수 :', len(test_data))
```

```
훈련용 리뷰의 개수 : 4887
테스트용 리뷰의 개수 : 2095
```

```
train_data['target'].value_counts().plot(kind = 'bar')
```

```
<Axes: >
```



긍정 / 부정 value_counts()

```
train_data['content'] = train_data['content'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
train_data['content'].replace('', np.nan, inplace=True)
print(train_data.isnull().sum())
train_data = train_data.dropna(how='any') # Null 값 제거
print(train_data['content'].head(20))
print('전처리 후 훈련용 샘플의 개수 :',len(train_data))
```

```
content    13
target      0
dtype: int64
11835                                롯데리아 햄벅도 너무 비싸
4721                                                 굿
1584                                               좋아요
13087                                  드라이브스루라 좋아요
5606                              허쉬 핫초코 따뜻하고 적당한 양 좋네요
13455    롯데리아에서 귀여운 포켓몬 하우스 데려왔어요피카츄는 없네요롯데리아포켓몬하우스하우스고라파덕
9408                                               빠름
81                                매번 먹는 단골 롯데이아ㅎㅎㅎㅎ쪼아요
7450     미치겠다 아직도 롯데잇츠 서버폭발했나앱도 들어가지 못했다아니 롯데잇츠 롯데리아 대기...
2659           햄버거 먹으러 속초 맛집 롯데리아 롯데속초리조트 왔어요 새우버거 완전 맛있어
4437                                               굿
810      마라가들어간 햄버거라고 새로 출시되었다고해서 함사서 먹어요먹팔해요 먹스타맞팔 먹스타...
3369                             롯데리아 비빔한마당 전빔벅전주비빔라이스버거
7355                                               굿
```

```
test_data['content'] = test_data['content'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","") # 정규 표현식 수행
test_data['content'].replace('', np.nan, inplace=True) # 공백은 Null 값으로 변경
print(train_data.isnull().sum())
test_data = test_data.dropna(how='any') # Null 값 제거
print('전처리 후 테스트용 샘플의 개수 :',len(test_data))
```

```
content    0
target     0
dtype: int64
전처리 후 테스트용 샘플의 개수 : 2089
```

train / test 전처리

04

Mecab을 사용하여 형태소 분석을 한 뒤,
단어의 빈도 수가 높은 20개의 단어를 추출해 보았다.

라벨링한 긍정 리뷰와 부정 리뷰의 평균 길이 수를 구해보니
긍정 리뷰보다 부정 리뷰의 평균 길이가 더 길다.
=> 부정 단어보다 긍정 단어의 영향력이 더 크다.

데이터 전처리

```
stopwords = ['도', '는', '다', '의', '가', '이', '은', '한', '에', '하', '고', '을', '를', '인', '듯', '과', '와', '네', '들', '듯', '지', '임', '게']
```

```
train_data['tokenized'] = train_data['content'].apply(mecab.morphs)
train_data['tokenized'] = train_data['tokenized'].apply(lambda x: [item for item in x if item not in stopwords])
```

```
test_data['tokenized'] = test_data['content'].apply(mecab.morphs)
test_data['tokenized'] = test_data['tokenized'].apply(lambda x: [item for item in x if item not in stopwords])
```

```
negative_words = np.hstack(train_data[train_data.target == 0]['tokenized'].values)
positive_words = np.hstack(train_data[train_data.target == 1]['tokenized'].values)
```

```
negative_word_count = Counter(negative_words)
print(negative_word_count.most_common(20))
```

```
[('롯데', 435), ('버거', 419), ('리아', 405), ('먹', 387), ('는데', 222), ('맛', 206), ('안', 200), ('어', 178), ('었', 175), ('라이스버거', 174), ('있', 173), ('없', 162), ('햄버거', 151),
```

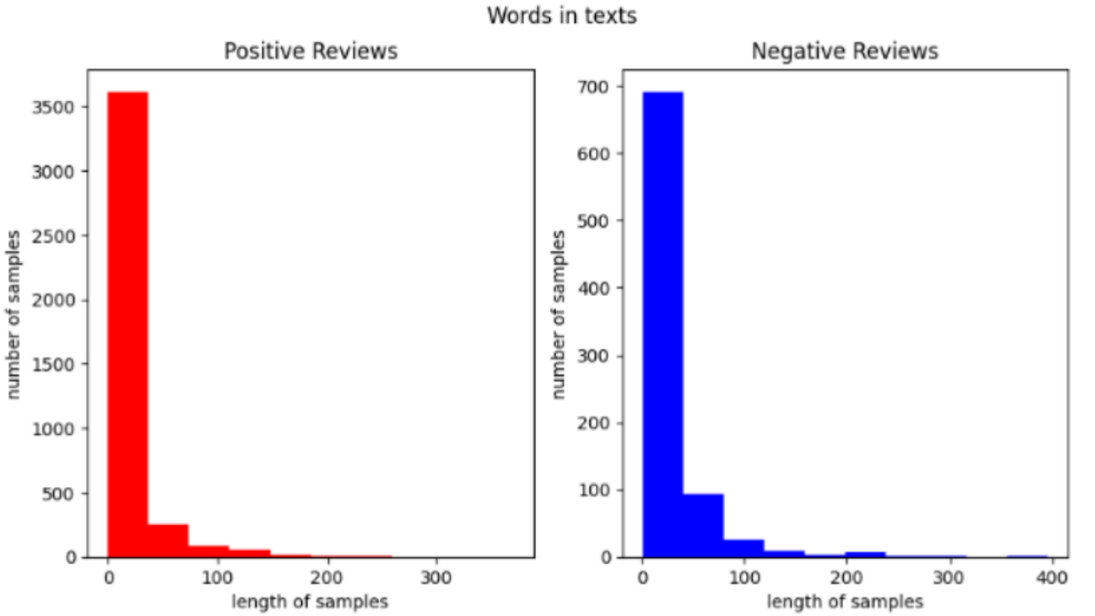
```
positive_word_count = Counter(positive_words)
print(positive_word_count.most_common(20))
```

```
[('롯데', 2033), ('리아', 1803), ('버거', 1362), ('좋', 1298), ('먹', 1269), ('맛있', 1040), ('아요', 850), ('어요', 771), ('굿', 746), ('햄버거', 605), ('맛', 449), ('그램', 448), ('있', 43
```

```
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(10,5))
text_len = train_data[train_data['target']==1]['tokenized'].map(lambda x: len(x))
ax1.hist(text_len, color='red')
ax1.set_title('Positive Reviews')
ax1.set_xlabel('length of samples')
ax1.set_ylabel('number of samples')
print('긍정 리뷰의 평균 길이 :', np.mean(text_len))

text_len = train_data[train_data['target']==0]['tokenized'].map(lambda x: len(x))
ax2.hist(text_len, color='blue')
ax2.set_title('Negative Reviews')
fig.suptitle('Words in texts')
ax2.set_xlabel('length of samples')
ax2.set_ylabel('number of samples')
print('부정 리뷰의 평균 길이 :', np.mean(text_len))
plt.show()
```

```
긍정 리뷰의 평균 길이 : 14.999010390895597
부정 리뷰의 평균 길이 : 25.651442307692307
```



예측 모델
- GRU

05

리뷰의 길이가 390 이하인 샘플만 가져와
서로 다른 길이의 샘플들의 길이를 동일하게 맞춰주는 패딩 진행

모델은 다대일 구조의 LSTM을 개선한 모델인 GRU를 사용하여
두 개의 선택지 (긍정 / 부정) 중 하나를 예측하는 이진 분류 문제 수행하는 모델 구현

GRU 모델

```
def below_threshold_len(max_len, nested_list):
  count = 0
  for sentence in nested_list:
    if(len(sentence) <= max_len):
        count = count + 1
  print('전체 샘플 중 길이가 %s 이하인 샘플의 비율: %s'%(max_len, (count / len(nested_list))*100))
```

```
max_len = 390
below_threshold_len(max_len, X_train)
```

```
전체 샘플 중 길이가 390 이하인 샘플의 비율: 99.97948297086582
```

```
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

```
from tensorflow.keras.layers import Embedding, Dense, GRU
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

embedding_dim = 100
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(1, activation='sigmoid'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=64, validation_split=0.2)
```

```
loaded_model = load_model('best_model.h5')
print("테스트 정확도: %.4f" % (loaded_model.evaluate(X_test, y_test)[1]))
```

05

긍정 / 부정 판단 - 예측해보기

```
def sentiment_predict(new_sentence):
  new_sentence = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣 ]','', new_sentence)
  new_sentence = mecab.morphs(new_sentence)
  new_sentence = [word for word in new_sentence if not word in stopwords]
  encoded = tokenizer.texts_to_sequences([new_sentence])
  pad_new = pad_sequences(encoded, maxlen = max_len)

  score = float(loaded_model.predict(pad_new))
  if(score > 0.5):
    print("{:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
  else:
    print("{:.2f}% 확률로 부정 리뷰입니다.".format((1 - score) * 100))
```

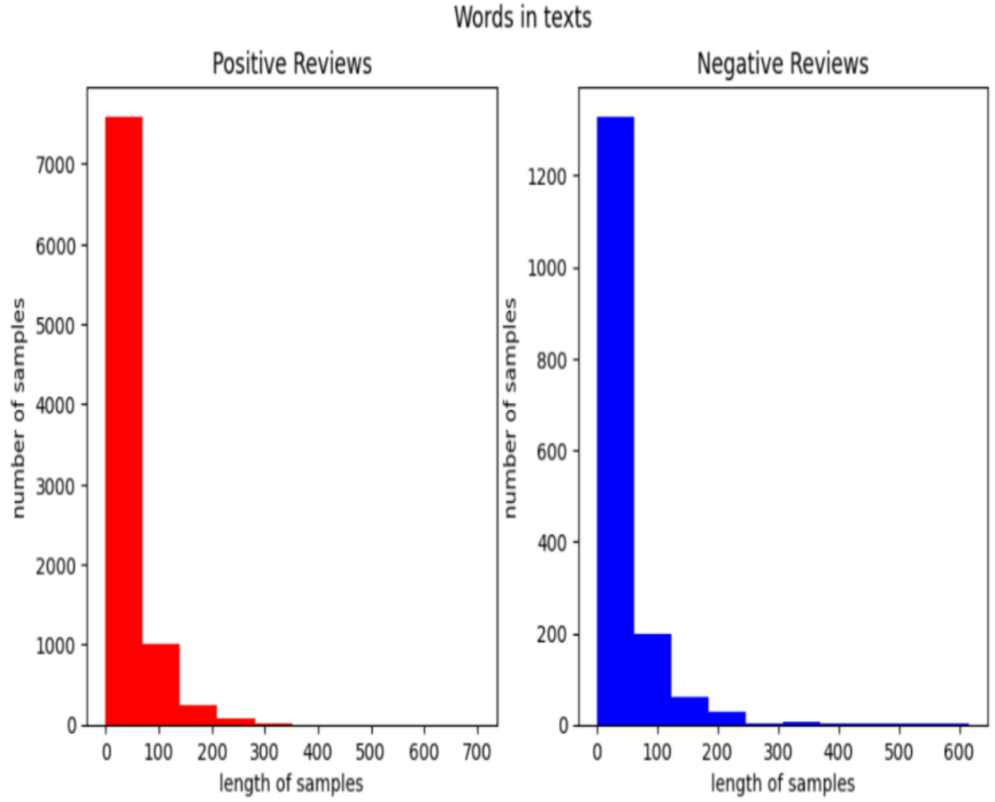
```
sentiment_predict('친절합니다.~~맛은기본으로 맛나요~~')
```

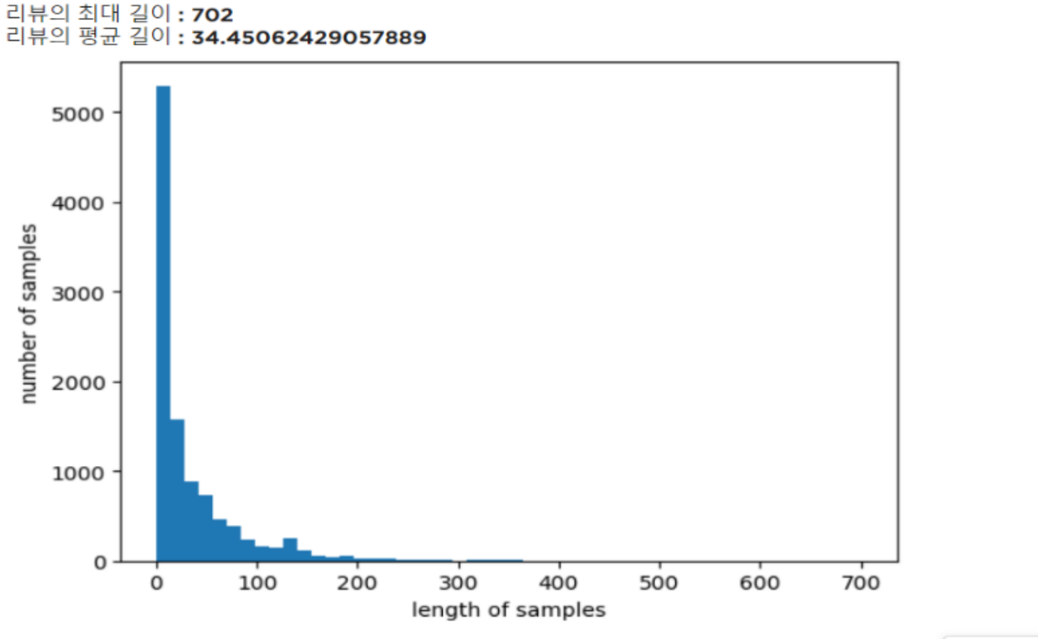
```
1/1 [==============================] - 1s 525ms/step
99.81% 확률로 긍정 리뷰입니다.
```

모델링
예측 결과

06

모델로 예측된 라벨링 결과 - 롯데리아





전체 리뷰의 최대 / 평균 길이

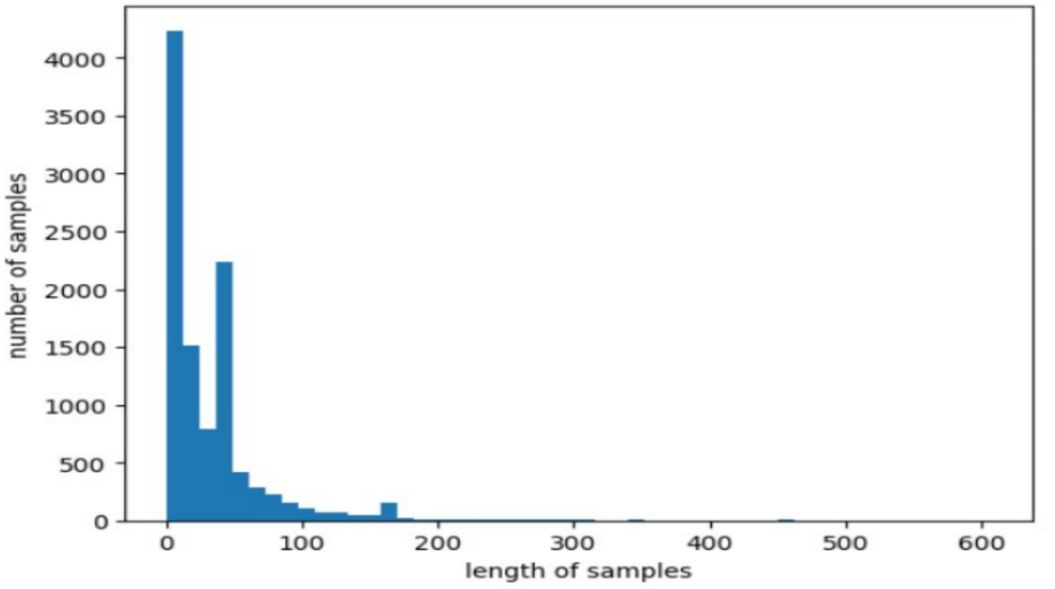
143/143 [==============================] - 15s 102ms/step - loss: 0.1582 - acc: 0.9439
테스트 정확도: 0.9439

모델의 정확도

06

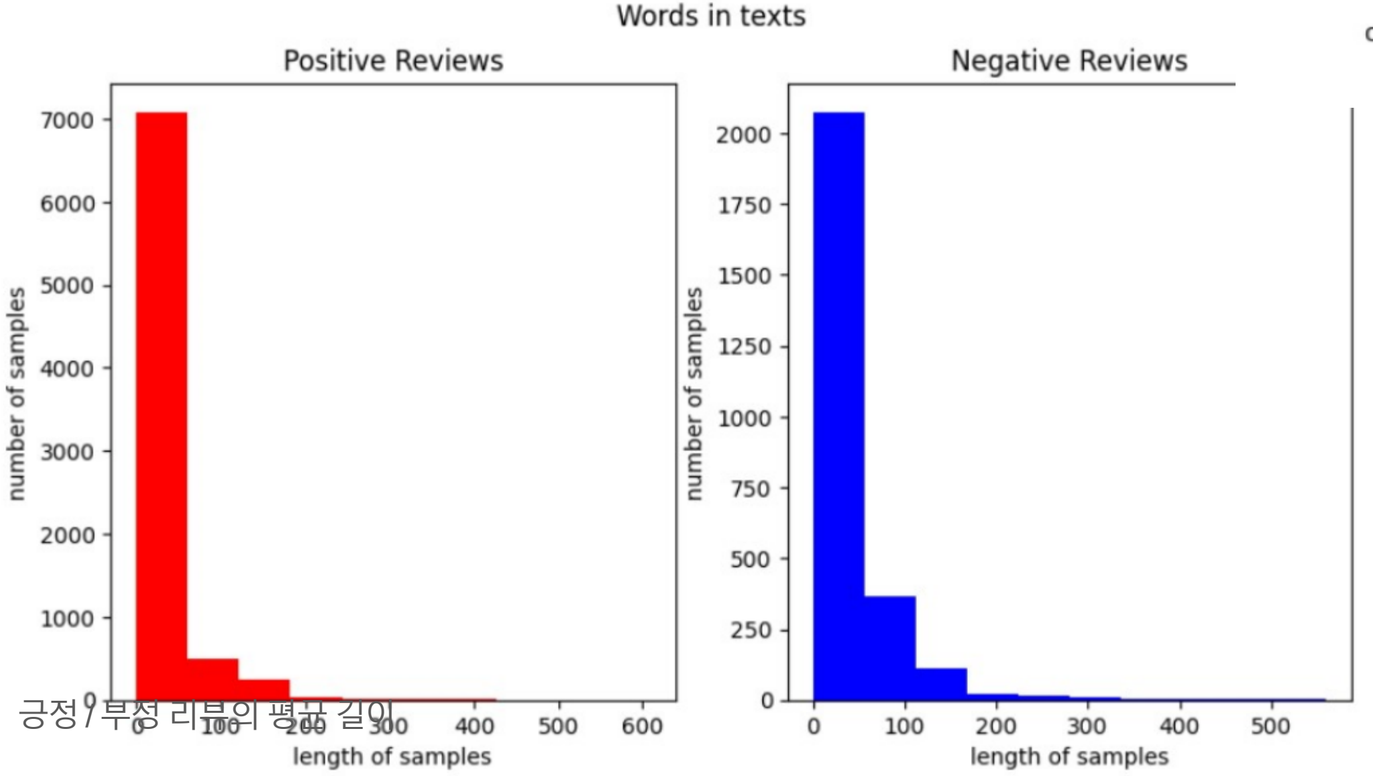
모델로 예측된 라벨링 결과 - 맥도날드

```
리뷰의 최대 길이 : 608
리뷰의 평균 길이 : 32.167956434508454
```



전체 리뷰의 최대 / 평균 길이

```
긍정 리뷰의 평균 길이 : 29.99758392675483
부정 리뷰의 평균 길이 : 38.72493276988091
```



긍정 / 부정 리뷰의 평균 길이

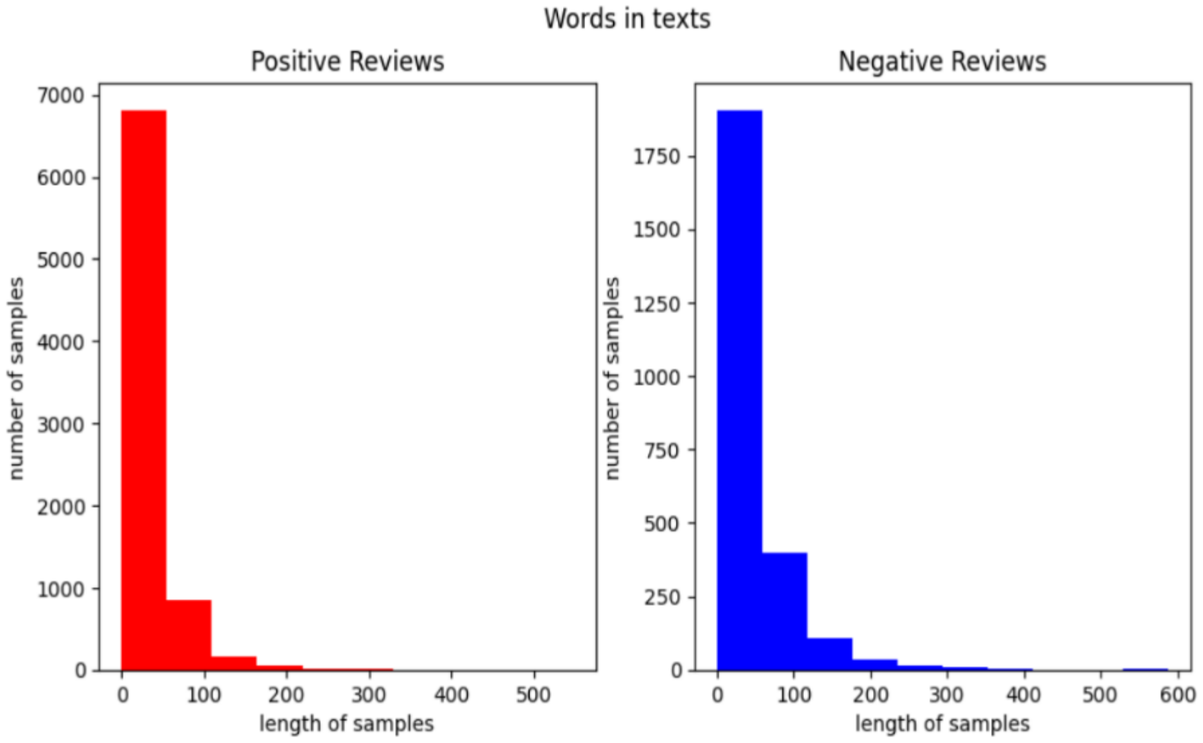
```
141/141 [==============================] - 9s 58ms/step - loss: 0.1809 - acc: 0.9410
테스트 정확도: 0.9410
```

모델의 정확도

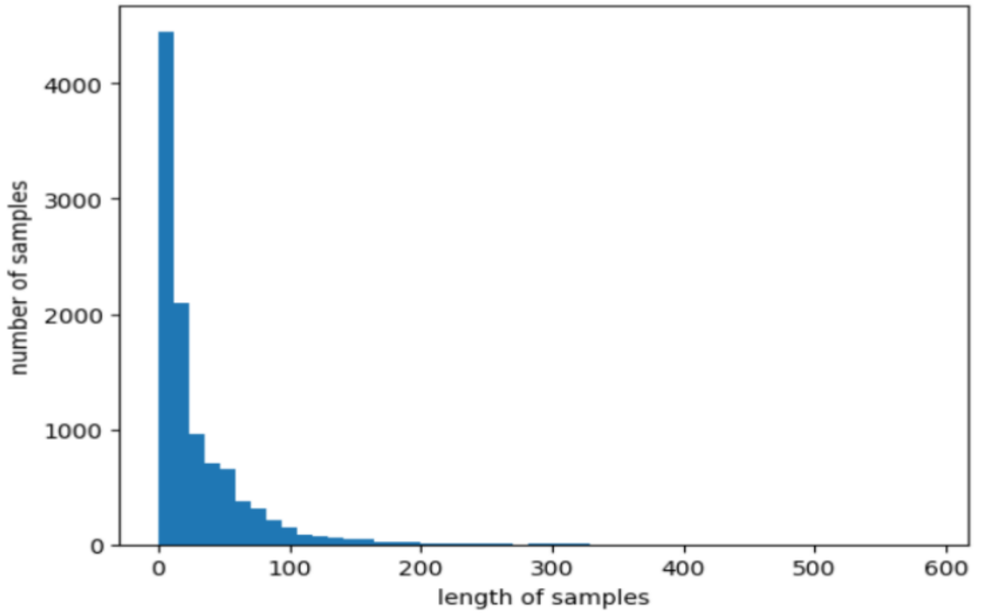
06

모델로 예측된 라벨링 결과 - 버거킹

긍정 리뷰의 평균 길이 : 25.378419452887538
부정 리뷰의 평균 길이 : 43.278137651821865



리뷰의 최대 길이 : 588
리뷰의 평균 길이 : 29.643546208759407



전체 리뷰의 최대 / 평균 길이

139/139 [==============================] - 17s 119ms/step - loss: 0.1629 - acc: 0.9327

테스트 정확도: 0.9327

모델의 정확도

Thank you