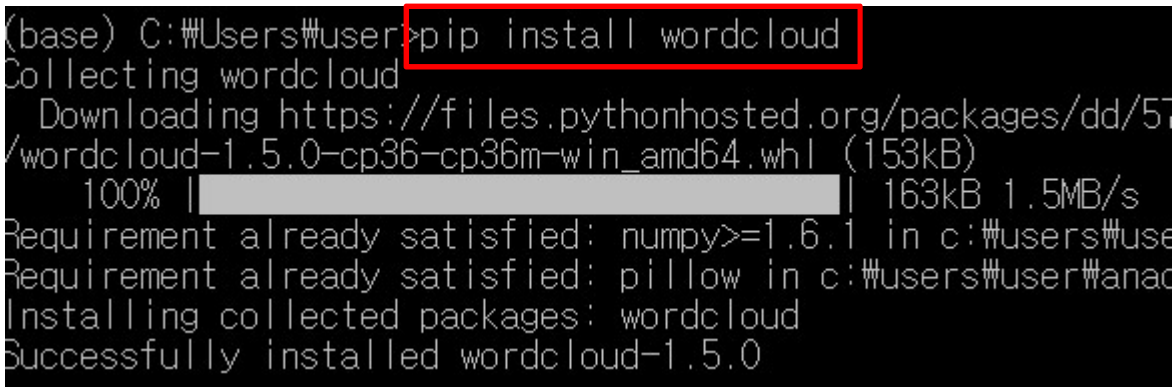
문장 분류

Prepared by Prof. Youn-Sik Hong

복제/수정/배포 금지, All rights reserved by Youn-Sik Hong, 2020

wordcloud 설치 : anaconda prompt 에서



목차

단계 1: 문제 정의 - review-data-analysis. ipynb

한글 텍스트 분류

- Konlpy 사용

Dataset

- Naver sentiment movie corpus v1.0 https://github.com/e9t/nsmc
- https://movie.naver.com/movie/point/af/list.nhn 사이트에서 scraping

Data 설명

- 영화 리뷰는 3개 columns (**id, document, label**)으로 구성
  - **id**: 리뷰 아이디
  - **label**: 평가(긍정-1, 부정-0) → 긍정, 부정 비율은 50%씩 같음
    – 긍정: ratings 9-10에 해당, 부정: ratings 1-4에 해당 ➔ ratings 5-8은 제외
  - **document**: 리뷰 본문 – 최대 140개 문자

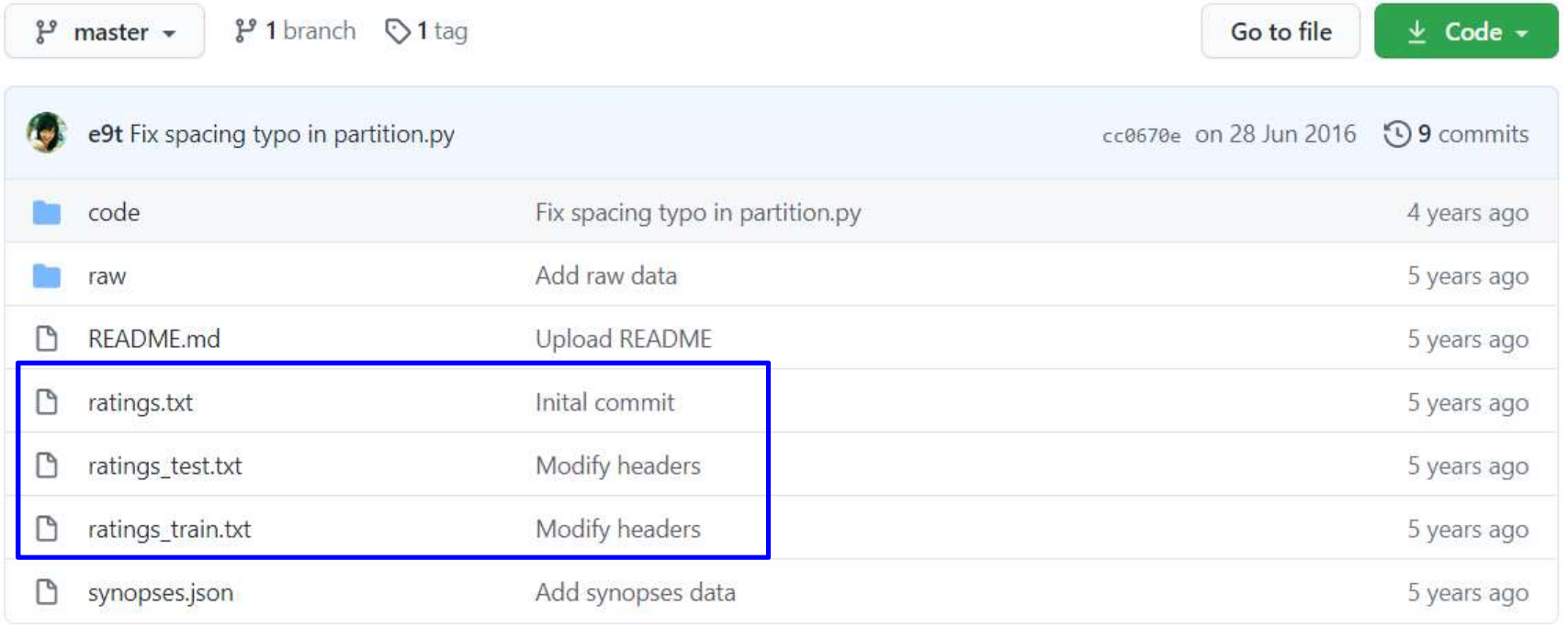
파일

- **ratings.txt** – whole dataset (200K)
- **ratings_test.txt (50K), ratings_train.txt (150K)**

목차

단계 2: 파일 다운로드 – 텍스트 파일(3개)

GitHub 에서 3개 파일 다운로드



단계 2: 파일을 폴더(data_in)에 저장

현재 작업 중인 폴더에서 하위 폴더 **data_in** 생성

단계 2 : 파일 사이즈 확인 및 DataFrame 변환 (1/4)

```
DATA_IN_PATH = './data_in/'
print('파일 크기: ')
for file in os.listdir(DATA_IN_PATH):
    if 'txt' in file:
        fsize = round(os.path.getsize(DATA_IN_PATH + file)/1000000, 2)
        file_unit = str(fsize) + 'MB'
        print(file.ljust(30)+ file_unit)
```

```
파일 크기:
ratings.txt                    19.52MB
ratings_test.txt               4.89MB
ratings_train.txt              14.63MB
```

```
train_fname = DATA_IN_PATH + 'ratings_train.txt'
train_data = pd.read_csv(train_fname, header=0, delimiter='\t', quoting=csv.QUOTE_NONE)
# quoting 옵션은 csv파일을 dataframe으로 바꿀 때 따옴표(")는 무시하겠다는 뜻.
# ratings.txt 파일에는 따옴표는 없음. naver에서 말뭉치로 바꾸면서 모두 없앤 것으로 보임.
```

단계 2 : raw data, 속성 및 실측값 확인

```
train_data.head()
```

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...	1

긍정-1, 부정-0

```
#train_data.shape
#train_data.info() # document 중에 null 값이 5개 있음.
#train_data["document"][:5]
train_data[train_data['document'].isnull()]
```

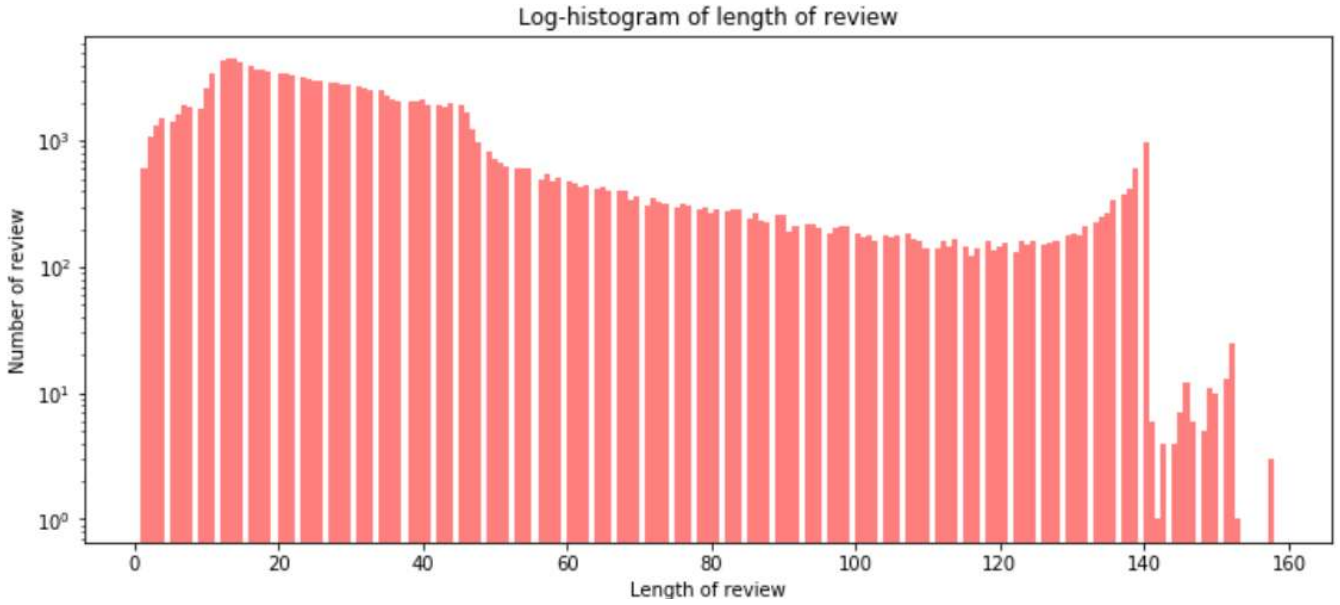
	id	document	label
25857	2172111	NaN	1
55737	6369843	NaN	1
110014	1034280	NaN	0
126782	5942978	NaN	0
140721	1034283	NaN	0

영화 감상 글은 없이
평점만 올림

단계 2 : 영화 리뷰 길이 분석 – histogram

```
f = lambda x: len(x)
train_length = train_data['document'].astype(str).apply(f)
train_length.head()
#train_length.sort_values(ascending=False)
```

```
0    19
1    33
2    17
3    29
4    61
Name: document, dtype: int64
```

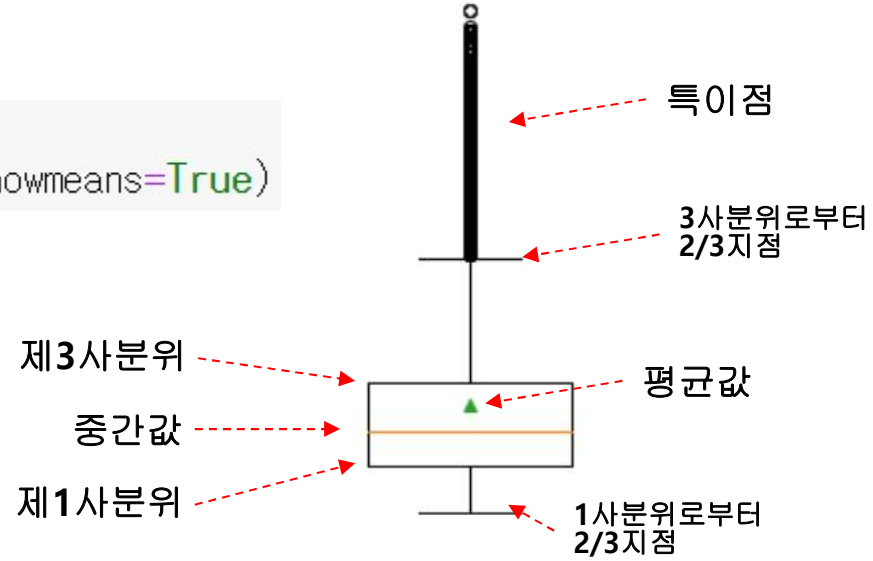


단계 2 : 영화 리뷰 길이 분석 - boxplot

```
print('리뷰 길이 최대값: %d' % (np.max(train_length)))
print('리뷰 길이 최소값: %d' % (np.min(train_length)))
print('리뷰 길이 평균값: %.2f' % (np.mean(train_length)))
print('리뷰 길이 표준편차: %.2f' % (np.std(train_length)))
print('리뷰 길이 중간값: %.1f' % (np.median(train_length)))
print('리뷰 길이 제1사분위: %.1f' % (np.percentile(train_length, 25)))
print('리뷰 길이 제3사분위: %.1f' % (np.percentile(train_length, 75)))
```

```
리뷰 길이 최대값: 158
리뷰 길이 최소값: 1
리뷰 길이 평균값: 35.24
리뷰 길이 표준편차: 29.58
리뷰 길이 중간값: 27.0
리뷰 길이 제1사분위: 16.0
리뷰 길이 제3사분위: 42.0
```

```
plt.figure(figsize=(12,5))
plt.boxplot(train_length, labels=['counts'], showmeans=True)
```



단계 2 : word cloud 출력

```
train_review = [review for review in train_data['document'] if type(review) is str]
#train_review[:2]
```

```
hangul_ttf = DATA_IN_PATH + 'NanumGothic.ttf'
wordcloud = WordCloud(font_path=hangul_ttf).generate(' '.join(train_review))
```


```
neg_review = train_data['label'].value_counts()[0]
pos_review = train_data['label'].value_counts()[1]
print('긍정 리뷰 개수:{}'.format(pos_review))
print('부정 리뷰 개수:{}'.format(neg_review))
```

긍정 리뷰 개수:74827
부정 리뷰 개수:75173

음절에 대한 데이터 분석 과정과 동일하게 단어에 대한 데이터 분석 수행

목차

단계 3 : review-preprocessing. ipynb

```
DATA_IN_PATH = './data_in/'
train_file = DATA_IN_PATH + 'ratings_train.txt'
train_data = pd.read_csv(train_file, header=0, delimiter='\t', quoting=csv.QUOTE_NONE)
```

```
train_data['document'][:5]
```

```
0                                  아 더빙.. 진짜 짜증나네요 목소리
1                                  흠...포스터보고 초딩영화줄...오버연기조차 가볍지 않구나
2                                  너무재밋었다그래서보는것을추천한다
3                                  교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정
4                                  사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...
```

```
# 정규표현(re)을 사용해 아래에 해당하지 않는 기호('^')는 모두 제거
# 한글음절 11,174자 ('가'-'힣'), 자음('ㄱ'-'ㅎ'), 모음('ㅏ'-'ㅣ'), whitespace char(\s)
print(train_data['document'][0])
review_text = re.sub('[^가-힣ㄱ-ㅎㅏ-ㅣ\\s]', '', train_data['document'][0])
print(review_text)
```

```
아 더빙.. 진짜 짜증나네요 목소리
아 더빙 진짜 짜증나네요 목소리
```

공백 문자는
제거하지 않음

단계 3 : 한글 형태소 분석 및 불용어 처리

```
okt = Okt()
review_text = okt.morphs(review_text, stem=True)
print(review_text)
```

어간(stem)을 추출

['아', '더빙', '진짜', '짜증나다', '목소리']

```
with open('./kr_stopwords.txt', encoding='utf8') as f:
    stopwords = f.readlines()
stopwords = [x.strip() for x in stopwords]
print(stopwords[:10])
```

한국어 불용어 사전

['아', '휴', '아이구', '아이쿠', '아이고', '어', '나', '우리', '저희', '따라']

```
print(review_text)
revised_text = [w for w in review_text if len(w) > 1]
clean_review = [w for w in revised_text if not w in stopwords]
print(clean_review)
```

음절 하나인 단어는 제거

['더빙', '진짜', '짜증나다', '목소리']

단계 3 : 함수 정의 (한글 음절 + 불용어 처리) (1/2)

```
def preprocessing(review, remove_stopwords, stop_words):
    review_text = re.sub('[^가-힣ㄱ-ㅎㅏ-ㅣㅠㅜs]', '', review)
    word_review = okt.morphs(review_text, stem=True)

    if remove_stopwords:
        revised_text = [w for w in word_review if len(w) > 1]
        word_review = [token for token in revised_text
                       if not token in stop_words]

    return word_review
```

함수 정의

```
clean_train_review = []

i = 0
max = len(train_data['document'])
for review in train_data['document']:
    if (i % 1500 == 0):
        print('진행률= %d 퍼센트' % ((i/max * 100)+1))
    if type(review) == str:
        clean_train_review.append(preprocessing(review, True, stopwords))
    else:
        clean_train_review.append([])
    i += 1
```

진행률 표시 – too long!
5분 이상 걸림

단계 3 : 테스트 파일에 전처리 함수 적용

```
test_file = DATA_IN_PATH + 'ratings_test.txt'
test_data = pd.read_csv(test_file, header=0, delimiter='₩t', quoting=csv.QUOTE_NONE)
clean_test_review = []

i = 0
max = len(test_data['document'])
for review in test_data['document']:
    if i % 500 == 0:
        print('진행률= %d 퍼센트' % ((i/max * 100)+1))

    if type(review) == str:
        clean_test_review.append(preprocessing(review,True, stopwords))
    else:
        clean_test_review.append([])
    i += 1
```

진행률 표시 – too long!
3분 이상 걸림

'NaN'(Not a Number)'
값이 없다는 뜻.

	id	document	label
25857	2172111	NaN	1
55737	6369843	NaN	1
110014	1034280	NaN	0
126782	5942978	NaN	0
140721	1034283	NaN	0

단계 3 : index 벡터 변환

- tensorflow를 사용해 기계학습모델에 적용하기 위해서는 텍스트 데이터인 단어를 수치 데이터로 변환
  - text_to_sequences : 전 처리가 끝난 train_review와 test_review 벡터를 index 벡터로 변환.
  - 모든 index는 word_vocab에 저장.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_train_review)
train_sequence = tokenizer.texts_to_sequences(clean_train_review)
test_sequence = tokenizer.texts_to_sequences(clean_test_review)

word_vocab = tokenizer.word_index
```

```
print(clean_train_review[0])
print(train_sequence[0])
print(len(word_vocab))
```

```
['더빙', '진짜', '짜증나다', '목소리']
[463, 20, 265, 664]
43756
```

```
{'영화': 1, '보다': 2, '하다': 3, '에': 4, '을': 5, '도': 6, '를': 7, '없다': 8, '이다': 9, '있다': 10, '좋다': 11, '너무': 12, '다': 13, '정말': 14, '재밌다': 15, '되다': 16, '적': 17, '만': 18, '같다': 19, '진짜': 20, '으로': 21, '로': 22, '아니다': 23, '않다': 24, '점': 25, '에서': 26, '만들다': 27, '과': 28, '나오다': 29, '연기': 30, '평점': 31,
```

단계 3 : 일정 길이 벡터로 변환

```
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
```

각 벡터는 서로 길이가 다름. 이 길이를 하나로 통일해야 기계학습모델에 적용할 수 있음. 최대 길이(MAX_SEQUENCE_LENGTH=8)를 정하고, 이 길이보다 긴 벡터는 자르며, 이 길이보다 짧은 벡터는 빈 자리에 0을 추가(padding)한다.

```
MAX_SEQUNCE_LENGTH = 8

train_inputs = pad_sequences(train_sequence, maxlen=MAX_SEQUNCE_LENGTH, padding='post')
train_labels = np.array(train_data['label'])
```

```
print(train_inputs[:5])
```

```
[[  463    20   265   664     0     0     0     0]
 [  604     1   219  1459    30   969   680    24]
 [  393  2456 25028  2323  5680     2   226    13]
 [ 6504   110  8137   224    61     8    31  3621]
 [ 1115   244     3 14259 19430  1083   259   244]]
```

```
test_inputs = pad_sequences(test_sequence, maxlen=MAX_SEQUNCE_LENGTH, padding='post')
test_labels = np.array(test_data['label'])
```

단계 3 : 변환 결과를 저장

```
DATA_IN_PATH = './data_in/'
TRAIN_INPUT_DATA = 'nsmc_train_input.npy'
TRAIN_LABEL_DATA = 'nsmc_train_label.npy'
TEST_INPUT_DATA = 'nsmc_test_input.npy'
TEST_LABEL_DATA = 'nsmc_test_label.npy'
DATA_CONFIGS = 'data_configs.json'
```

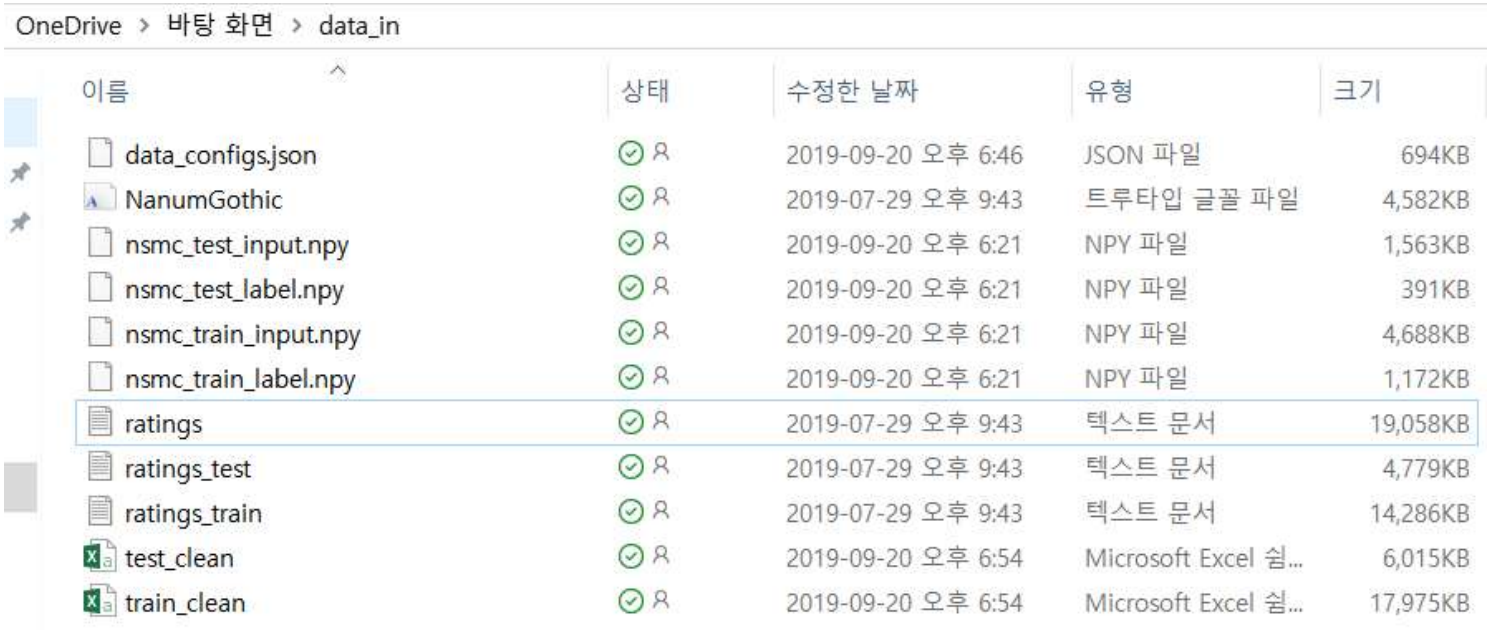
```
data_configs = {}

data_configs['vocab'] = word_vocab
data_configs['vocab_size'] = len(word_vocab)+1
```

```
np.save(open(DATA_IN_PATH + TRAIN_INPUT_DATA, 'wb'), train_inputs)
np.save(open(DATA_IN_PATH + TRAIN_LABEL_DATA, 'wb'), train_labels)
np.save(open(DATA_IN_PATH + TEST_INPUT_DATA, 'wb'), test_inputs)
np.save(open(DATA_IN_PATH + TEST_LABEL_DATA, 'wb'), test_labels)
```

```
json.dump(data_configs, open(DATA_IN_PATH + DATA_CONFIGS, 'w'), ensure_ascii=False)
```

단계 3 : 최종 결과

```
import pandas as pd
TRAIN_CLEAN_DATA = 'train_clean.csv'
TEST_CLEAN_DATA = 'test_clean.csv'
clean_train_df = pd.DataFrame({'review':clean_train_review,
                               'sentiment':train_data['label']})
clean_test_df = pd.DataFrame({'review':clean_test_review,
                              'sentiment':test_data['label']})
clean_train_df.to_csv(DATA_IN_PATH + TRAIN_CLEAN_DATA, index=False)
clean_test_df.to_csv(DATA_IN_PATH + TEST_CLEAN_DATA, index=False)
```



목차

한글 텍스트 분류 : 영화 리뷰 감정 분석

단계 4 : 모델링 – review-modeling-regression. ipynb (1/2)

```
DATA_IN_PATH = './data_in/'
TRAIN_CLEAN_DATA = 'train_clean.csv'
train_data = pd.read_csv(DATA_IN_PATH + TRAIN_CLEAN_DATA)
```

```
reviews = list(train_data['review'])
sentiments = list(train_data['sentiment'])
```

Tfidf 방식으로 벡터 변환 → Logistic Regression 적용

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
vectorizer = TfidfVectorizer(min_df = 0.0, analyzer="char",
            sublinear_tf=True, ngram_range=(1,3), max_features=5000)

X = vectorizer.fit_transform(reviews)
y = np.array(sentiments)
```

단계 4 : 모델링 (2/2)

```
RANDOM_SEED = 42
TEST_SPLIT = 0.2
X_train, X_eval, y_train, y_eval = train_test_split(X, y,
                test_size=TEST_SPLIT, random_state=RANDOM_SEED)
```

```
lgs = LogisticRegression(class_weight='balanced')
lgs.fit(X_train, y_train)
```

```
predicted = lgs.predict(X_eval)
```

```
print(predicted[277])
print(y_eval[277])
#print(predicted[1000])
#print(y_eval[1000])
```

```
0
0
```

```
print("Accuracy: %f" % lgs.score(X_eval, y_eval))
```

```
Accuracy: 0.822267
```