네이버 영화리뷰 감정분석 - 딥러닝

part1. 데이터 전처리

[1] 0초
```
import warnings
# 경고메세지 끄기
warnings.filterwarnings(action='ignore')
```

[2] 26초
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3] 11초
```
import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt

# !pip install konlpy

df = pd.read_csv('/content/drive/MyDrive/강의자료/핀인사이트_강의자료 /2023 강의자료 업데이트/자연어처리/실습/data/review.csv',
                 sep = '\t')
df.head()
```

```
Collecting konlpy
  Downloading konlpy-0.6.0-py2.py3-none-any.whl (19.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.4/19.4 MB 60.9 MB/s eta 0:00:00
Collecting JPype1>=0.7.0 (from konlpy)
  Downloading JPype1-1.4.1-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (465 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 465.3/465.3 kB 46.0 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (4.9.
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.23.
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from JPype1>=0.7.0->
Installing collected packages: JPype1, konlpy
Successfully installed JPype1-1.4.1 konlpy-0.6.0
```

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

[5]
```python
import re

# '한글'을 제외한 다른 문자 모두 제거
remove_except_ko = re.compile(r"[^가-힣ㄱ-ㅎㅏ-ㅣ\\s]")
def preprocess_remove(text):
    text = re.sub(remove_except_ko,' ',text).strip()
    return text

df['document'] = df['document'].map(lambda x : preprocess_remove(x))
```

[7]
```python
from konlpy.tag import Okt

# 토큰화 / 불용어처리
stop_pos = ['Josa','Eomi','Punctuation','Foreign','Number','Unknown','KoreanParticle']
stop_word = ['영화']

tokenizer= Okt()
df['morphs'] = None
for i,row in df.iterrows():
  tokens = tokenizer.pos(row['document'])
  token_ls = []
  for token in tokens:
    if len(token[0]) > 1:
      if token[0] not in stop_word:
        if token[1] not in stop_pos:
          token_ls.append(token[0])
  # print(token_ls)
  df['morphs'][i] = ' '.join(token_ls)
```

```python
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer

X_train, X_test, y_train, y_test = train_test_split(df['morphs'], df['label'], random_state = 0)

tokenizer = Tokenizer() # 선언
# make train word set
tokenizer.fit_on_texts(X_train)
```

[9]
```python
print(len(tokenizer.word_index))
tokenizer.word_index
```

```
16111
{'정말': 1,
 '너무': 2,
 '진짜': 3,
 '연기': 4,
 '평점': 5,
 '최고': 6,
 '스토리': 7,
 '사람': 8,
 '보고': 9,
```

[9]
```
 '보고': 9,
 '이런': 10,
 '드라마': 11,
 '하는': 12,
 '생각': 13,
 '감동': 14,
 '시간': 15,
 '그냥': 16,
 '감독': 17,
 '배우': 18,
 '내용': 19,
 '재미': 20,
 '쓰레기': 21,
 '없는': 22,
 '작품': 23,
 '사랑': 24,
 '하나': 25,
 '주인공': 26,
 '없다': 27,
 '이건': 28,
 '있는': 29,
 '같은': 30,
 '정도': 31,
 '다시': 32,
 '마지막': 33,
 '이렇게': 34,
 '봤는데': 35,
 '액션': 36,
 '완전': 37,
 '처음': 38,
 '연출': 39,
 '입니다': 40,
 '장면': 41,
 '최악': 42,
 '지금': 43,
 '느낌': 44,
 '역시': 45,
 '명작': 46,
 '없고': 47,
 '보는': 48,
 '이야기': 49,
 '봐도': 50,
 '그리고': 51,
 '별로': 52,
 '좋은': 53,
 '많이': 54,
 '보면': 55,
 '같다': 56,
 '이해': 57,
```

0초
```
1 X_train[:2]
```

```
2967                                    존경 하시는
700     음악 취향 저격 비긴 어게인 대중 흥행 하긴 힘들 하지만 여주 너무 매력
Name: morphs, dtype: object
```

[11]
```
#단어를 숫자배열로 변환
X_train_arrary_list = tokenizer.texts_to_sequences(X_train)
X_test_arrary_list = tokenizer.texts_to_sequences(X_test)

# 레이블링 데이터 행렬변환
Y_train = np.array(y_train) # 형 변환 : x 와 타입을 동일하게 맞춰주기 위해서 진행
Y_test = np.array(y_test)
print('훈련데이터 X: ',len(X_train_arrary_list))
print('훈련데이터 Y: ',len(Y_train))
```

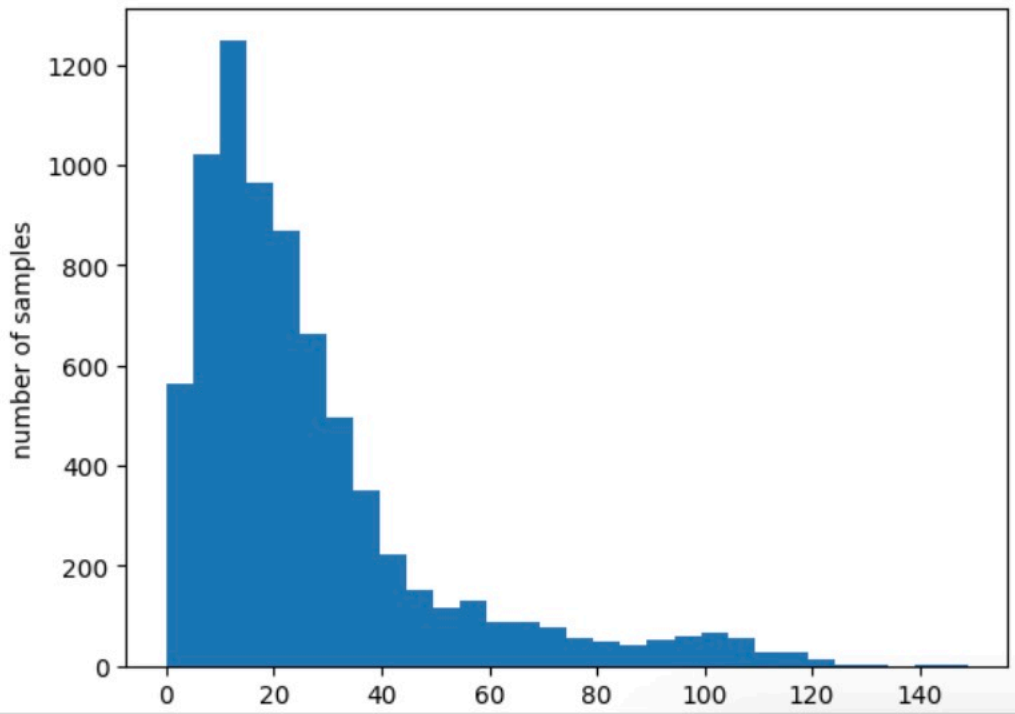
```
훈련데이터 X:  7500
훈련데이터 Y:  7500
```

[12]
```
# 확인
print(X_train[:2])
print(X_train_arrary_list[:2])
```

```
2967                                          존경 하시는
700     음악 취향 저격 비긴 어게인 대중 흥행 하긴 힘들 하지만 여주 너무 매력
Name: morphs, dtype: object
[[616, 1616], [118, 464, 5826, 3548, 3549, 1985, 295, 1986, 1987, 77, 351, 2, 74]]
```

```
print('리뷰의 최대 길이 :',max(len(l) for l in X_train))
print('리뷰의 평균 길이 :',sum(map(len, X_train))/len(X_train))
plt.hist([len(s) for s in X_train], bins=30)
plt.xlabel('length of samples')
plt.ylabel('number of samples')
plt.show()
```

```
리뷰의 최대 길이 : 149
리뷰의 평균 길이 : 26.0936
```



```
[14]  1 # 편하게 사용하기  위해서 재 할당
      2 X_train = X_train_arrary_list
      3 X_test = X_test_arrary_list
```

```
[15]  1 from tensorflow.keras.preprocessing.sequence import pad_sequences
      2
      3 # 데이터 길이 35로 패딩
      4 max_len = 35
      5 X_train = pad_sequences(X_train, maxlen = max_len)
      6 X_test = pad_sequences(X_test, maxlen = max_len)
```

```
[16]  1 X_train[0]
```

```
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,
        616, 1616], dtype=int32)
```

part2. 모델링

```
[17]  1 # 구글 드라이브 경로 설정
      2 path = '/content/'
```

```
[18]  1 from tensorflow.keras.layers import Embedding, Dense, LSTM
      2 from tensorflow.keras.models import Sequential
      3 from tensorflow.keras.models import load_model
      4 from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
      5
      6 vocab_size = len(X_train) # 학습데이터 길이 = vocab size
      7 embedding_dim = 100
      8
      9 model = Sequential() # 모델 선언
     10 model.add(Embedding(vocab_size, embedding_dim))
     11 model.add(LSTM(128))
     12 model.add(Dense(128, activation='relu'))
     13 model.add(Dense(64, activation='relu'))
     14 model.add(Dense(8, activation='relu'))
     15 model.add(Dense(1, activation='sigmoid'))
```

참고

- monitor : 학습 조기종료를 위해 관찰하는 항목 (default : val_loss)
- patience : 개선이 안된다고 바로 종료시키지 않고, 개선을 위해 몇번의 에포크를 기다릴지 설정 (default = 0)
- verbose : 언제 traing을 멈추었는지에 대한것을 화면에 출력할지 여부(1 = 출력)
- mode : 관찰항목에 대해 개선이 없다고 판단하기 위한 기준을 설정. monitor에서 설정한 항목이 val_loss 이면 값이 감소되지 않을 때 종료하여야 하므로 min 을 설정하고, val_accuracy 의 경우에는 max를 설정해야함. (default = auto)
  - auto : monitor에 설정된 이름에 따라 자동으로 지정
  - min : 관찰값이 감소하는 것을 멈출 때, 학습을 종료
  - max: 관찰값이 증가하는 것을 멈출 때, 학습을 종료

[19] 0초

```
# 검증 데이터 손실(val_loss)이 증가하면, 과적합이 될수 있기 때문에 검증 데이터 손실이 4회 증가하면 학습을 조기 종료(Early Stopping).
# ModelCheckpoint를 사용하여 검증 데이터의 정확도(val_acc)가 이전보다 좋아질 경우에만 모델을 저장
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('./data/naver_movie_model_lstm', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
```

1분

```
# generate model
# loss = 'sparse_categorical_crossentropy' # 분류모델사용
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, Y_train, epochs=15, callbacks=[es, mc], batch_size=16, validation_split=0.2)
```

```
Epoch 1/15
375/375 [==============================] - ETA: 0s - loss: 0.6934 - acc: 0.4918
Epoch 1: val_acc improved from -inf to 0.61867, saving model to ./data/naver_movie_model_lstm
375/375 [==============================] - 37s 76ms/step - loss: 0.6934 - acc: 0.4918 - val_loss: 0.6913 - val_acc: 0.6187
Epoch 2/15
374/375 [============================>.] - ETA: 0s - loss: 0.5620 - acc: 0.7029
Epoch 2: val_acc improved from 0.61867 to 0.74733, saving model to ./data/naver_movie_model_lstm
375/375 [==============================] - 13s 34ms/step - loss: 0.5621 - acc: 0.7028 - val_loss: 0.4869 - val_acc: 0.7473
Epoch 3/15
374/375 [============================>.] - ETA: 0s - loss: 0.3931 - acc: 0.8245
Epoch 3: val_acc improved from 0.74733 to 0.76733, saving model to ./data/naver_movie_model_lstm
375/375 [==============================] - 8s 22ms/step - loss: 0.3931 - acc: 0.8248 - val_loss: 0.4676 - val_acc: 0.7673
Epoch 4/15
370/375 [============================>.] - ETA: 0s - loss: 0.3207 - acc: 0.8659
Epoch 4: val_acc improved from 0.76733 to 0.77667, saving model to ./data/naver_movie_model_lstm
375/375 [==============================] - 10s 27ms/step - loss: 0.3202 - acc: 0.8660 - val_loss: 0.4996 - val_acc: 0.7767
Epoch 5/15
369/375 [============================>.] - ETA: 0s - loss: 0.2706 - acc: 0.8848
Epoch 5: val_acc improved from 0.77667 to 0.77733, saving model to ./data/naver_movie_model_lstm
375/375 [==============================] - 8s 22ms/step - loss: 0.2699 - acc: 0.8853 - val_loss: 0.5072 - val_acc: 0.7773
```

```
Epoch 6/15
375/375 [==============================] - ETA: 0s - loss: 0.2254 - acc: 0.9080
Epoch 6: val_acc did not improve from 0.77733
375/375 [==============================] - 3s 8ms/step - loss: 0.2254 - acc: 0.9080 - val_loss: 0.5534 - val_acc: 0.7587
Epoch 7/15
371/375 [============================>.] - ETA: 0s - loss: 0.1873 - acc: 0.9223
Epoch 7: val_acc did not improve from 0.77733
375/375 [==============================] - 3s 8ms/step - loss: 0.1867 - acc: 0.9228 - val_loss: 0.6036 - val_acc: 0.7587
Epoch 7: early stopping
```

[21]
```
print("\n 테스트 정확도: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

```
79/79 [==============================] - 0s 4ms/step - loss: 0.6052 - acc: 0.7736

 테스트 정확도: 0.7736
```

[22]
```
y_predict = model.predict(X_test)
y_predict
```

```
79/79 [==============================] - 1s 3ms/step
array([[0.86680305],
       [0.9985902 ],
       [0.9991875 ],
       ...,
       [0.35415748],
       [0.00227166],
       [0.98270744]], dtype=float32)
```

[23]
```
y_predict2 = []
for x in y_predict:
  if x >=0.5:
    y_predict2.append(1)
  else:
    y_predict2.append(0)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict2, target_names=['부정( 0 )', '긍정( 1 )']))
```

```
              precision    recall  f1-score   support

      부정( 0 )       0.74      0.85      0.79      1248
      긍정( 1 )       0.83      0.69      0.75      1252

    accuracy                           0.77      2500
   macro avg       0.78      0.77      0.77      2500
weighted avg       0.78      0.77      0.77      2500
```

[ ]
```
1
```

[ ]
```
1
```