TF-IDF

scikit-learn을 활용한 TF-IDF구현

```
1 from sklearn.feature_extraction.text import CountVectorizer
```

```
1 corpus = [
2           'you know I want your love',
3           'I like you',
4           'what should I do'
5 ]
```

```
1 vector = CountVectorizer()
```

```
1 print(vector.fit_transform(corpus).toarray())
```

```
[[0 1 0 1 0 1 0 1 1]
 [0 0 1 0 0 0 0 1 0]
 [1 0 0 0 1 0 1 0 0]]
```

```
1 print(vector.vocabulary_)
```

```
{'you': 7, 'know': 1, 'want': 5, 'your': 8, 'love': 3, 'like': 2, 'what': 6, '
```

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
```

```
1 corpus = [
2           'you know I want your love',
3           'I like you',
4           'what should I do'
5 ]
```

```
1 tfidfv = TfidfVectorizer().fit(corpus)
```

```
1 print(tfidfv.transform(corpus).toarray())
```

```
[[0.         0.46735098 0.         0.46735098 0.         0.46735098
  0.         0.35543247 0.46735098]
 [0.         0.         0.79596054 0.         0.         0.
  0.         0.60534851 0.        ]
 [0.57735027 0.         0.         0.         0.57735027 0.
  0.57735027 0.         0.        ]]
```

```
1 print(tfidfv.vocabulary_)
```

```
{'you': 7, 'know': 1, 'want': 5, 'your': 8, 'love': 3, 'like': 2, 'what': 6, '
```

로이터 데이터로 TF-IDF 학습하기

```
from tensorflow.keras.datasets import reuters
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

```
(x_train, y_train), (x_test, y_test) = reuters.load_data(num_words=10000, test_s
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datas
2113536/2110848 [==============================] - 0s 0us/step
2121728/2110848 [==============================] - 0s 0us/step
```

```
print('훈련 샘플의 수 : {}'.format(len(x_train)))
```

```
훈련 샘플의 수 : 8982
```

```
print('테스트 샘플의 수 : {}'.format(len(x_test)))
```

```
테스트 샘플의 수 : 2246
```

```
print(x_train[0])
print(x_test[0])
```

```
[1, 2, 2, 8, 43, 10, 447, 5, 25, 207, 270, 5, 3095, 111, 16, 369, 186, 90, 67,
[1, 4, 1378, 2025, 9, 697, 4622, 111, 8, 25, 109, 29, 3650, 11, 150, 244, 364,
```

```
print(y_train[0]) # 훈련 기사의 레이블
```

```
3
```

```
num_classes = max(y_train)+1
print('클래스의 수 : {}'.format(num_classes))
```

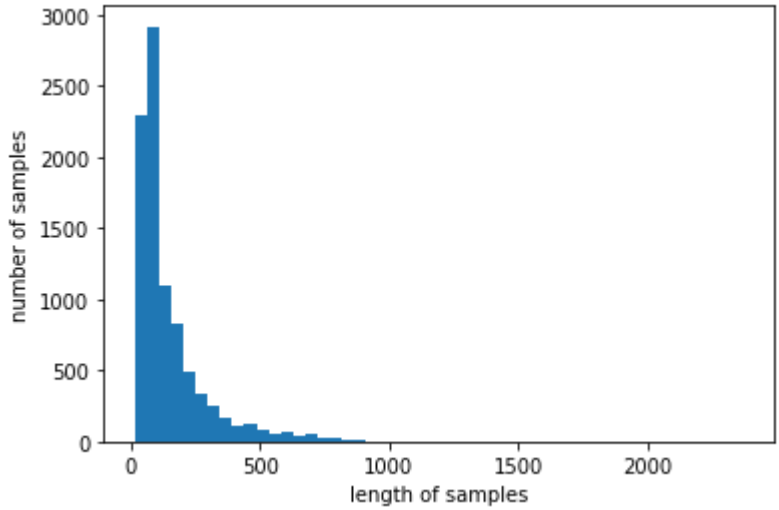
```
클래스의 수 : 46
```

```
print('훈련용 뉴스의 최대 길이 : {}'.format(max(len(l) for l in x_train)))
print('훈련용 뉴스의 평균 길이 : {}'.format(sum(map(len, x_train))/len(x_train)))
```

```
훈련용 뉴스의 최대 길이 : 2376
훈련용 뉴스의 평균 길이 : 145.5398574927633
```

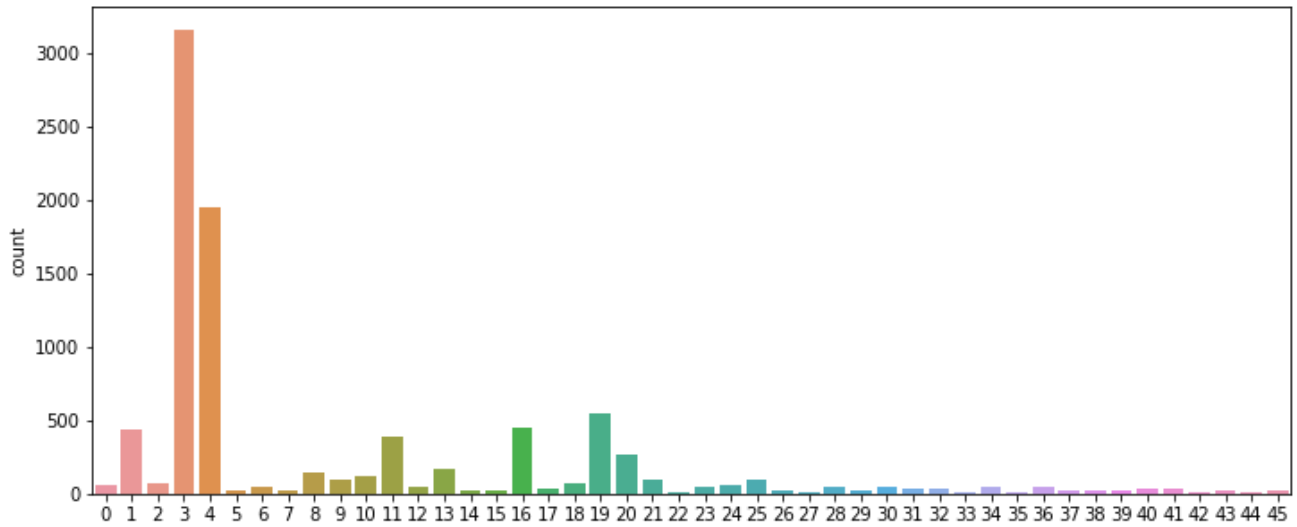
```
plt.hist([len(s) for s in x_train], bins= 50)
plt.xlabel('length of samples')
plt.ylabel('number of samples')
```

```
4 plt.show()
```




```
1 fig, axe = plt.subplots(ncols =1)
2 fig.set_size_inches(12, 5)
3 sns.countplot(y_train)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnir
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fe8a2d8f310>
```




```
1 unique_elements, counts_elements = np.unique(y_train, return_counts=True)
2 print("각 클래스 빈도수 : ")
3 print(np.asarray((unique_elements, counts_elements)))
```

```
각 클래스 빈도수 :
[[   0    1    2    3    4    5    6    7    8    9   10   11   12   13
    14   15   16   17   18   19   20   21   22   23   24   25   26   27
    28   29   30   31   32   33   34   35   36   37   38   39   40   41
    42   43   44   45]
 [  55  432   74 3159 1949   17   48   16  139  101  124  390   49  172
    26   20  444   39   66  549  269  100   15   41   62   92   24   15
    48   19   45   39   32   11   50   10   49   19   19   24   36   30
    13   21   12   18]]
```

```
word_to_index = reuters.get_word_index()
print(word_to_index)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datas
557056/550378 [==============================] - 0s 0us/step
565248/550378 [==============================] - 0s 0us/step
{'mdbl': 10996, 'fawc': 16260, 'degussa': 12089, 'woods': 8803, 'hanging': 137
```

```
index_to_word = {index +3 : word for word, index in word_to_index.items()}
```

```
word_to_index['the']
```

```
1
```

```
word_to_index['it']
```

```
13
```

```
print(index_to_word[4]) # 빈도수 상위 1번 단어
print(index_to_word[16]) # 빈도수 상위 13번 단어
```

```
the
it
```

```
# 0 <pad>
# 1 <sos>
# 2 <unk>
for index, token in enumerate(("<pad>", "<sos>", "<unk>")):
    index_to_word[index] = token
```

```
print(' '.join([index_to_word[index] for index in x_train[0]]))
```

```
<sos> <unk> <unk> said as a result of its december acquisition of space co it
```

```
# 전체 훈련 데이터에 대해서 decoded
decoded = []
for i in range(len(x_train)):
    t = ' '.join([index_to_word[index] for index in x_train[i]])
    decoded.append(t)

x_train = decoded
```

```
# 전체 테스트 데이터에 대해서 decoded
decoded = []
for i in range(len(x_test)):
    t = ' '.join([index_to_word[index] for index in x_test[i]])
    decoded.append(t)

x_test = decoded
```

```
1 x_train[:5]
```

```
['<sos> <unk> <unk> said as a result of its december acquisition of space co i
 '<sos> generale de banque sa lt <unk> br and lt heller overseas corp of chica
 '<sos> shr 3 28 dlrs vs 22 cts shr diluted 2 99 dlrs vs 22 cts net 46 0 mln v
 "<sos> the farmers home administration the u s agriculture department's farm
 '<sos> seton co said its board has received a proposal from chairman and chie
```

```
1 x_test[:5]
```

```
['<sos> the great atlantic and pacific tea co said its three year 345 mln dlr
 "<sos> philippine sugar production in the 1987 88 crop year ending august has
 "<sos> the agriculture department's widening of louisiana gulf differentials
 '<sos> <unk> <unk> oil and gas partnership said it completed the sale of inte
 '<sos> strong south <unk> winds were keeping many vessels trapped in the ice
```

TF-IDF로 로이터 데이터 학습하기

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.feature_extraction.text import CountVectorizer
3 from sklearn.feature_extraction.text import TfidfTransformer
4 from sklearn.naive_bayes import MultinomialNB
5 from sklearn import metrics
```

```
1 # 단어의 수를 카운트하는 사이킷런의 카운트벡터라이저.
2 count_vect = CountVectorizer()
3 # fit_transform : 학습 할 때와 동일한 기반 설정으로 동일하게 테스트 데이터를 변환해야 하는 것
4 x_train_counts = count_vect.fit_transform(x_train)
5
6 # 카운트벡터라이저의 결과로부터 TF-IDF 결과를 얻습니다.
7 tfidf_transformer = TfidfTransformer()
8 x_train_tfidf = tfidf_transformer.fit_transform(x_train_counts)
9
10 # 나이브 베이즈 분류기를 수행
11 # x_train은 TF-IDF의 벡터, y_train 레이블
12 clf = MultinomialNB().fit(x_train_tfidf, y_train)
```

```
1 def tfidf_vectorizer(data):
2     data_counts = count_vect.transform(data)
3     data_tfidf = tfidf_transformer.transform(data_counts)
4     return data_tfidf
```

```
1 y_pred = clf.predict(tfidf_vectorizer(x_test))
```

```
1 print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        12
```

```
           1       0.62      0.69      0.65       105
           2       0.00      0.00      0.00        20
           3       0.81      0.90      0.85       813
           4       0.51      0.96      0.67       474
           5       0.00      0.00      0.00         5
           6       0.00      0.00      0.00        14
           7       0.00      0.00      0.00         3
           8       0.00      0.00      0.00        38
           9       1.00      0.08      0.15        25
          10       0.00      0.00      0.00        30
          11       0.66      0.63      0.64        83
          12       0.00      0.00      0.00        13
          13       1.00      0.03      0.05        37
          14       0.00      0.00      0.00         2
          15       0.00      0.00      0.00         9
          16       0.69      0.56      0.61        99
          17       0.00      0.00      0.00        12
          18       0.00      0.00      0.00        20
          19       0.60      0.78      0.68       133
          20       1.00      0.04      0.08        70
          21       0.00      0.00      0.00        27
          22       0.00      0.00      0.00         7
          23       0.00      0.00      0.00        12
          24       0.00      0.00      0.00        19
          25       1.00      0.03      0.06        31
          26       0.00      0.00      0.00         8
          27       0.00      0.00      0.00         4
          28       0.00      0.00      0.00        10
          29       0.00      0.00      0.00         4
          30       0.00      0.00      0.00        12
          31       0.00      0.00      0.00        13
          32       0.00      0.00      0.00        10
          33       0.00      0.00      0.00         5
          34       0.00      0.00      0.00         7
          35       0.00      0.00      0.00         6
          36       0.00      0.00      0.00        11
          37       0.00      0.00      0.00         2
          38       0.00      0.00      0.00         3
          39       0.00      0.00      0.00         5
          40       0.00      0.00      0.00        10
          41       0.00      0.00      0.00         8
          42       0.00      0.00      0.00         3
          43       0.00      0.00      0.00         6
          44       0.00      0.00      0.00         5
          45       0.00      0.00      0.00         1

    accuracy                           0.66      2246
   macro avg       0.17      0.10      0.10      2246
weighted avg       0.59      0.66      0.58      2246

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308
```

LstM모델 구현

```
1 from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, LSTM, Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model
```

```
(x_train, y_train), (x_test, y_test) = reuters.load_data(num_words=1000, test_sp
```

```
max_len = 100
x_train = pad_sequences(x_train, maxlen= max_len)
x_test = pad_sequences(x_test, maxlen=max_len)
```

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
vocab_size = 1000
embedding_dim = 128
hidden_units = 128
num_classes = 46

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(LSTM(hidden_units))
model.add(Dense(num_classes, activation='softmax'))
```

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

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc']
```

```
history = model.fit(x_train, y_train, batch_size = 128, epochs= 30, callbacks=[e
```

```
Epoch 1/30
71/71 [==============================] - ETA: 0s - loss: 2.6258 - acc: 0.3477
Epoch 00001: val_acc improved from -inf to 0.36198, saving model to best_model
71/71 [==============================] - 21s 274ms/step - loss: 2.6258 - acc:
Epoch 2/30
71/71 [==============================] - ETA: 0s - loss: 2.1235 - acc: 0.4591
Epoch 00002: val_acc improved from 0.36198 to 0.50000, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 2.1235 - acc:
Epoch 3/30
71/71 [==============================] - ETA: 0s - loss: 1.8979 - acc: 0.5125
Epoch 00003: val_acc improved from 0.50000 to 0.53117, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 1.8979 - acc:
Epoch 4/30
71/71 [==============================] - ETA: 0s - loss: 1.7365 - acc: 0.5530
Epoch 00004: val_acc improved from 0.53117 to 0.56055, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 1.7365 - acc:
Epoch 5/30
71/71 [==============================] - ETA: 0s - loss: 1.6632 - acc: 0.5757
Epoch 00005: val_acc did not improve from 0.56055
71/71 [==============================] - 19s 269ms/step - loss: 1.6632 - acc:
```

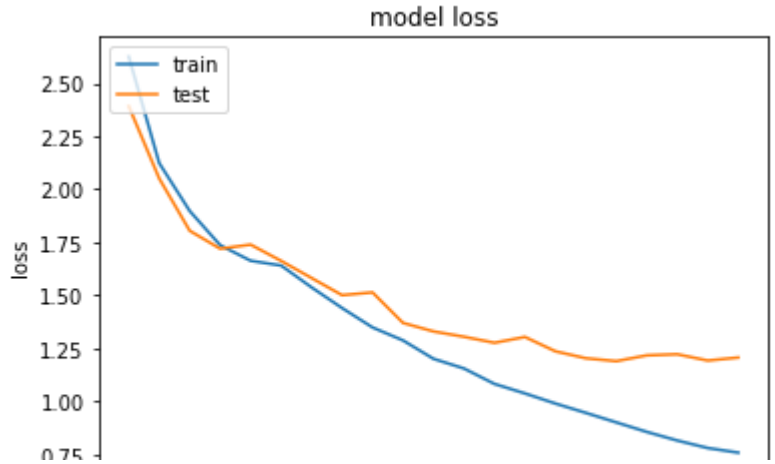
```
Epoch 6/30
71/71 [==============================] - ETA: 0s - loss: 1.6406 - acc: 0.5835
Epoch 00006: val_acc improved from 0.56055 to 0.56679, saving model to best_mo
71/71 [==============================] - 19s 269ms/step - loss: 1.6406 - acc:
Epoch 7/30
71/71 [==============================] - ETA: 0s - loss: 1.5375 - acc: 0.6100
Epoch 00007: val_acc improved from 0.56679 to 0.60686, saving model to best_mo
71/71 [==============================] - 19s 269ms/step - loss: 1.5375 - acc:
Epoch 8/30
71/71 [==============================] - ETA: 0s - loss: 1.4402 - acc: 0.6342
Epoch 00008: val_acc improved from 0.60686 to 0.62556, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 1.4402 - acc:
Epoch 9/30
71/71 [==============================] - ETA: 0s - loss: 1.3486 - acc: 0.6548
Epoch 00009: val_acc improved from 0.62556 to 0.63090, saving model to best_mo
71/71 [==============================] - 19s 270ms/step - loss: 1.3486 - acc:
Epoch 10/30
71/71 [==============================] - ETA: 0s - loss: 1.2875 - acc: 0.6740
Epoch 00010: val_acc improved from 0.63090 to 0.64871, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 1.2875 - acc:
Epoch 11/30
71/71 [==============================] - ETA: 0s - loss: 1.2007 - acc: 0.6981
Epoch 00011: val_acc improved from 0.64871 to 0.66429, saving model to best_mo
71/71 [==============================] - 19s 269ms/step - loss: 1.2007 - acc:
Epoch 12/30
71/71 [==============================] - ETA: 0s - loss: 1.1551 - acc: 0.7050
Epoch 00012: val_acc improved from 0.66429 to 0.67097, saving model to best_mo
71/71 [==============================] - 19s 269ms/step - loss: 1.1551 - acc:
Epoch 13/30
71/71 [==============================] - ETA: 0s - loss: 1.0824 - acc: 0.7244
Epoch 00013: val_acc improved from 0.67097 to 0.67231, saving model to best_mo
71/71 [==============================] - 19s 269ms/step - loss: 1.0824 - acc:
Epoch 14/30
71/71 [==============================] - ETA: 0s - loss: 1.0375 - acc: 0.7337
Epoch 00014: val_acc improved from 0.67231 to 0.67409, saving model to best_mo
71/71 [==============================] - 19s 268ms/step - loss: 1.0375 - acc:
Epoch 15/30
71/71 [==============================] - ETA: 0s - loss: 0.9899 - acc: 0.7518
```

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

```
71/71 [==============================] - 2s 24ms/step - loss: 1.2067 - acc: 0.

 테스트 정확도 : 0.7168
```

```
1 epochs = range(1, len(history.history['acc'])+1)
2 plt.plot(epochs, history.history['loss'])
3 plt.plot(epochs, history.history['val_loss'])
4 plt.title('model loss')
5 plt.ylabel('loss')
6 plt.xlabel('epoch')
7 plt.legend(['train', 'test'], loc='upper left')
8 plt.show()
```



실제 뉴스 기사 크롤링 및 분류

```
1 !pip install beautifulsoup4
2 !pip install newspaper3k
3 !pip install konlpy
```

```
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist
Collecting newspaper3k
  Downloading newspaper3k-0.2.8-py3-none-any.whl (211 kB)
     |████████████████████████████████| 211 kB 12.5 MB/s
Collecting tldextract>=2.0.1
  Downloading tldextract-3.1.2-py2.py3-none-any.whl (87 kB)
     |████████████████████████████████| 87 kB 5.9 MB/s
Requirement already satisfied: PyYAML>=3.11 in /usr/local/lib/python3.7/dist-p
Collecting feedfinder2>=0.0.4
  Downloading feedfinder2-0.0.4.tar.gz (3.3 kB)
Requirement already satisfied: beautifulsoup4>=4.4.1 in /usr/local/lib/python3
Collecting jieba3k>=0.35.1
  Downloading jieba3k-0.35.1.zip (7.4 MB)
     |████████████████████████████████| 7.4 MB 70.5 MB/s
Collecting tinysegmenter==0.3
  Downloading tinysegmenter-0.3.tar.gz (16 kB)
Collecting cssselect>=0.9.2
  Downloading cssselect-1.1.0-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: Pillow>=3.3.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: requests>=2.10.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: nltk>=3.2.1 in /usr/local/lib/python3.7/dist-pa
Collecting feedparser>=5.2.1
  Downloading feedparser-6.0.8-py3-none-any.whl (81 kB)
     |████████████████████████████████| 81 kB 9.4 MB/s
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python
Requirement already satisfied: lxml>=3.6.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (
Collecting sgmllib3k
  Downloading sgmllib3k-1.0.0.tar.gz (5.8 kB)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dis
Collecting requests-file>=1.4
  Downloading requests_file-1.5.1-py2.py3-none-any.whl (3.7 kB)
Building wheels for collected packages: tinysegmenter, feedfinder2, jieba3k, s
```

```
  Building wheel for tinysegmenter (setup.py) ... done
  Created wheel for tinysegmenter: filename=tinysegmenter-0.3-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/df/67/41/faca10fa501ca010be41b4
  Building wheel for feedfinder2 (setup.py) ... done
  Created wheel for feedfinder2: filename=feedfinder2-0.0.4-py3-none-any.whl s
  Stored in directory: /root/.cache/pip/wheels/7f/d4/8f/6e2ca54744c9d7292d88dc
  Building wheel for jieba3k (setup.py) ... done
  Created wheel for jieba3k: filename=jieba3k-0.35.1-py3-none-any.whl size=739
  Stored in directory: /root/.cache/pip/wheels/4c/91/46/3c208287b726df325a5979
  Building wheel for sgmllib3k (setup.py) ... done
  Created wheel for sgmllib3k: filename=sgmllib3k-1.0.0-py3-none-any.whl size=
  Stored in directory: /root/.cache/pip/wheels/73/ad/a4/0dff4a6ef231fc0dfa12ff
Successfully built tinysegmenter feedfinder2 jieba3k sgmllib3k
Installing collected packages: sgmllib3k, requests-file, tldextract, tinysegme
Successfully installed cssselect-1.1.0 feedfinder2-0.0.4 feedparser-6.0.8 jieb
Collecting konlpy
  Downloading konlpy-0.5.2-py2.py3-none-any.whl (19.4 MB)
     |████████████████████████████████| 19.4 MB 1.2 MB/s
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: tweepy>=3.7.0 in /usr/local/lib/python3.7/dist-
Collecting colorama
```

```
1 !git clone https://github.com/SOMJANG/Mecab-ko-for-Google-Colab.git
```

```
Cloning into 'Mecab-ko-for-Google-Colab'...
remote: Enumerating objects: 91, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 91 (delta 0), reused 0 (delta 0), pack-reused 90
Unpacking objects: 100% (91/91), done.
```

```
1 cd Mecab-ko-for-Google-Colab/
```

```
/content/Mecab-ko-for-Google-Colab
```

```
1 !bash install_mecab-ko_on_colab190912.sh
```

```
Installing konlpy.....
Requirement already satisfied: konlpy in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: JPype1>=0.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: beautifulsoup4==4.6.0 in /usr/local/lib/python3
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: tweepy>=3.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/d
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/d
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python
Done
Installing mecab-0.996-ko-0.9.2.tar.gz.....
Downloading mecab-0.996-ko-0.9.2.tar.gz.......
from https://bitbucket.org/eunjeon/mecab-ko/downloads/mecab-0.996-ko-0.9.2.tar
```

```
--2021-12-01 05:35:07--  https://bitbucket.org/eunjeon/mecab-ko/downloads/meca
Resolving bitbucket.org (bitbucket.org)... 104.192.141.1, 2406:da00:ff00::22c0
Connecting to bitbucket.org (bitbucket.org)|104.192.141.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://bbuseruploads.s3.amazonaws.com/eunjeon/mecab-ko/downloads/me
--2021-12-01 05:35:08--  https://bbuseruploads.s3.amazonaws.com/eunjeon/mecab-
Resolving bbuseruploads.s3.amazonaws.com (bbuseruploads.s3.amazonaws.com)... 5
Connecting to bbuseruploads.s3.amazonaws.com (bbuseruploads.s3.amazonaws.com)|
HTTP request sent, awaiting response... 200 OK
Length: 1414979 (1.3M) [application/x-tar]
Saving to: ‘mecab-0.996-ko-0.9.2.tar.gz’

mecab-0.996-ko-0.9. 100%[===================>]   1.35M  2.61MB/s    in 0.5s

2021-12-01 05:35:08 (2.61 MB/s) - ‘mecab-0.996-ko-0.9.2.tar.gz’ saved [1414979

Done
Unpacking mecab-0.996-ko-0.9.2.tar.gz.......
Done
Change Directory to mecab-0.996-ko-0.9.2.......
installing mecab-0.996-ko-0.9.2.tar.gz........
configure
make
make check
make install
ldconfig
Done
Change Directory to /content
Downloading mecab-ko-dic-2.1.1-20180720.tar.gz.......
from https://bitbucket.org/eunjeon/mecab-ko-dic/downloads/mecab-ko-dic-2.1.1-2
--2021-12-01 05:36:17--  https://bitbucket.org/eunjeon/mecab-ko-dic/downloads/
Resolving bitbucket.org (bitbucket.org)... 104.192.141.1, 2406:da00:ff00::22cc
Connecting to bitbucket.org (bitbucket.org)|104.192.141.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://bbuseruploads.s3.amazonaws.com/a4fcd83e-34f1-454e-a6ac-c242c
--2021-12-01 05:36:17--  https://bbuseruploads.s3.amazonaws.com/a4fcd83e-34f1-
```

```
1 from konlpy.tag import Mecab
2 mecab = Mecab()
3 mecab.morphs('하하하하하이')
```

```
['하하하하', '하이']
```

BeautifulSoup의 사용법

```
1 from bs4 import BeautifulSoup
```

```
1 html = '''
2 <html>
3     <head>
4     </head>
5     <body>
6         <h1>장바구니
7             <p id='clothes' class= 'name' title = '라운드티'> 라운드티
8                 <span class = 'number'> 25 </span>
```

```
                <span class = 'price'> 29000 </span>
                <span class = 'menu'> 의류 </span>
                <a href = "http://www.naver.com"> 바로가기 </a>
            </p>
            <p id='watch' class= 'name' title = '시계'> 시계
                <span class = 'number'> 28 </span>
                <span class = 'price'> 32000 </span>
                <span class = 'menu'> 악세사리 </span>
                <a href = "http://www.facebook.com"> 바로가기 </a>
            </p>
        </h1>
    </body>
</html>
'''
```

```
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.select('body'))
```

```
[<body>
<h1>장바구니
            <p class="name" id="clothes" title="라운드티"> 라운드티
                <span class="number"> 25 </span>
<span class="price"> 29000 </span>
<span class="menu"> 의류 </span>
<a href="http://www.naver.com"> 바로가기 </a>
</p>
<p class="name" id="watch" title="시계"> 시계
                <span class="number"> 28 </span>
<span class="price"> 32000 </span>
<span class="menu"> 악세사리 </span>
<a href="http://www.facebook.com"> 바로가기 </a>
</p>
</h1>
</body>]
```

```
print(soup.select('p'))
```

```
[<p class="name" id="clothes" title="라운드티"> 라운드티
                <span class="number"> 25 </span>
<span class="price"> 29000 </span>
<span class="menu"> 의류 </span>
<a href="http://www.naver.com"> 바로가기 </a>
</p>, <p class="name" id="watch" title="시계"> 시계
                <span class="number"> 28 </span>
<span class="price"> 32000 </span>
<span class="menu"> 악세사리 </span>
<a href="http://www.facebook.com"> 바로가기 </a>
</p>]
```

```
print(soup.select('h1 .name .menu'))
```

```
[<span class="menu"> 의류 </span>, <span class="menu"> 악세사리 </span>]
```

```
print(soup.select('html > h1'))
```

```
print(soup.select('html > h1'))
```

```
[]
```

Newspaper3k 패키지

```
from newspaper import Article
```

```
url = 'https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=030&
```

```
article = Article(url, language='ko')
article.download()
article.parse()
```

```
print('기사 제목 : ')
print(article.title)
```

```
기사 제목 :
[AI 사피엔스 시대]자연어처리 기술, 컴퓨팅 파워 경쟁 시대로
```

```
print('기사 내용 :')
print(article.text)
```

```
기사 내용 :
[Copyright © 전자신문 & 전자신문인터넷, 무단전재 및 재배포 금지]

주로 아이디어와 기술력으로 경쟁했던 자연어처리 인공지능(AI) 분야는 점차 컴퓨팅 파워 싸움으로 무게 추기
```

BeautifulSoup와 newspaper3k를 통해 크롤러 만들기

```
import requests
import pandas as pd
from bs4 import BeautifulSoup
```

```
def make_urllist(page_num, code, date):
  urllist= []
  for i in range(1, page_num + 1):
    url = 'https://news.naver.com/main/list.nhn?mode=LSD&mid=sec&sid1='+str(code
    headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
    news = requests.get(url, headers=headers)

    soup = BeautifulSoup(news.content, 'html.parser')

    # CASE 1
    news_list = soup.select('.newsflash_body .type06_headline li dl')
    # CASE 2
    news_list.extend(soup.select('.newsflash_body .type06 li dl'))

    # 각 뉴스로부터 a 태그의 <a href ='주소'> 에서 '주소'만을 가져옵니다.
```

```
    # 각 뉴스로부터 a 태그인 <a href = 주소 > 에서 '주소' 만을 가져옵니다.
    for line in news_list:
        urllist.append(line.a.get('href'))
  return urllist
```

```
url_list = make_urllist(2, 101, 20200506)
print('뉴스 기사의 갯수 :', len(url_list))
```

```
뉴스 기사의 갯수 : 40
```

```
url_list[:5]
```

```
['https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=057&aic
 'https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=057&aic
 'https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=057&aic
 'https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=003&aic
 'https://news.naver.com/main/read.naver?mode=LSD&mid=sec&sid1=101&oid=057&aic
```

```
idx2word = {'101' : '경제', '102':'사회', '103' :'생활/문화', '105' : 'IT/과학'}
```

```
from newspaper import Article

# 데이터프레임을 생성하는 함수
def make_date(urllist, code):
    text_list = []
    for url in urllist:
        article = Article(url, language='ko')
        article.download()
        article.parse()
        text_list.append(article.text)

    # 데이터프레임의 'news'티 아래 파싱한 텍스트를 밸류로 붙여준다.
    df = pd.DataFrame({'news':text_list})

    # 데이터프레임의 'code'키 아래 한글 카테고리명을 붙여준다.
    df['code'] = idx2word[str(code)]
    return df
```

```
data = make_date(url_list, 101)
data[:10]
```

	news	code
0	고려은단이 5월을 맞아 응원 메시지를 공유하는 '5월 5글자로 응원 부탁해!' 이벤...	경제
1	코리아나화장품의 민감성 피부를 위한 저자극 스킨케어 브랜드 '프리엔제'가 마르고 건...	경제
2	서울장수주식회사가 부드럽고 달콤한 맛으로 인기를 모으고 있는 생막걸리 '인생막걸리'...	경제

데이터 수집 및 전처리

```
code_list = [102, 103, 105]
code_list
```

```
[102, 103, 105]
```

```
def make_total_data(page_num, code_list, date):
    df = None

    for code in code_list:
        url_list = make_urllist(page_num, code, date)
        df_temp = make_date(url_list, code)
        print(str(code) + '번 코드에 대한 데이터를 만들었습니다.')

        if df is not None:
            df = pd.concat([df, df_temp])
        else:
            df = df_temp

    return df
```

```
df = make_total_data(1, code_list, 20200506)
```

```
102번 코드에 대한 데이터를 만들었습니다.
103번 코드에 대한 데이터를 만들었습니다.
105번 코드에 대한 데이터를 만들었습니다.
```

```
print('뉴스 기사의 갯수 :', len(df))
```

```
뉴스 기사의 갯수 : 60
```

```
df.sample(10)
```

	news	code
19	넷플릭스와 유튜브, 페이스북 등에게 국내 이용자를 위한 '서비스 안정성'을 유지할 ...	IT/과학
2	황범순 의정부시 부시장 을지대학교 의정부캠퍼스 및 부속병원 공사현장 안전점검. 사진...	사회
18	기사 섹션 분류 안내\n\n기사의 섹션 정보는 해당 언론사의 분류를 따르고 있습니다...	IT/과학
10	기사 섹션 분류 안내\n\n기사의 섹션 정보는 해당 언론사의 분류를 따르고 있습니다...	생활/문화
13	블랙홀을 품은 삼중성계 HR 6819 상상도 [ESO/L. Calcada 제공/ 재...	IT/과학

```
1 df = make_total_data(100, code_list, 20200506)
```

```
102번 코드에 대한 데이터를 만들었습니다.
103번 코드에 대한 데이터를 만들었습니다.
105번 코드에 대한 데이터를 만들었습니다.
```

4	지난 2016년 포항공대에 구죽한 4세대 선형 방사광가속기. /연합뉴스 지난 201...	IT/과학

```
1 import os
2
3 csv_path = '../news_data.csv'
4 df.to_csv(csv_path, index=False)
```

```
1 if os.path.exists(csv_path):
2     print('{} File Saved!'.format(csv_path))
```

```
../news_data.csv File Saved!
```

```
1 #csv_path = '구글드라이브 주소 경로'
2 df = pd.read_table(csv_path, sep=',')
3 df.head()
```

	news	code
0	파주시청. 사진제공=파주시 파주시청. 사진제공=파주시\n\n[파주=파이낸셜뉴스 강근...	사회
1	동영상 뉴스\n\n이천 물류창고 화재 발화지점으로 지목된 지하 2층에서 산소절단기의...	사회
2	황범순 의정부시 부시장 을지대학교 의정부캠퍼스 및 부속병원 공사현장 안전점검. 사진...	사회
3	귀갓길 여성을 쫓아가 성범죄를 시도한 20대 남성이 구속됐습니다.서울 강남경찰서는 ...	사회
4	(서울=연합뉴스) 대한약사회가 6일부터 코로나바이러스 감염증 대응 체계를 '사회적 ...	사회

```
1 df['news'] = df['news'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]", "")
2 df['news']
```

```
0       파주시청사진제공파주시 파주시청사진제공파주시파주파이낸셜뉴스 강근주 기자파주시는 일 관...
1       동영상 뉴스이천 물류창고 화재 발화지점으로 지목된 지하 층에서 산소절단기의 산소 공...
2       황범순 의정부시 부시장 을지대학교 의정부캠퍼스 및 부속병원 공사현장 안전점검사진제공...
3       귀갓길 여성을 쫓아가 성범죄를 시도한 대 남성이 구속됐습니다서울 강남경찰서는 강간상...
4       서울연합뉴스대한약사회가 일부터 코로나바이러스 감염증 대응 체계를 사회적 거리두기에서...
                              ...
4950    신종 코로나바이러스 감염증코로나사태 이후 가정의 달월에도 언택트비대면신풍속도가 이어...
4951    는 소비자로부터 월 이용료 만만원을 받고 초고속 인터넷을 제공한다그런 브로드밴드가와...
4952    머리를 긁고 있는 오랑우탄몸을 긁는 행동을 따라 하는 것은 부정적 감정과 관련이 있...
```

```
4953    가 오는 일 정식 출시하는 스마트폰 벨벳이 사실상 공짜폰이 될 전망이다단말기 가격 ...
4954    이미지제공게티이미지뱅크 이미지제공게티이미지뱅크전자신문 전자신문인터넷무단전재 및 재배...
Name: news, Length: 3987, dtype: object
```

```
1 len(df)
```

```
5249
```

```
1 print(df.isnull().sum())
```

```
news    0
code    0
dtype: int64
```

```
1 df.drop_duplicates(subset=['news'], inplace=True)
2 print('뉴스 기사의 갯수 :', len(df))
```

```
뉴스 기사의 갯수 : 3987
```

데이터 탐색

```
1 df['code'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe892c45cd0>/usr/local/lib/pythor
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183:
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183:
  font.set_text(s, 0, flags=flags)
```

```
print(df.groupby('code').size().reset_index(name='count'))
```

```
     code  count
0  IT/과학    903
1     사회   1670
2  생활/문화   1414
```

```
from konlpy.tag import Mecab
tokenizer = Mecab()
```

```
kor_text = '밤에 귀가하던 여성에게 범죄를 시도한 대 남성이 구속됐다서울 제주경찰서는 \
           상해 혐의로 씨를 구속해 수사하고 있다고 일 밝혔다씨는 지난달 일 피해 여성을 \
           인근 지하철 역에서부터 따라가 폭행을 시도하려다가 도망간 혐의를 받는다피해 \
           여성이 저항하자 놀란 씨는 도망갔으며 신고를 받고 주변을 수색하던 경찰에 \
           체포됐다피해 여성은 이 과정에서 경미한 부상을 입은 것으로 전해졌다'
```

```
print(tokenizer.morphs(kor_text))
```

```
['밤', '에', '귀가', '하', '던', '여성', '에게', '범죄', '를', '시도', '한', '대', '남성
```

불용어 제거

```
stopwords = ['에','는','은','을','했','에게','있','이','의','하','한','다','과','때문','할
```

```
def preprocessing(data):
    text_data = []

    for sentence in data:
        temp_data = []
        temp_data = tokenizer.morphs(sentence)
        temp_data = [word for word in temp_data if not word in stopwords]
```

```
        text_data.append(temp_data)

    text_data = list(map(' '.join, text_data))
    return text_data
```

```
text_data = preprocessing(df['news'])
```

```
print(text_data[0])
```

```
파주 시청 사진 제공 = 파주시 파주 시청 사진 제공 = 파주시 [ 파주 = 강근주 파 주 시 4 일 관내 취약
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

```
x_train, x_test, y_train, y_test = train_test_split(text_data, df['code'], rando
```

```
print('훈련용 뉴스 기사의 갯수 : ', len(x_train))
print('테스트용 뉴스 기사의 갯수 : ', len(x_test))
print('훈련용 레이블의 갯수 : ', len(y_train))
print('테스트용 레이블의 갯수 : ', len(y_test))
```

```
훈련용 뉴스 기사의 갯수 :  2990
테스트용 뉴스 기사의 갯수 :  997
훈련용 레이블의 갯수 :  2990
테스트용 레이블의 갯수 :  997
```

```
# DTM
count_vect = CountVectorizer()
x_train_couts = count_vect.fit_transform(x_train)
```

```
# TF-IDF 행렬
tfidf_transformer = TfidfTransformer()
x_train_tfidf = tfidf_transformer.fit_transform(x_train_couts)
```

```
# 모델 학습 (나이브베이즈모델)
clf = MultinomialNB().fit(x_train_tfidf, y_train)
```

평가 및 테스트

```
def tfidf_vectorizer(data):
    data_counts = count_vect.transform(data)
    data_tfidf = tfidf_transformer.transform(data_counts)
    return data_tfidf
```

```
new_sent = preprocessing("민주당 일각에서 법사위의 체계 자구 심사 기능을 없애야 한다는 "
```

```
new_sent = preprocessing(["한주형 컬럼에서 법사위의 체계·자구 심사 기능을 없애야 한다는 \
                          주장이 나오는데 대해 “체계·자구 심사가 법안 지연의 수단으로 \
                         쓰이는 것은 바람직하지 않다”면서도 “국회를 통과하는 법안 중 위헌\
                         법률이 1년에 10건 넘게 나온다. 그런데 체계·자구 심사까지 없애면 매우
])
```

```
print(clf.predict(tfidf_vectorizer(new_sent)))
```

```
['사회']
```

```
new_sent = preprocessing(["인도 로맨틱 코미디 영화 <까립까립 싱글>(2017)을 봤을 때 나는 두 눈
                         저 사람이 남자 주인공이라고? 노안에 가까운 이목구비와 기름때로 뭉친 파
                         대충 툭툭 던지는 말투 등 전혀 로맨틱하지 않은 외모였다. 반감이 일면서
                         ‘난 외모지상주의자가 아니다’라고 자부했던 나에 대해 회의가 들었다.\
                          티브이를 꺼버릴까? 다른 걸 볼까? 그런데, 이상하다. 왜 이렇게 매력 있
                           같이 툭 불거진 눈망울 안에는 어떤 인도 배우에게서도 느끼지 못한 \
                           부드러움과 선량함, 무엇보다 슬픔이 있었다. 2시간 뒤 영화가 끝나고 ㄴ
```

```
print(clf.predict(tfidf_vectorizer(new_sent)))
```

```
['생활/문화']
```

```
new_sent = preprocessing(["20분기 연속으로 적자에 시달리는 LG전자가 브랜드 이름부터 성능, 디자인
                         적용한 LG 벨벳은 등장 전부터 온라인 커뮤니티를 뜨겁게 달궜다. 사용자들
                         “슬림하다”는 반응을 보이며 LG 벨벳에 대한 기대감을 드러냈다."])
```

```
print(clf.predict(tfidf_vectorizer(new_sent)))
```

```
['IT/과학']
```

```
y_pred = clf.predict(tfidf_vectorizer(x_test))
print(metrics.classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

         IT/과학       0.93      0.79      0.85       240
            사회       0.79      0.91      0.85       425
         생활/문화       0.85      0.78      0.81       332

      accuracy                           0.84       997
     macro avg       0.86      0.83      0.84       997
  weighted avg       0.85      0.84      0.84       997
```

```
1
```