```
import pandas as pd
import urllib.request
urllib.request.urlretrieve("https://raw.githubusercontent.com/franciscadias/data/master/abcnews-dat
data = pd.read_csv('abcnews-date-text.csv', error_bad_lines=False)

print(len(data))
```

```
1082168
```

```
print(data.head(5))
```

```
   publish_date                                     headline_text
0      20030219  aba decides against community broadcasting lic...
1      20030219     act fire witnesses must be aware of defamation
2      20030219     a g calls for infrastructure protection summit
3      20030219           air nz staff in aust strike for pay rise
4      20030219      air nz strike to affect australian travellers
```

```
text = data[['headline_text']]

import nltk
nltk.download('punkt')
# word tokenization
text['headline_text'] = text.apply(lambda row: nltk.word_tokenize(row['headline_text']), axis=1)

# stop words removal
from nltk.corpus import stopwords
nltk.download('stopwords')

stop = stopwords.words('english')
text['headline_text'] = text['headline_text'].apply(lambda x: [word for word in x if word not in (s

print(text.head(5))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
                                     headline_text
0   [aba, decides, community, broadcasting, licence]
1    [act, fire, witnesses, must, aware, defamation]
2     [g, calls, infrastructure, protection, summit]
3          [air, nz, staff, aust, strike, pay, rise]
4  [air, nz, strike, affect, australian, travellers]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  del sys.path[0]
```

```
# Lemmatization

from nltk.stem import WordNetLemmatizer
```

```
nltk.download('wordnet')

text['headline_text'] = text['headline_text'].apply(lambda x: [WordNetLemmatizer().lemmatize(word,

tokenized_doc = text['headline_text'].apply(lambda x: [word for word in x if len(word) > 3])

print(tokenized_doc[:5])
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

0       [decide, community, broadcast, licence]
1      [fire, witness, must, aware, defamation]
2    [call, infrastructure, protection, summit]
3                   [staff, aust, strike, rise]
4      [strike, affect, australian, travellers]
Name: headline_text, dtype: object
```

```
# 역토큰화 (토큰화 작업을 되돌림)
detokenized_doc = []
for i in range(len(text)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)

text['headline_text'] = detokenized_doc # 다시 text['headline_text']에 재저장

from sklearn.feature_extraction.text import TfidfVectorizer
# 상위 1,000개의 단어를 보존
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000)
# (과제)max_features  =3000 , 5000
X = vectorizer.fit_transform(text['headline_text'])
X.shape # TF-IDF 행렬의 크기 확인
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  import sys
(1082168, 1000)
```

```
# 역토큰화 (토큰화 작업을 되돌림)
detokenized_doc = []
for i in range(len(text)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)

text['headline_text'] = detokenized_doc # 다시 text['headline_text']에 재저장

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer2 = TfidfVectorizer(stop_words='english', max_features= 3000)
```

```
Y = vectorizer2.fit_transform(text['headline_text'])
Y.shape # TF-IDF 행렬의 크기 확인
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  import sys
(1082168, 3000)
```

```
# 역토큰화 (토큰화 작업을 되돌림)
detokenized_doc = []
for i in range(len(text)):
    t = ' '.join(tokenized_doc[i])
    detokenized_doc.append(t)

text['headline_text'] = detokenized_doc # 다시 text['headline_text']에 재저장

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer3 = TfidfVectorizer(stop_words='english', max_features= 5000)

Z = vectorizer3.fit_transform(text['headline_text'])
Z.shape # TF-IDF 행렬의 크기 확인
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  import sys
(1082168, 5000)
```

```
from sklearn.decomposition import LatentDirichletAllocation

lda_model=LatentDirichletAllocation(n_components=10,learning_method='online',random_state=777,max_i
lda_top=lda_model.fit_transform(X)

terms = vectorizer.get_feature_names() # 단어 집합. 1,000개의 단어가 저장됨.

def get_topics(components, feature_names, n=10):   # (과제)n=10
  for idx, topic in enumerate(components):
    print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(2))
for i in topic.argsort()[:-n - 1:-1]])

get_topics(lda_model.components_,terms)
```

```
Topic 1: [('government', 8725.19), ('sydney', 8393.29), ('queensland', 7720.12), ('change', 5874.27), ('hom
Topic 2: [('australia', 13691.08), ('australian', 11088.95), ('melbourne', 7528.43), ('world', 6707.7), ('s
Topic 3: [('death', 5935.06), ('interview', 5924.98), ('kill', 5851.6), ('jail', 4632.85), ('life', 4275.27
Topic 4: [('house', 6113.49), ('2016', 5488.19), ('state', 4923.41), ('brisbane', 4857.21), ('tasmania', 46
Topic 5: [('court', 7542.74), ('attack', 6959.64), ('open', 5663.0), ('face', 5193.63), ('warn', 5115.01),
Topic 6: [('market', 5545.86), ('rural', 5502.89), ('plan', 4828.71), ('indigenous', 4223.4), ('power', 396
Topic 7: [('charge', 8428.8), ('election', 7561.63), ('adelaide', 6758.36), ('make', 5658.99), ('test', 506
Topic 8: [('police', 12092.44), ('crash', 5281.14), ('drug', 4290.87), ('beat', 3257.58), ('rise', 2934.92)
Topic 9: [('fund', 4693.03), ('labor', 4047.69), ('national', 4038.68), ('council', 4006.62), ('claim', 360
Topic 10: [('trump', 11966.41), ('perth', 6456.53), ('report', 5611.33), ('school', 5465.06), ('woman', 545
```

```
lda_model_Y=LatentDirichletAllocation(n_components=10,learning_method='online',random_state=777,max
lda_top=lda_model_Y.fit_transform(Y)

terms = vectorizer2.get_feature_names() # 단어 집합. 3,000개의 단어가 저장됨.
def get_topics(components, feature_names, n=10):   # (과제)n=10
  for idx, topic in enumerate(components):
    print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(2))
for i in topic.argsort()[:-n - 1:-1]])

get_topics(lda_model_Y.components_,terms)
```

```
Topic 1: [('government', 7189.41), ('rural', 4815.37), ('home', 4601.0), ('hospital', 3603.5), ('fund', 306
Topic 2: [('australia', 10776.06), ('adelaide', 5572.36), ('make', 4408.25), ('tasmanian', 3993.18), ('tasm
Topic 3: [('trump', 9074.61), ('house', 4980.23), ('test', 3981.1), ('national', 3343.49), ('family', 3193.
Topic 4: [('court', 5986.55), ('change', 4761.32), ('year', 4626.85), ('face', 4117.53), ('donald', 3721.05
Topic 5: [('attack', 5430.09), ('perth', 5236.15), ('open', 4393.39), ('indigenous', 3424.3), ('2015', 3201
Topic 6: [('melbourne', 6123.73), ('election', 6091.44), ('school', 4508.48), ('warn', 4104.63), ('drug', 3
Topic 7: [('south', 5488.55), ('plan', 4829.94), ('market', 4706.07), ('woman', 4548.8), ('report', 4520.94
Topic 8: [('australian', 8773.55), ('sydney', 6729.42), ('world', 5247.38), ('country', 4737.61), ('kill',
Topic 9: [('queensland', 6273.39), ('canberra', 4964.87), ('interview', 4437.44), ('brisbane', 4032.89), ('
Topic 10: [('charge', 6918.42), ('police', 5855.52), ('murder', 5302.26), ('death', 4789.43), ('crash', 435
```

```
lda_model=LatentDirichletAllocation(n_components=10,learning_method='online',random_state=777,max_i
lda_top=lda_model.fit_transform(Z)

terms = vectorizer3.get_feature_names() # 단어 집합. 5,000개의 단어가 저장됨.

def get_topics(components, feature_names, n=10):   # (과제)n=10
  for idx, topic in enumerate(components):
    print("Topic %d:" % (idx+1), [(feature_names[i], topic[i].round(2))
for i in topic.argsort()[:-n - 1:-1]])

get_topics(lda_model.components_,terms)
```

```
Topic 1: [('government', 6740.89), ('attack', 5055.92), ('home', 4300.02), ('miss', 3392.94), ('labor', 308
Topic 2: [('sydney', 6259.51), ('canberra', 4651.42), ('change', 4448.86), ('market', 4437.78), ('year', 42
Topic 3: [('police', 6239.97), ('charge', 5411.59), ('court', 4459.22), ('report', 4192.1), ('murder', 4053
Topic 4: [('trump', 8500.1), ('australian', 8121.97), ('election', 5714.34), ('adelaide', 5189.51), ('make'
Topic 5: [('south', 5141.55), ('2016', 4280.73), ('live', 4140.6), ('crash', 4100.22), ('warn', 3808.13), (
Topic 6: [('perth', 4882.96), ('kill', 4368.32), ('china', 3453.81), ('accuse', 3062.13), ('work', 2653.12)
Topic 7: [('australia', 10018.61), ('queensland', 5823.94), ('house', 4598.21), ('school', 4185.44), ('open
Topic 8: [('melbourne', 5721.55), ('tasmania', 3573.78), ('plan', 3542.0), ('league', 3108.2), ('arrest', 2
Topic 9: [('record', 3030.08), ('lose', 2643.41), ('farm', 2539.57), ('federal', 2484.52), ('return', 2482.
Topic 10: [('world', 4831.66), ('rural', 4616.27), ('country', 4584.93), ('coast', 4087.57), ('state', 3834
```

```
from sklearn.manifold import TSNE
import matplotlib.font_manager as fm
import matplotlib.pyplot as plt
```

```
tsne = TSNE(n_components=2, n_iter=10000, verbose=1)
Z = tsne.fit_transform(X.T)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 1000 samples in 0.010s...
[t-SNE] Computed neighbors for 1000 samples in 0.371s...
```

```
[t-SNE] Computed conditional probabilities for sample 1000 / 1000
[t-SNE] Mean sigma: 4.222419
[t-SNE] KL divergence after 250 iterations with early exaggeration: 152.620392
[t-SNE] KL divergence after 1150 iterations: 2.468912
```

```
print(Z[0:5])
print('Top words: ', len(Z))
```

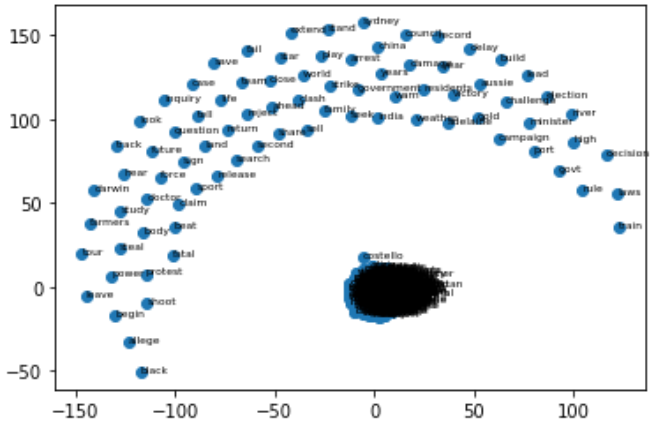
```
[[ -1.8148546  -4.83768  ]
 [ -1.5498776  -4.9258566]
 [ -1.0407622  -5.263114 ]
 [ -6.3660207  -5.152963 ]
 [  5.45876   -16.781092 ]]
Top words:  1000
```

```
tfidf_dict = vectorizer.get_feature_names()
print(tfidf_dict)
```

```
['2013', '2014', '2015', '2016', 'abbott', 'aboriginal', 'abuse', 'accc', 'accept', 'access', 'accident', '
```

```
fontprop = fm.FontProperties(size=6)
plt.scatter(Z[:,0], Z[:,1])
for i in range(len(tfidf_dict)):
    plt.annotate(s=tfidf_dict[i].encode("utf8").decode("utf8"), xy=(Z[i,0], Z[i,1]),fontProperties

plt.draw()
```



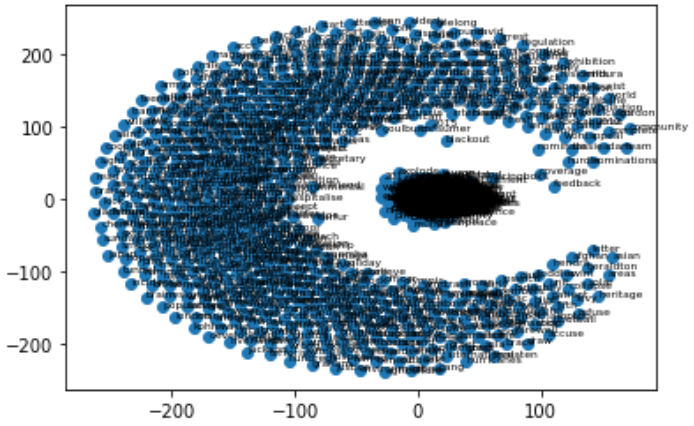

```
tfidf_dict1 = vectorizer2.get_feature_names()
print(tfidf_dict1)
tsne = TSNE(n_components=2, n_iter=10000, verbose=1)
Z = tsne.fit_transform(Y.T)
```

```
['1000', '100m', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', 'abalone', 'abandon', 'aba
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 3000 samples in 0.012s...
[t-SNE] Computed neighbors for 3000 samples in 0.922s...
[t-SNE] Computed conditional probabilities for sample 1000 / 3000
[t-SNE] Computed conditional probabilities for sample 2000 / 3000
[t-SNE] Computed conditional probabilities for sample 3000 / 3000
[t-SNE] Mean sigma: 1.902923
[t-SNE] KL divergence after 250 iterations with early exaggeration: 120.407135
[t-SNE] KL divergence after 5500 iterations: 2.285638
```

```
fontprop = fm.FontProperties(size=6)
```

```
plt.scatter(Z[:,0],·Z[:,1])
for·i·in·range(len(tfidf_dict1)):
····plt.annotate(s=tfidf_dict1[i].encode("utf8").decode("utf8"),·xy=(Z[i,0],·Z[i,1]),fontProperties

plt.draw()
```

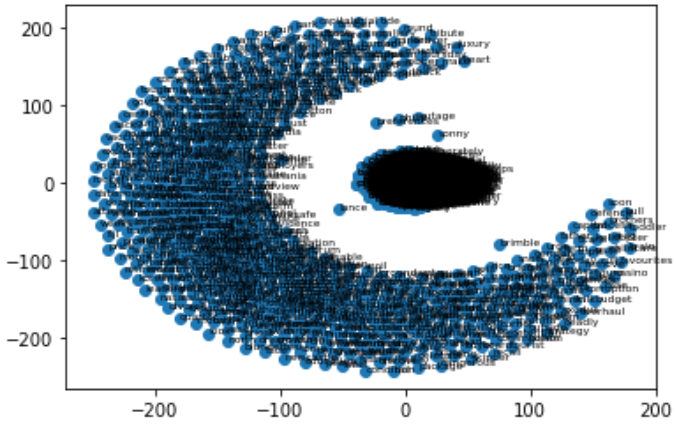


```
tfidf_dict3 = vectorizer3.get_feature_names()
print(tfidf_dict3)
tsne = TSNE(n_components=2, n_iter=10000, verbose=1)
Z = tsne.fit_transform(Z.T)
```

```
['1000', '10000', '100000', '100k', '100m', '12yo', '14yo', '2000', '2004', '2005', '2006', '2007', '2008',
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.017s...
[t-SNE] Computed neighbors for 5000 samples in 1.319s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 1.539092
[t-SNE] KL divergence after 250 iterations with early exaggeration: 139.681656
[t-SNE] KL divergence after 5650 iterations: 3.012808
```

```
fontprop = fm.FontProperties(size=6)
plt.scatter(Z[:,0], Z[:,1])
for i in range(len(tfidf_dict3)):
    plt.annotate(s=tfidf_dict3[i].encode("utf8").decode("utf8"), xy=(Z[i,0], Z[i,1]),fontProperties

plt.draw()
```



✓ 24초 오전 12:06에 완료됨