자연어처리 과제#3

1. 참고자료의 21-03절에 있는 LDA 분석 프로그램을 수행하여 최종 결과를 얻음

데이터 읽기

```
import pandas as pd
import urllib.request
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation

file_path, _ =
urllib.request.urlretrieve("https://raw.githubusercontent.com/ukairia777/tensorflow-nlp-tutorial/main/19.%20Topic%20Modeling%20(LDA%2C%20BERT-Based)/dataset/abcnews-date-text.csv")

                           # error_bad_lines (X)
data = pd.read_csv(file_path, on_bad_lines='skip')

print('뉴스 제목 개수 :',len(data))
```

```
뉴스 제목 개수 : 1082168
```

urllib.request.urlretrieve 함수에 입력된 링크의 csv 파일을 사용하기 위해 file_path에 저장해주었고, pd.read_csv에 file_path 를 사용해주었다. error_bad_lines는 최근 버전의 pandas에서는 더 이상 지원되지 않는 매개변수이다. 그 대신에 on_bad_lines라는 매개변수를 사용했는데, on_bad_lines='skip'은 error_bad_lines=False와 같은 기능을 수행한다.

데이터 내용

```
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
```

데이터 일부 보기

```
text = data[['headline_text']]
text['headline_text'] = text.apply(lambda row:
nltk.word_tokenize(row['headline_text']), axis=1)

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

```
                                       headline_text
0  [aba, decides, against, community, broadcastin...
1  [act, fire, witnesses, must, be, aware, of, de...
2  [a, g, calls, for, infrastructure, protection,...
3  [air, nz, staff, in, aust, strike, for, pay, r...
4  [air, nz, strike, to, affect, australian, trav...
```

텍스트 전처리

```
# 불용어 제거
stop_words = stopwords.words('english')
text['headline_text'] = text['headline_text'].apply(lambda x: [word for
word in x if word not in (stop_words)])

# Lemmatization
text['headline_text'] = text['headline_text'].apply(lambda x:
[WordNetLemmatizer().lemmatize(word, pos='v') for word in x])

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

print(tokenized_doc[:5])
```

```
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
```

tf-idf 행렬 만들기

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

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

from sklearn.feature_extraction.text import TfidfVectorizer

# 상위 1,000개의 단어를 보존
vectorizer = TfidfVectorizer(stop_words='english', max_features= 1000)
X = vectorizer.fit_transform(text['headline_text'])

# TF-IDF 행렬의 크기 확인
print('TF-IDF 행렬의 크기 :',X.shape)
```

```
TF-IDF 행렬의 크기 : (1082168, 1000)
```

토픽 모델링

```
from sklearn.decomposition import LatentDirichletAllocation

lda_model=LatentDirichletAllocation(n_components=10,learning_method='onlin
e',random_state=777,max_iter=1)
lda_top=lda_model.fit_transform(X)

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

def get_topics(components, feature_names, n=5):
    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: [('australia', 20556.0), ('sydney', 11219.29), ('melbourne',
8765.73), ('kill', 6646.06), ('court', 6004.12)]
Topic 2: [('coronavirus', 41719.62), ('covid', 28960.68), ('government',
9793.89), ('change', 7576.98), ('home', 7457.74)]
Topic 3: [('south', 7102.98), ('death', 6825.39), ('speak', 5402.31),
('care', 4521.48), ('interview', 4058.71)]
Topic 4: [('donald', 8536.49), ('restrictions', 6456.4), ('world',
6320.61), ('state', 6087.5), ('water', 4219.67)]
Topic 5: [('vaccine', 8040.87), ('open', 6915.17), ('coast', 5990.55),
('warn', 5472.84), ('morrison', 5247.93)]
Topic 6: [('trump', 14878.13), ('charge', 7717.96), ('health', 6836.95),
('murder', 6663.45), ('house', 6624.13)]
Topic 7: [('australian', 13885.88), ('queensland', 13373.53), ('record',
9037.88), ('test', 7713.9), ('help', 5922.05)]
Topic 8: [('case', 13146.83), ('police', 11143.1), ('live', 7528.03),
('border', 6855.54), ('tasmania', 5664.64)]
Topic 9: [('victoria', 11777.5), ('school', 6009.78), ('attack', 5503.39),
('national', 4672.53), ('concern', 4112.28)]
Topic 10: [('election', 8942.32), ('news', 8568.78), ('china', 8452.56),
('people', 6645.62), ('make', 6482.1)]
```

TfidfVectorizer에서 get_feature_names() 메서드는 scikit-learn의 최신 버전에서 더 이상 사용되지 않는다. 대신에 get_feature_names_out() 메서드로 대체되었다.

---

2. 참고자료에서는 각 topic의 단어들을 5개씩 출력했는데, 10개씩 출력하도록 프로그램을 수정

바로 위 **토픽 모델링** 부분의 코드에서 get_topics() 함수의 매개변수 n을 10으로 바꿔주었다.

```
from sklearn.decomposition import LatentDirichletAllocation

lda_model=LatentDirichletAllocation(n_components=10,learning_method='onlin
e', random_state=777,max_iter=1)
lda_top=lda_model.fit_transform(X)

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

def get_topics(components, feature_names, 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: [('australia', 20556.0), ('sydney', 11219.29), ('melbourne',
8765.73), ('kill', 6646.06), ('court', 6004.12), ('crash', 5938.8),
('years', 5867.84), ('face', 5781.98), ('woman', 5640.01), ('life',
4594.51)]
Topic 2: [('coronavirus', 41719.62), ('covid', 28960.68), ('government',
9793.89), ('change', 7576.98), ('home', 7457.74), ('year', 6506.2),
('market', 5801.69), ('arrest', 4835.59), ('rise', 4747.81), ('north',
4688.49)]
Topic 3: [('south', 7102.98), ('death', 6825.39), ('speak', 5402.31),
('care', 4521.48), ('interview', 4058.71), ('workers', 3706.18), ('talk',
3640.16), ('final', 3577.07), ('drum', 3467.18), ('search', 3207.39)]
Topic 4: [('donald', 8536.49), ('restrictions', 6456.4), ('world',
6320.61), ('state', 6087.5), ('water', 4219.67), ('storm', 3361.63),
('mark', 3096.44), ('save', 3089.93), ('extend', 3041.62), ('india',
3021.49)]
Topic 5: [('vaccine', 8040.87), ('open', 6915.17), ('coast', 5990.55),
('warn', 5472.84), ('morrison', 5247.93), ('leave', 5157.45), ('bushfire',
5098.62), ('scott', 4992.32), ('women', 4963.77), ('return', 4778.59)]
Topic 6: [('trump', 14878.13), ('charge', 7717.96), ('health', 6836.95),
('murder', 6663.45), ('house', 6624.13), ('report', 5586.04), ('minister',
5260.79), ('federal', 4876.65), ('victorian', 4844.81), ('protest',
4641.06)]
Topic 7: [('australian', 13885.88), ('queensland', 13373.53), ('record',
9037.88), ('test', 7713.9), ('help', 5922.05), ('brisbane', 5829.45),
('family', 5479.77), ('shoot', 5099.24), ('miss', 4654.21), ('premier',
4495.1)]
Topic 8: [('case', 13146.83), ('police', 11143.1), ('live', 7528.03),
('border', 6855.54), ('tasmania', 5664.64), ('canberra', 5573.71),
('indigenous', 5009.47), ('high', 3750.2), ('release', 3741.03),
('finance', 3546.02)]
Topic 9: [('victoria', 11777.5), ('school', 6009.78), ('attack', 5503.39),
('national', 4672.53), ('concern', 4112.28), ('hospital', 3974.56),
('regional', 3799.94), ('climate', 3722.73), ('flood', 3601.97), ('need',
3537.71)]
Topic 10: [('election', 8942.32), ('news', 8568.78), ('china', 8452.56),
('people', 6645.62), ('make', 6482.1), ('adelaide', 5747.66), ('time',
4720.34), ('quarantine', 4692.07), ('australias', 4665.36), ('trial',
4599.47)]
```

3. 이 자료 9쪽에 있는 처리 단어 수를 2,000으로 변화시켜서 프로그램을 수행시키고 위의 2번 결과와 비교함

TfidfVectorizer함수의 매개변수 max_features를 2000으로 바꾸어주었다.

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

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

from sklearn.feature_extraction.text import TfidfVectorizer

# 상위 2,000개의 단어를 보존
vectorizer = TfidfVectorizer(stop_words='english', max_features= 2000)
X = vectorizer.fit_transform(text['headline_text'])

# TF-IDF 행렬의 크기 확인
print('TF-IDF 행렬의 크기 :',X.shape)
```

```
TF-IDF 행렬의 크기 : (1082168, 2000)
```

```
from sklearn.decomposition import LatentDirichletAllocation

lda_model=LatentDirichletAllocation(n_components=10,learning_method='onlin
e',
random_state=777,max_iter=1)
lda_top=lda_model.fit_transform(X)

terms = vectorizer.get_feature_names_out() # 단어 집합. 2,000개의 단어가 저장됨.

def get_topics(components, feature_names, 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: [('australia', 17728.36), ('queensland', 11973.44),
('government', 8704.03), ('news', 7076.6), ('home', 6612.9), ('border',
6282.06), ('house', 5869.66), ('world', 5401.4), ('brisbane', 5073.64),
('family', 4834.04)]
Topic 2: [('case', 12141.11), ('record', 8325.11), ('change', 6666.37),
('year', 5635.92), ('state', 5393.32), ('market', 5359.05), ('canberra',
4873.65), ('morrison', 4838.59), ('attack', 4671.76), ('rise', 4206.54)]
Topic 3: [('test', 6786.87), ('restrictions', 5999.36), ('warn', 4813.02),
('leave', 4474.92), ('north', 4143.5), ('concern', 3667.75), ('hospital',
3527.81), ('travel', 3345.94), ('west', 3212.2), ('hotel', 3164.88)]
Topic 4: [('south', 6255.95), ('face', 5033.46), ('court', 4656.46),
('arrest', 4269.48), ('miss', 4043.12), ('help', 3718.33), ('interview',
3541.83), ('regional', 3359.03), ('climate', 3302.91), ('president',
3238.87)]
Topic 5: [('melbourne', 7686.26), ('coronavirus', 5936.47), ('people',
5910.65), ('crash', 5203.23), ('australians', 3592.72), ('jail', 3502.79),
('woman', 3450.54), ('island', 3449.33), ('release', 3199.48), ('child',
3065.99)]
Topic 6: [('sydney', 9587.97), ('death', 5934.71), ('kill', 5691.96),
('shoot', 4500.78), ('women', 4369.12), ('care', 4107.75), ('fight',
3699.7), ('2020', 3401.79), ('community', 3392.93), ('business', 3346.61)]
Topic 7: [('police', 12117.96), ('health', 6170.62), ('open', 5831.46),
('school', 5313.18), ('speak', 4739.36), ('minister', 4649.3),
('bushfire', 4521.91), ('indigenous', 4352.34), ('time', 4072.75),
('covid19', 3189.53)]
Topic 8: [('trump', 12965.61), ('victoria', 10599.03), ('charge',
6706.23), ('murder', 5920.52), ('coast', 5207.3), ('years', 5129.18),
('scott', 4558.16), ('life', 3984.43), ('premier', 3970.63), ('dead',
3721.87)]
Topic 9: [('australian', 11764.16), ('election', 7664.38), ('donald',
7608.78), ('live', 6531.08), ('lockdown', 6097.18), ('make', 5540.09),
('adelaide', 5085.11), ('tasmania', 5007.74), ('quarantine', 4363.03),
('federal', 4362.3)]
Topic 10: [('coronavirus', 31127.73), ('covid', 25943.65), ('china',
7258.43), ('vaccine', 7138.89), ('report', 4905.54), ('perth', 4140.4),
('work', 4039.49), ('announce', 3886.75), ('royal', 3611.85), ('high',
3327.11)]
```

처리 단어 수가 1000개로 적을 때엔 각 topic에서 일부 단어들이 빈번하게 등장하며 지역 및 사건 중심의 단어들로 집중되어 있고, 처리 단어 수가 2000개로 많을 때엔 topic들이 더 다양한 단어로 구성되어 있는데, 이는 더 많은 단어들이 추가되면서 topic의 구성이 조금 더 세부적으로 나뉘었다는 것을 의미한다.

처리 단어 수가 늘어나면 모델이 더 많은 정보를 바탕으로 각 topic을 분류할 수 있기 때문에, topic에 포함되는 단어들이 더 세분화되고 다양해지는 것이 자연스러운 결과이다.