Data Analytics Week 8 Assignment

202011431 산업공학과 차승현

Text Mining

건국대학교

Analysis Procedure

```
In [21]: import pandas as pd
         import json

In [22]: with open('data_week8.json', 'r', encoding='utf8') as f:
             data = json.load(f)
         data
```

```
Out[22]: [{'index': 221,
           'prod_name': 'Amazon echo dot',
           'review_head': 'Awesome.  It helps a lot',
           'review_body': 'At first, I was reluctant to talk to a hockey puck.  Okay so I am old fashioned.  I use smart sockets in some areas
         in my home.  I have them connected to my phone.  But I found out you could program the Echo Dot to them.  I have gotten used to talking
         to it.  I now command it to turn on and turn off certain things around my home.  You can command it to guard your home.  You can ask wh
         ere your orders are.  If you say good morning it will give you information as to what happened on that date years ago.  You can joke wi
         th it too.  The things you can do with it are endless.'},
          {'index': 222,
           'prod_name': 'Amazon echo dot',
           'review_head': 'Love my Echo Dot!',
           'review_body': 'I purchased the Ring Doorbell at Best Buy and the Echo Dot was included with the purchase. Thanks Best Buy!  I love m
         y Echo Dot!  All you need to do is plug it in, get the app on your phone and follow the prompts.  I ask Alexa silly questions, I ask he
         r to play music, tell me jokes, set up my alarm, set up reminders, I ask for the weather, the time, etc.  love it!'},
          {'index': 223,
           'prod_name': 'Amazon echo dot',
           'review_head': 'Echo Dot',
           'review_body': 'This was a gift for my granddaughter' s friend.  He has MS and is confined in a wheelchair.  He could use his arms a
         couple year' s ago, but only his fingers now.  Echo has changed his life.  He can talk on the phone and reach his family and friends.
```

1) pandas와 json 모듈을 import하고, json file의 data를 불러온다.

```
In [26]: review_lst = []

         for i in range(len(data)):
             review_lst.append(data[i]['review_body'])

In [27]: review_lst
```

```
Out[27]: ['At first, I was reluctant to talk to a hockey puck.  Okay so I am old fashioned.  I use smart sockets in some areas in my home.  I h
         ave them connected to my phone.  But I found out you could program the Echo Dot to them.  I have gotten used to talking to it.  I now c
         ommand it to turn on and turn off certain things around my home.  You can command it to guard your home.  You can ask where your orders
         are.  If you say good morning it will give you information as to what happened on that date years ago.  You can joke with it too.  The
         things you can do with it are endless.',
          'I purchased the Ring Doorbell at Best Buy and the Echo Dot was included with the purchase. Thanks Best Buy!  I love my Echo Dot!  All
         you need to do is plug it in, get the app on your phone and follow the prompts.  I ask Alexa silly questions, I ask her to play music,
         tell me jokes, set up my alarm, set up reminders, I ask for the weather, the time, etc.  love it!',
          'This was a gift for my granddaughter' s friend.  He has MS and is confined in a wheelchair.  He could use his arms a couple year' s a
         go, but only his fingers now.  Echo has changed his life.  He can talk on the phone and reach his family and friends.  It was the best
         gift EVER.',
          'I really like this speaker it can do a lot of things like in the morning i have an alarm to go to work when i turn off the alarm i se
         t it up to tell me how is the traffic to work and tell me weather and if i want i could set it up to tell me also news, radio and more
         when i arrive From work i tell it to play music the sound is really good when im going to sleep i tell it to open sleep sounds or i say
         good night and it gives me a list of sounds it can make that will help me go to sleep i leave the sound the entire night it helps me go
         to sleep, I don' t have anything smart yet like light bulbs but if i did I would be able to do a lot more with this',
          'These Echoes are small but versatility and convenient.  A nice feature is that with 2 of them you can link them as stereo speakers
         (just for streaming music services).  I got one for free as part of a bundle with Echo Show and bought the second one for less than $25
         so it was a great deal.',
```

2) review_lst 라는 리스트를 제작하여, json file의 'review_body'부분을 리스트에 삽입하였다. 이외 index, prod_name, review_head는 분석에 포함하지 않았다.

```
In [29]: print(len(review_lst))

         print(review_lst[0])
```

```
100
At first, I was reluctant to talk to a hockey puck.  Okay so I am old fashioned.  I use smart sockets in some areas in my home.  I have t
hem connected to my phone.  But I found out you could program the Echo Dot to them.  I have gotten used to talking to it.  I now command i
t to turn on and turn off certain things around my home.  You can command it to guard your home.  You can ask where your orders are.  If y
ou say good morning it will give you information as to what happened on that date years ago.  You can joke with it too.  The things you ca
n do with it are endless.
```

리뷰 리스트의 길이는 100이고, 리뷰 리스트에 잘 저장이 된 모습을 확인할 수 있다.

```
In [31]: count = {} #동시출현 빈도가 저장될 dict

         for line in review_lst:
             words = list(set(line.split()))

             for i, a in enumerate(words):
                 for b in words[i+1:]:
                     if a>b:
                         count[b, a] = count.get((b, a), 0) + 1

                     else:
                         count[a, b] = count.get((a, b), 0) + 1
```

```
In [32]: count
```

```
Out[32]: {('guard', 'old'): 1,
          ('old', 'to'): 3,
          ('old', 'that'): 3,
          ('If', 'old'): 1,
          ('in', 'old'): 3,
          ('it.', 'old'): 1,
          ('old', 'smart'): 2,
          ('old', 'them.'): 2,
          ('information', 'old'): 1,
          ('old', 'reluctant'): 1,
          ('connected', 'old'): 1,
          ('old', 'out'): 2,
          ('old', 'orders'): 1,
          ('now', 'old'): 1,
          ('happened', 'old'): 1,
          ('off', 'old'): 1,
          ('old', 'you'): 2,
          ('my', 'old'): 4,
          ('and', 'old'): 4,
          ('old', 'too.'): 2
```

3) count라는 딕셔너리 형태의 변수에 review_lst의 문장들을 split하여 삽입하고, 연관 출현 단어의 빈도를 표시해보았다.

```
('first,', 'old'): 1,
('old', 'used'): 1,
('date', 'old'): 1,
('certain', 'old'): 1,
('gotten', 'old'): 1,
('hockey', 'old'): 1,
('old', 'your'): 2,
('are.', 'old'): 1,
```

그러나, ‘first,’나 ‘are.’같은 문장의 부호, ‘Dot’같이 대소문자 구분이 골고루 되지 않았기에, 단어의 전처리가 필요하다고 생각했다.

```
In [33]: from tensorflow.keras.preprocessing.text import text_to_word_sequence

In [34]: review_lst2 = []
         for i in range(len(review_lst)):
             review_lst2.append(text_to_word_sequence(review_lst[i]))

         review_lst2

Out[34]: [['at',
           'first',
           'i',
           'was',
           'reluctant',
           'to',
           'talk',
           'to',
           'a',
           'hockey',
           'puck',
           'okay',
           'so',
           'i',
           'am',
           'old',
           'fashioned',
           'i',
           'use',
```

4) 토큰화 기법을 사용했다. 토큰화 기법으로는 word_tokenize, WordPunctTokenizer, text_to_word_sequence, TreebankWordTokenizer 등 다양한 기법이 있었지만, '띄어쓰기'를 기준으로 단어를 토큰화시켜주는 기법은 text_to_word_sequence가 제일 적합하다고 판단하였다.

```
In [37]: import nltk

         from nltk.corpus import stopwords
```

```
In [66]: stop_words = list(set(stopwords.words('english')))

         review_lst_fin = []

         for w in range(len(review_lst2)):

             new_review = []

             for j in review_lst2[w]:

                 if j not in stop_words:
                     new_review.append(j)

             review_lst_fin.append(new_review)
```

```
In [67]: review_lst_fin
```

```
Out[67]: [['first',
           'reluctant',
           'talk',
           'hockey',
           'puck',
           'okay',
           'old',
           'fashioned',
           'use',
           'smart',
           'sockets',
           'areas',
           'home',
           'connected',
           'phone',
           'found',
           'could',
           'program',
           'echo',
```

5) 불용어를 제거해주었다. I, we, my 같은 분석의 의미가 낮은 단어를 제거해주는 모듈(nltk의 stopwords)을 사용하여 문장의 불용어를 제거하였다.

```
In [68]: dic = {}
         for i in range(len(review_lst_fin)):
             dic[i+1] = review_lst_fin[i]

In [69]: dic

Out[69]: {1: ['first',
              'reluctant',
              'talk',
              'hockey',
              'puck',
              'okay',
              'old',
              'fashioned',
              'use',
              'smart',
              'sockets',
              'areas',
              'home',
              'connected',
              'phone',
              'found',
              'could',
              'program',
              'echo',
```

6) 해당 불용어를 제거한 단어의 토큰을 각 문장마다 dictionary의 형태로 저장해주었다.

```
In [70]: import itertools as it

         count2 = {}

         for k, v in dic.items():
             for a, b in it.combinations(v, 2):
                 tmp = (a, b) if b<a else (b, a)

                 if count2.get(tmp, False):
                     count2[tmp] += 1

                 else:
                     count2[tmp] = 1
```

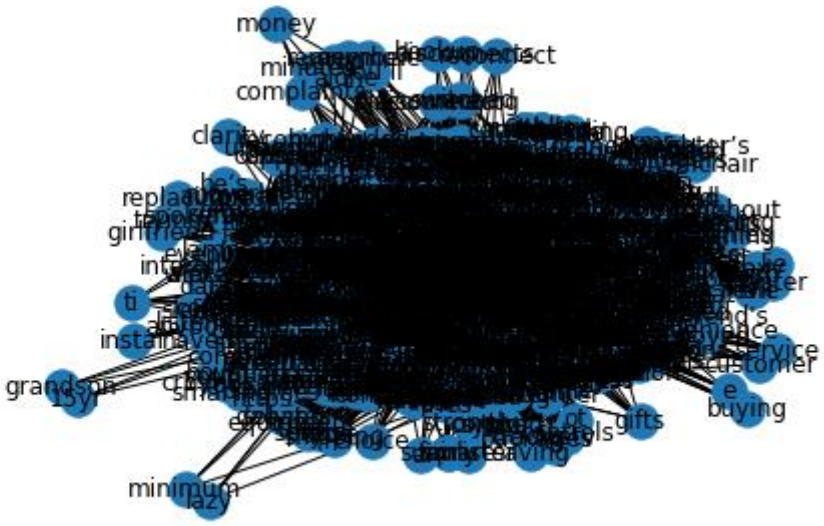
```
In [71]: count2

         ('first', 'fashioned'): 1,
         ('use', 'first'): 13,
         ('smart', 'first'): 8,
         ('sockets', 'first'): 1,
         ('first', 'areas'): 1,
         ('home', 'first'): 9,
         ('first', 'connected'): 4,
         ('phone', 'first'): 1,
         ('found', 'first'): 1,
         ('first', 'could'): 1,
         ('program', 'first'): 1,
         ('first', 'echo'): 10,
         ('first', 'dot'): 11,
         ('gotten', 'first'): 1,
         ('used', 'first'): 1,
         ('talking', 'first'): 1,
         ('first', 'command'): 2,
         ('turn', 'first'): 4,
```

7) 3)에서 실행했던 것과 마찬가지로, 단어의 토큰화와 불용어 제거를 마친 이후의 연관 단어 출현의 빈도를 나타내주었다.

```
In [99]: P = nx.Graph()
         P.add_weighted_edges_from(lst1)

In [102]: nx.draw_spring(P, with_labels=True)
          plt.show()
```



8) (단어1, 단어2, 빈도)로 이루어진 리스틀 input값으로 설정하고, 네트워크를 그려보았다.

```
In [104]: keyword = list(nx.closeness_centrality(P).keys())
          dgr = list(nx.degree_centrality(P).values())
          btw = list(nx.betweenness_centrality(P).values())
          cls = list(nx.closeness_centrality(P).values())

In [105]: df = pd.DataFrame({'keyword':keyword, 'Degree Centrality': dgr, 'Betweenness Centrality': btw, 'Closeness Centrality' : cls})

In [106]: df
```

Out [106]:

	keyword	Degree Centrality	Betweenness Centrality	Closeness Centrality
0	reluctant	0.035652	0.000000	0.493986
1	first	0.271304	0.004608	0.577309
2	talk	0.099130	0.000952	0.525354
3	hockey	0.058261	0.000139	0.512706
4	puck	0.058261	0.000139	0.512706
...	...	...	...	...
1146	catching	0.018261	0.000000	0.497405
1147	current	0.018261	0.000000	0.497405
1148	events	0.018261	0.000000	0.497405
1149	laughing	0.018261	0.000000	0.497405
1150	throughout	0.018261	0.000000	0.497405

1151 rows × 4 columns

9) keword와, degree centrality, betweenness centrality, closeness centrality를 사용하여 데이터프레임의 형태로 제작해주었다.

```
df.sort_values(by=['Degree Centrality'], ascending=False)[:10]
```

	keyword	Degree Centrality	Betweenness Centrality	Closeness Centrality
18	echo	0.699130	0.062908	0.767690
57	alexa	0.651304	0.055216	0.740502
19	dot	0.622609	0.037157	0.725095
61	music	0.618261	0.049440	0.722816
132	one	0.565217	0.039905	0.696126
50	love	0.555652	0.033957	0.691106
64	set	0.530435	0.022422	0.679669
141	great	0.528696	0.034522	0.678867
8	use	0.512174	0.026578	0.671337
9	smart	0.492174	0.023573	0.662442

> Degree Centrality를 기준으로 내림차순 정렬의 결과
(echo, alexa, dot, music, one, love, set, great, use, smart)

```
df.sort_values(by=['Betweenness Centrality'], ascending=False)[:10]
```

	keyword	Degree Centrality	Betweenness Centrality	Closeness Centrality
18	echo	0.699130	0.062908	0.767690
57	alexa	0.651304	0.055216	0.740502
61	music	0.618261	0.049440	0.722816
132	one	0.565217	0.039905	0.696126
19	dot	0.622609	0.037157	0.725095
141	great	0.528696	0.034522	0.678867
50	love	0.555652	0.033957	0.691106
8	use	0.512174	0.026578	0.671337
9	smart	0.492174	0.023573	0.662442
64	set	0.530435	0.022422	0.679669

> Betweenness Centrality를 기준으로 내림차순 정렬의 결과

(echo, alexa, music, one, dot, great, love, use, smart, set)

```
df.sort_values(by=['Closeness Centrality'], ascending=False)[:10]
```

	keyword	Degree Centrality	Betweenness Centrality	Closeness Centrality
18	echo	0.699130	0.062908	0.767690
57	alexa	0.651304	0.055216	0.740502
19	dot	0.622609	0.037157	0.725095
61	music	0.618261	0.049440	0.722816
132	one	0.565217	0.039905	0.696126
50	love	0.555652	0.033957	0.691106
64	set	0.530435	0.022422	0.679669
141	great	0.528696	0.034522	0.678867
8	use	0.512174	0.026578	0.671337
9	smart	0.492174	0.023573	0.662442

> Closness Centrality를 기준으로 내림차순 정렬의 결과
(echo, alexa, dot, music, one, love, set, great, use, smart)

세 가지의 Centrality 기준 정렬 결과, keyword의 centrality가 높은 단어들이 echo, alexa, dot, music, … 등등 매우 유사함을 확인할 수 있었다.

추가적으로, 위에서 시행한 (단어1, 단어2, 빈도)의 리스틀 데이터프레임화하고, 빈도수를 기준으로 내림차순 정렬을 진행해보았다.

```
df3 = df2.sort_values(by=['freq'], ascending=False)

df3
```

	term1	term2	freq
586	echo	dot	105
2871	one	echo	51
1016	echo	echo	49
1023	echo	alexa	43
309	use	echo	42
...	...	...	...
22179	i'm	bar	1
22180	living	i'm	1
22181	i'm	area	1
22182	i'm	fills	1
50763	throughout	day	1

해당 데이터프레임에서 빈도가 5 이상인 단어들로 다시 동일한 centrality 지표로 계산해보았다.

```
import numpy as np
import networkx as nx
import operator
import matplotlib.pyplot as plt

len((np.where(df3['freq'] >= 5))[0])
#빈도가 5개 이상인 것들만 추출하면 2311개가 나온다.
```

```
2311
```

```
G = nx.Graph()
for i in range(len((np.where(df3['freq'] >= 5))[0])):

    G.add_edge(df3['term1'][i], df3['term2'][i], weight=int(df3['freq'][i]))
```

```
dgr = nx.degree_centrality(G)
btw = nx.betweenness_centrality(G)
cls = nx.closeness_centrality(G)

# itemgetter(0): key 또는 itemgetter(1): value로 sort key, reverse=True (descending order)
sorted_dgr = sorted(dgr.items(), key=operator.itemgetter(1), reverse=True)
sorted_btw = sorted(btw.items(), key=operator.itemgetter(1), reverse=True)
sorted_cls = sorted(cls.items(), key=operator.itemgetter(1), reverse=True)
```

```
print("** degree **")
for x in range(10):
    print(sorted_dgr[x])
print("** betweenness **")
for x in range(10):
    print(sorted_btw[x])
print("** closeness **")
for x in range(10):
    print(sorted_cls[x])
```

```
** degree **
('could', 0.8220338983050848)
('echo', 0.7372881355932204)
('phone', 0.7203389830508474)
('turn', 0.6949152542372882)
('things', 0.6949152542372882)
('morning', 0.6779661016949152)
('tell', 0.6186440677966102)
('set', 0.6186440677966102)
('alarm', 0.6186440677966102)
('dot', 0.6016949152542372)
** betweenness **
('could', 0.1327737950076061)
('phone', 0.06188884835125602)
('echo', 0.06188884835125602)
('turn', 0.0376927690269715)
('things', 0.0376927690269715)
('morning', 0.0376927690269715)
('dot', 0.02839064536152757)
('ask', 0.02839064536152757)
('play', 0.0255387408798969)
('tell', 0.0255387408798969)
** closeness **
('could', 0.8489208633093526)
('phone', 0.7814569536423841)
('echo', 0.7814569536423841)
('turn', 0.7564102564102564)
('things', 0.7564102564102564)
('morning', 0.7564102564102564)
('play', 0.7151515151515152)
('tell', 0.7151515151515152)
('set', 0.7151515151515152)
('alarm', 0.7151515151515152)
```

빈도수가 5 이상인 단어 집합들의 결과를 가지고 centrality를 계산해본 결과, 이전 모든 단어 집합들을 삽입하였을 때와 상당히 다른 행태를 보임을 확인할 수 있었다.

모든 단어의 집합들을 삽입하였을 때, 데이터프레임에 출현하지 않았던

‘could’, ‘phone’ ‘turn’, ‘morning’, ‘things’ 등 새로운 단어의 출현을 확인할 수 있었는데, 여기서 could와 things는 불용어에 포함되어 제거되어야 하는 대상이라고 생각하였다. Text mining을 할 때에는 분석을 시행하기 전, 빈도수의 threshold를 잘 적용하고 분석을 거듭하며 불용어를 말끔히 전처리하여야 한다는 유의점을 시사할 수 있었다.

Reference
https://ngio.co.kr/9289