Data Analytics

Assignment -Text Mining(Keyword Network) -

윤장혁 교수님

산업공학과

201811527

이영은

Week8

- **주어진 리뷰데이터(data_week8.json)를 활용하여 키워드 동시출현 네트워크를 작성하고 centrality 분석**

  [분석 과정]

  1. python (jupyter notebook)을 이용하여 json 파일을 어절(키워드)로 나누어 각각의 리뷰별로 출현 단어를 찾았습니다.

  2. 1의 결과로 Document-Term matrix로 만들었습니다.

  3. 2의 전치행렬인 T-D matrix와 2의 행렬 곱으로 T-T matrix를 만들었습니다.

  4. 3의 결과를 dataframe과 csv 파일로 저장하였습니다.

  5. csv 파일을 gephi를 통하여 네트워크를 시각화 하였습니다.

■ File open, Tokenize

실행 화면

```
In [1]: import pandas as pd
        import json
        import re

        with open('/Users/leeyeongeun/Desktop/DataAnalytics/week8/data_week8.json', encoding = "utf-8")as json_file:
            json_data = json.load(json_file)

        review_body = []
        for i in range(0, len(json_data)):
            review_body.append(json_data[i]["review_body"].lower())

        review = []
        for i in range(0, len(review_body)):
            rev = re.sub('[^a-zA-Z0-9가-힣\s]', '', review_body[i])
            review.append(rev)

        #Tokenize
        import nltk
        rev_token = []
        from nltk.tokenize import word_tokenize
        for i in range(0, len(review)):
            rev_token.append(word_tokenize(review[i]))

        print(rev_token)
```

```
[['at', 'first', 'i', 'was', 'reluctant', 'to', 'talk', 'to', 'a', 'hockey', 'puck', 'okay', 'so', 'i', 'am', 'ol
d', 'fashioned', 'i', 'use', 'smart', 'sockets', 'in', 'some', 'areas', 'in', 'my', 'home', 'i', 'have', 'them', 'c
onnected', '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', 'o
n', 'and', 'turn', 'off', 'certain', 'things', 'around', 'my', 'home', 'you', 'can', 'command', 'it', 'to', 'guar
d', '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',
```

코드 설명

- pandas 와 json그리고 re를 import 하였습니다.
- week8 과제로 제공받은 json file을 열었고, 이를 로드 하였습니다.(json_data)
- review_body 부분을 리스트에 저장하기 위해서 review_body라는 list를 만들어서 한줄씩 append 하였고, 소문자와 대문자를 따로 구분하지 않기 위하여 .lower()을 사용하여 모두 소문

자로 변환하였습니다.
- 특수문자를 제거하기 위하여 review_body 리스트를 하나씩 돌면서 특수문자를 제거 한후, review 리스트에 append 하였습니다.
- # Tokenize : 토큰화 하기 위하여 nltk 와 word_tokenize를 import하였습니다.
- 어절로 키워드를 나누어 연관성을 분석하기 위하여 rev_token이라는 리스트를 만들어서 토큰화 한 키워드들을 각 review별로 리스트에 append하여 토큰화한 모든 리뷰를 2차원 리스트로 만들었습니다.
- 결과의 일부를 함께 첨부하였습니다.

■ Stop words제거, Stemming, Lemmatizing

실행 화면

```
In [2]: #stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

without_sw = []
for i in range(0, len(rev_token)):
    a = []
    for j in range(0, len(rev_token[i])):
        if rev_token[i][j] not in stop_words:
            a.append(rev_token[i][j])
    without_sw.append(a)

#stemming, lemmatizing
filter_words = []
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
stemmer = PorterStemmer()
lematizer = WordNetLemmatizer()
for i in range(0, len(without_sw)):
    stems = [stemmer.stem(token) for token in without_sw[i]]
    filter_words.append(stems)

for i in range(0, len(filter_words)):
    lemmas = [lematizer.lemmatize(token) for token in filter_words[i]]
    filter_words.append(lemmas)
print(filter_words)
```

```
[['first', 'reluct', 'talk', 'hockey', 'puck', 'okay', 'old', 'fashion', 'use', 'smart', 'socket', 'area', 'home',
'connect', 'phone', 'found', 'could', 'program', 'echo', 'dot', 'gotten', 'use', 'talk', 'command', 'turn', 'turn',
'certain', 'thing', 'around', 'home', 'command', 'guard', 'home', 'ask', 'order', 'say', 'good', 'morn', 'give', 'i
nform', 'happen', 'date', 'year', 'ago', 'joke', 'thing', 'endless'], ['purchas', 'ring', 'doorbel', 'best', 'buy',
'echo', 'dot', 'includ', 'purchas', 'thank', 'best', 'buy', 'love', 'echo', 'dot', 'need', 'plug', 'get', 'app', 'p
hone', 'follow', 'prompt', 'ask', 'alexa', 'silli', 'question', 'ask', 'play', 'music', 'tell', 'joke', 'set', 'ala
rm', 'set', 'remind', 'ask', 'weather', 'time', 'etc', 'love'], ['gift', 'granddaught', 'friend', 'ms', 'confin',
'wheelchair', 'could', 'use', 'arm', 'coupl', 'year', 'ago', 'finger', 'echo', 'chang', 'life', 'talk', 'phone', 'r
each', 'famili', 'friend', 'best', 'gift', 'ever'], ['realli', 'like', 'speaker', 'lot', 'thing', 'like', 'morn',
'alarm', 'go', 'work', 'turn', 'alarm', 'set', 'tell', 'traffic', 'work', 'tell', 'weather', 'want', 'could', 'se
```

코드 설명

- # Stopwords : Stopword를 제거하기 위하여 nltk 의 stopwords를 import 하였습니다.
- stop_words를 stopwords에서 'english'를 불러오고, 중복을 제거하기 위하여 set을 통해 제거하였습니다.
- without_sw라는 리스트를 만들어서 만약 토큰화 한 키워드가 stopwords의 리스트에 없다면, without_sw라는 리스트에 append 하였습니다.
- without_sw는 stopwords를 제거한 키워드들의 집합입니다.

- # Stemming, Lemmatizing
- stemming, lemmatizing을 하기 위하여 PorterStemmer와 WordNetLemmatizer를 import 하였

습니다.
- stemmer = PosterStemmer() , lematizer = WordNetLemmatizer()로 정의하였습니다.
- without_sw에 저장된 단어 중에서 stemmer를 통해서 stems을 구했고, 이를 filter_words에 append 하였습니다.
- 같은 방법으로 Lemmatizing하여 filter_words에 append 하였습니다.
- 결과의 일부를 함께 첨부하였습니다.

■ matrix 만들기

실행 화면

```
In [3]: mat = []
        for i in range(0, len(filter_words)):
            for j in range(0, len(filter_words[i])):
                mat.append(filter_words[i][j])
        mat = list(set(mat))
        print(mat)
```

```
['dose', 'pop', 'soon', 'black', 'temperatur', 'medicin', 'age', 'favorit', 'thru', 'higher', 'crucial', 'coordin',
'audio', 'main', 'area', '15yr', 'solid', '6', 'boyfriend', 'devic', 'similarli', 'anymor', 'fart', 'reason', 'powe
r', 'hockey', 'weather', 'smaller', 'televis', 'honestli', 'respons', 'physic', 'fidel', 'includ', 'someon', 'answe
r', 'girl', 'silli', 'deal', 'distort', 'name', 'entir', 'intergr', 'send', 'get', 'check', 'quit', 'constantli', 'wh
oever', 'everli', 'thousand', 'total', 'excel', 'josh', 'finish', 'hope', 'wretch', 'interact', 'fabric', 'creat', 'a
ccent', 'way', 'alarm', 'believ', 'fantast', 'basement', 'understandit', 'bed', 'bundl', 'leap', 'adult', 'learn', 'k
now', 'third', 'scratch', 'cook', 'whole', 'requir', 'properli', 'improv', 'accessori', 'xm', 'clariti', 'replac', 'l
ike', 'studio', 'abl', 'modern', 'parent', 'gon', 'want', 'version', 'bathroom', 'account', 'husband', 'outlet', 'uno
btrus', 'toshiba', 'receiv', 'triva', 'wouldnt', 'bulb', 'worri', 'odd', 'event', 'camera', 'echosystem', 'winter',
'constant', 'also', 'echo', 'around', 'definit', 'link', 'upper', 'lost', 'track', 'children', 'm', '65', 'read', 'le
ast', 'p', 'matter', 'long', 'nice', 'alex', 'whistl', 'arlo', 'depend', 'instruct', 'alway', 'friday', 'crisp', 'shu
ffl', 'clock', 'us', 'asid', 'will', 'theyr', 'httpsyoutubecrbyn8selei', 'offer', 'worth', 'cel', 'touch', 'sale', 'e
njoy', 'fashion', '21st', '110', 'hookup', 'wifi', 'experi', 'manner', 'host', 'etcyou', 'complic', 'granddaught', 'f
eel', 'probabl', 'saw', 'jump', '2017', 'often', 'last', 'temptat', 'light', '2nd', 'work', 'diver', 'afford', 'amazo
n', 'stuck', 'counter', 'activ', 'fairli', 'strongli', 'bunch', 'listen', 'wish', 'novelti', 'alexa', 'reach', 'brows
er', 'intend', 'other', 'stereo', 'think', 'household', 'correct', 'lamp', 'theater', 'besid', 'found', 'fact', 'son
g', 'invest', 'girlfriend', 'low', 'contact', 'custom', 'ta', 'box', 'grandson', 'thriller', 'join', 'let', 'sport',
'level', 'could', 'iheart', 'explain', 'busi', 'voic', 'everywher', 'impress', 'ring', 'word', 'brief', 'howev', 'swi
tch', 'lessus', 'save', 'gateway', 'isnt', 'setup', 'updat', 'drift', 'bit', 'standard', 'anoth', 'countri', 'distan
c', 'ai', 'eye', 'teach', 'ever', 'ill', 'arm', 'dont', 'keep', 'amaz', 'discov', 'white', 'groceri', 'better', 'wond
er', 'caus', 'queri', 'benefit', 'correctli', 'go', 'hook', 'ring2', 'dinner', 'compar', 'readi', 'even', 'search',
'wend', 'woodland', 'alon', 'messag', 'sure', 'nog', 'drawback', 'condo', 'add', 'choic', 'helper', '123', 'greates
t', 'alreadi', 'person', 'never', 'secretari', 'gotten', 'savvi', 'enabl', 'question', 'start', 'own', 'bluetooth',
'cheaper', 'oper', 'almost', 'thunder', 'that', 'swore', 'endless', 'request', 'new', 'demand', 'final', '8', 'brothe
r', 'evalu', 'understand', 'slip', 'seem', 'user', '50', 'access', 'usag', 'simplest', 'must', 'function', 'conveni',
'part', 'smart', 'faster', 'mic', 'walk', 'got', 'caution', 'catch', 'rare', 'hand', 'membership', 'glad', 'follow',
'centuri', 'highli', 'everi', 'forecast', 'roku', 'rain', 'though', 'chair', 'schedul', 'color', 'placement', 'clearl
i', 'cold', 'cautiou', 'ok', 'audiophil', 'someth', 'gener', 'dog', 'le', 'surround', 'dim', 'one', 'divers', 'goog
```

코드 설명

- mat 라는 리스트를 만들어서 filter_words 에 있는 단어들읳 중복을 제거하였습니다.
matrix를 만들기 위해 만들었습니다. (행과 열 이름에 넣기 위해서 만들었습니다.)
- 이를 이용하여 D-T matrix를 만들고, T-D matrix를 만들 것입니다.
- 이후에 T-T matrix를 만들어서 co-occurrence analysis를 수행하겠습니다.
- 결과를 함께 첨부하였습니다.

■ D-T, T-D matrix로 T-T matrix 만들고, csv 파일로 저장

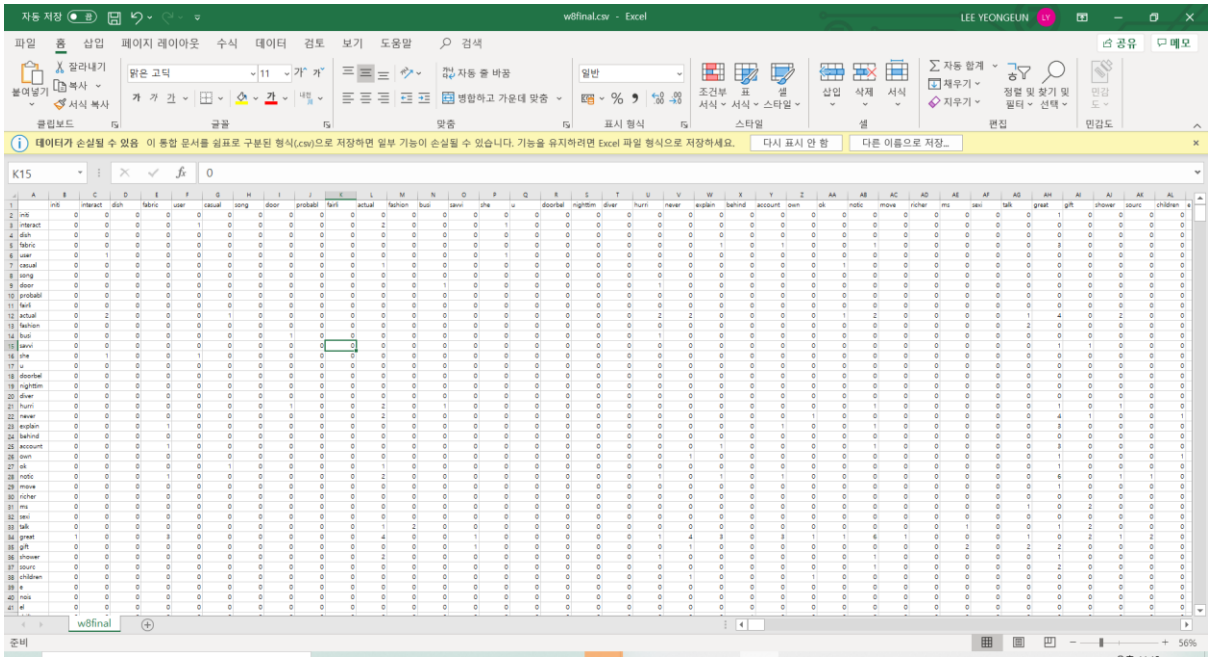
실행 화면

In [13]:
```
index_list = []
for i in range(1, 101):
    index_list.append(i)

df = pd.DataFrame(columns = mat, index = index_list)

def counter(input_list):
    word_count = {}
    for word in input_list:
        if word in word_count:
            word_count[word] +=1
        else:
            word_count[word] = 1
    return word_count


for i in range(0, len(filter_words)):
    word_count = counter(filter_words[i])
    for j in range(0, len(word_count)):
        df[list(word_count.keys())[j]][i+1] = int(word_count[list(word_count.keys())[j]])

df = df.fillna(int(0))
#df.to_csv("file.csv", mode='w')

import numpy as np
word_mat = np.dot(df.T, df)

word_df = pd.DataFrame(data = word_mat, columns = mat, index = mat)
for i in mat:
    word_df[i][i] = 0
word_df
word_df.to_csv("w8final.csv", mode = "w")
```

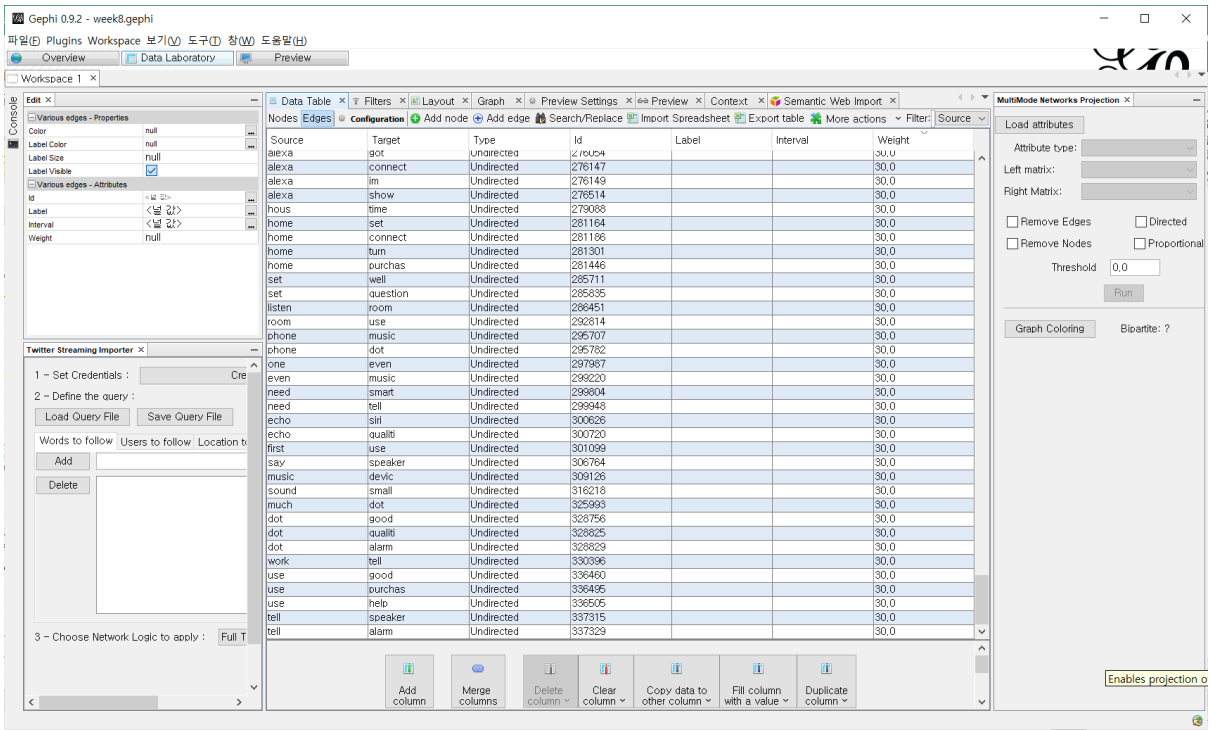
코드 설명

- index_list에 는 1부터 100까지 입력하였습니다. 이는 제공받은 json파일의 review가 100개 이기 때문에, Document number와 같은 역할을 합니다.
- dataframe을 정의하였습니다. column name은 term(mat)으로, index name은 1부터 100까지로 설정하였습니다.
- counter 함수를 정의 하였습니다.
- 이는 위에서 정의한 filter_words에 각 number별로 리스트를 돌면서 있는 단어의 개수를 matrix에 저장합니다.
- 칸이 채워지지 않은 matrix의 값은 한 번도 단어가 나타나지 않았으므로 0으로 채웠습니다.
- numpy를 이용하여 D-T행렬과 이의 전치행렬인 T-D 행렬을 구하고, 이 행렬들의 곱으로 T-T 행렬을 구하였습니다.
- word_df중에서 대각선에 있는, 즉 행과 열이름으로 같은 term을 가지는 값들을 0으로 바꾸었습니다.
- 이를 gephi에서 사용하기 위해서 csv 파일로 저장했습니다.

■ gephi를 통한 network 시각화



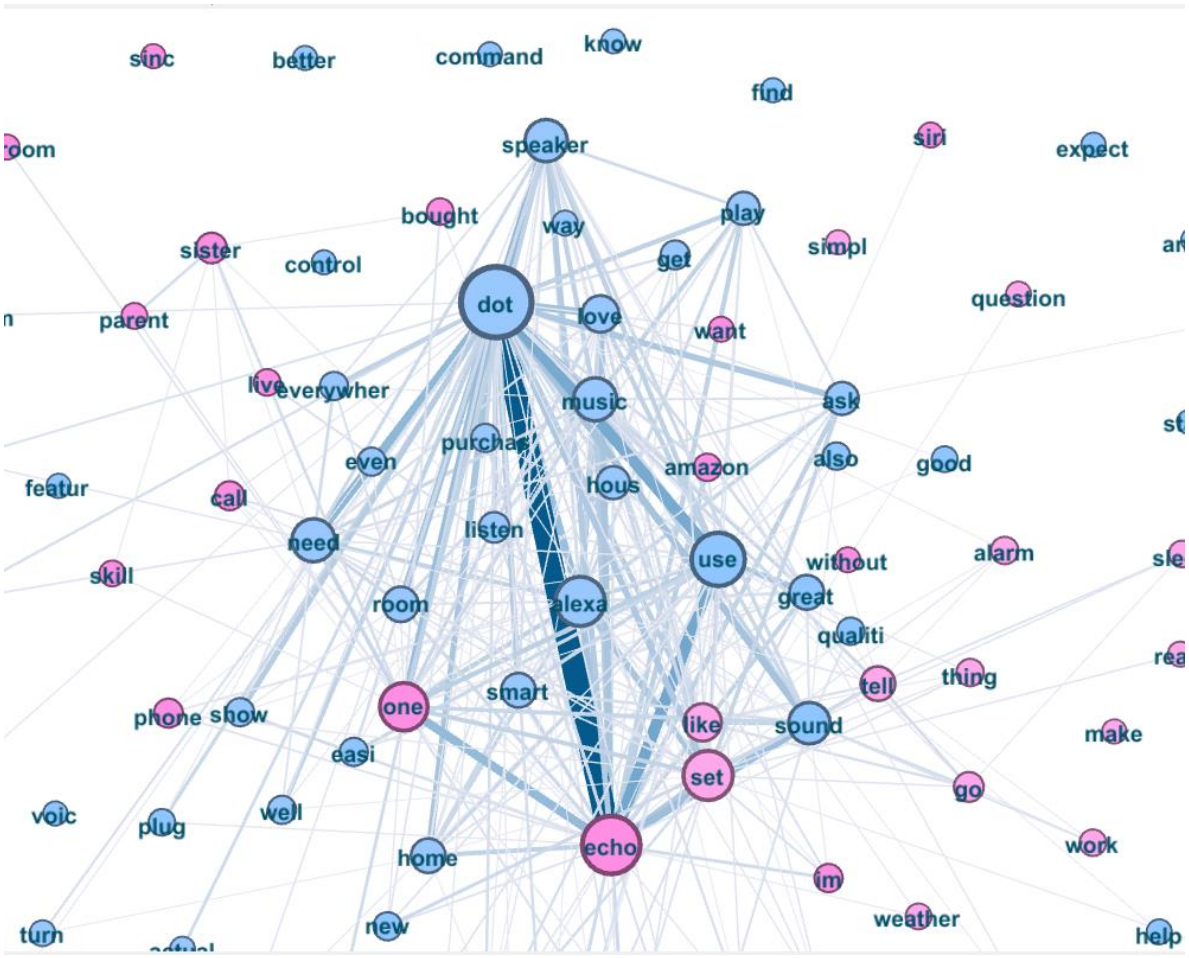
- 저장된 csv파일은 위에 첨부한 형태와 같습니다.



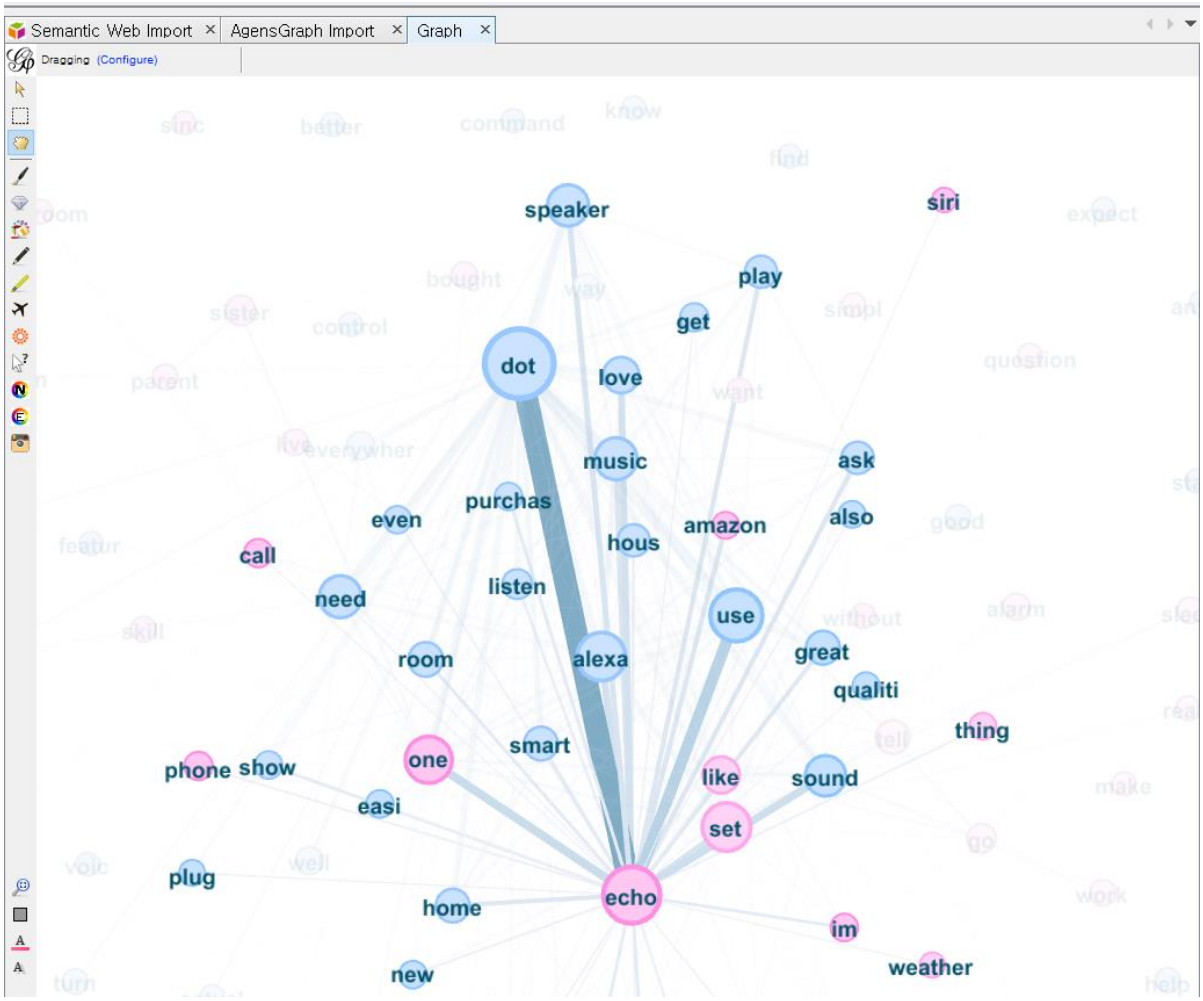
- gephi에서 import spreadsheet를 통해 불러왔습니다.
- 이후, undirected로 설정하였고, weight를 30미만으로 갖는 Edges들은 모두 제거하였습니

다.

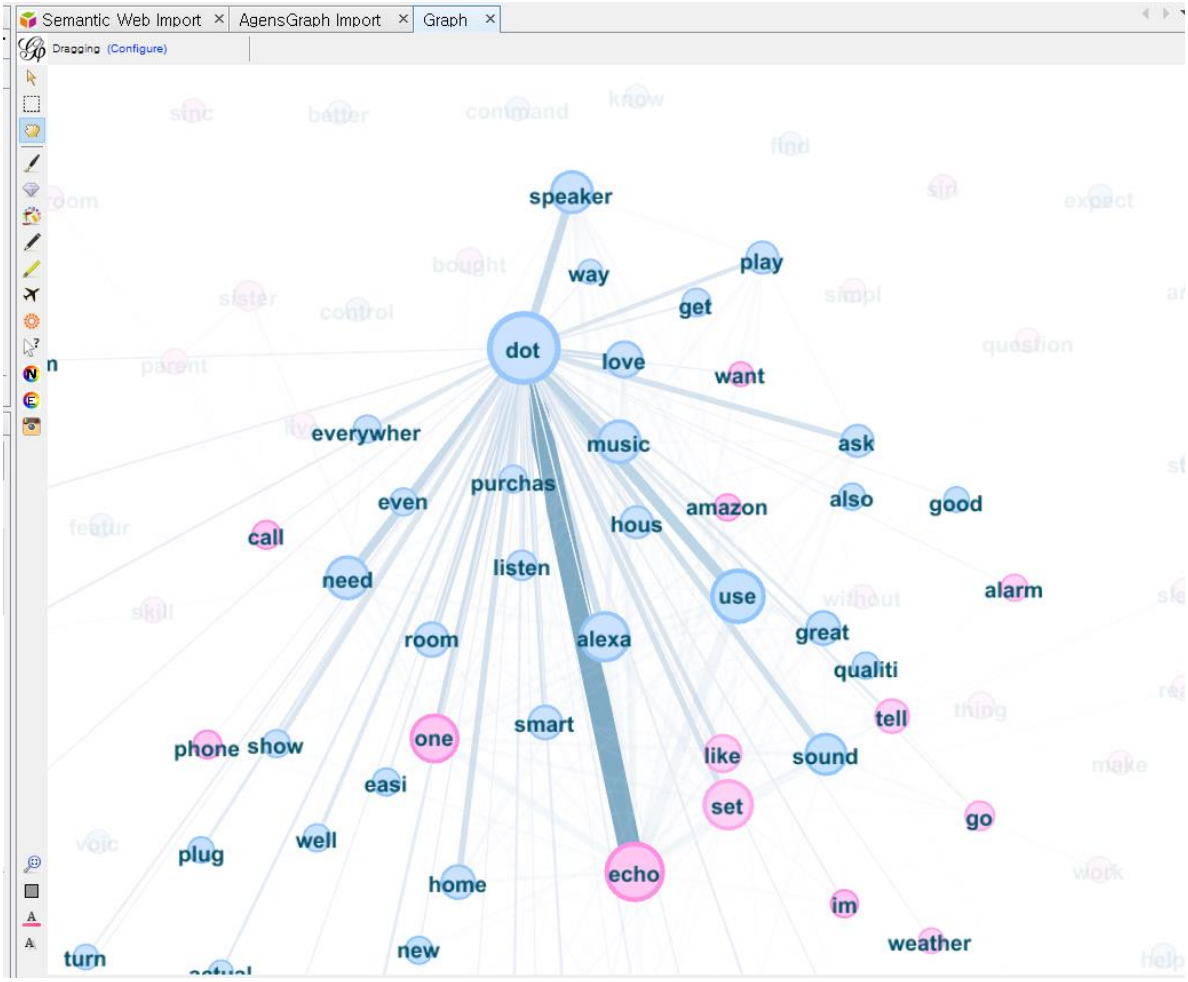
- centrality analysis 지표로 pagerank를 사용하기 위해서, gephi에서 pagerink에 따라서 edge의 굵기를 다르게 하였고(최소와 최대 설정), 이를 시각화하여 중심 부분 (굵은 edge를 갖는 부분)을 확대하였습니다.
- weight를 30이상 갖는 edges와 nodes들만 시각화 하였음에도, 많은 edge들이 존재하였고, 이중 중심을 갖는(많이 연결되어있는) 단어들 위주로 분석하였습니다.



- network의 중심입니다. dot과 echo가 co-occurrence를 높게 가지고 있음을 알 수 있고, 이 두개의 키워드가 center에 있음을 알 수 있습니다.
- 근접한 node들끼리, 같은 색상으로 설정하였습니다.



- echo의 분포는 set, one, like, phone, thing, call, amazon 등과 동시출현이 많이 되었고(근접하고) 그 외에도 dot을 비롯한 다른 키워드들과 동시출현 네트워크를 맺고 있습니다.



- dot의 경우에도 music, alexa, use, sound, need 등과 동시출현 네트워크를 갖고 있음을 알 수 있습니다.