Individual Assignment 1

IPC 동시 출현 네트워크 분석을 통한

특허 제목 분석

담당교수님: 윤장혁 교수님

과목명: Data Analytics

이름: 박민성

학번: 201611145

전공: 산업공학과

제출일: 2020.05.15

1. 주제선정

주제 선정 초기

주제를 선정하기에 앞서 이 특허 데이터들이 무엇을 의미하는지 조사를 면밀히 하였다. 후방인용, 전방인용 등이 무엇인지, IPC의 계층구조가 어떻게 이루어졌는지 등을 검색과 개인과제 PPT 등을 정독을 통해서 알아보았다. 또한, 어떤 방법들을 지금까지 내가 학습했는지 과연 이러한 방법들을 특허데이터에 쓸 수 있을지 고민을 많이 했다.

주제선정과정과 이유

가장 관심이 가는 항목은 바로 인용의 부분이었다. 처음에는 전방인용에 대해서 관심이 갔다. 전방인용 분석을 통해서 앞으로의 기술 동향 같은 것들에 대해서 분석을 해보고 싶었지만, 내가 가진 데이터와는 적합하지 않다고 생각했다.

또한, 초록의 문자열 데이터의 유사도가 IPC와 관련이 있을지 궁금했다. 문장과 문장 사이의 유사도를 구하고 유사하다면 IPC의 첫 글자, 즉, 섹션 간의 유의함이 있을까? 라는 의문을 가졌다. 하지만 이러한 데이터는 의미가 없다고 생각했다. 차라리 IPC데이터를 이용해서 Network 모델을 구상하고 그 것을 토대로 분석을 시작하기로 했다.

어떻게 하면 데이터를 선별해서 쓸 수 있을까? 모든 자료를 계산해서 정제하는 방법도 생각해봤지만 시간만 걸릴 뿐 의미가 없다고 생각했다.

고민하고 자료를 찾아보면 와중에 '어떤 특허자료'가 가치가 있는지에 대해서 생각해보았다. 고민도 해보고 여러 자료를 찾아본 결과 특허 기술의 지표는 여러가지 종류가 있고 분석에 따라 다양한 지표와 방법을 쓰는 것을 알게 되었다. 특정지표가 아닌 내 데이터가 어떤 데이터인지에 대해서 파악해 보기로 했다. 내가 가진 특허자료는 2017년도 한 해에 미국에 등록된 특허 자료이다. 우리나라 특허청의 자료에 의하면 미국은 특허의 단계가 다른 나라에 비해 성숙한 단계이고 기초기술에 집중을 많이 하고 있다고 한다. 이 말은 즉, 다른 나라에 비해 특허하나 당 후방인용의 수가 높고, 후방인용한 특허들의 특허등록연도의 평균 기간이 응용 기술에 비해 기간이 크다는 것이다. 따라서 후방인용수가 평균보다 많고, 후방인용특허들의 특허등록연도를 넉넉히 잡기로 했다. 이렇게 기준을 설정하면 가장 미국의 특허정보를 잘 활용 할 수 있을 것 같았다.

2017년도에 특허 데이터상에서 중심이 되는 섹션을 찾아보고, 섹션 사이의 관계, 더 나아가 어떤 섹션이 기초가 되는 분야이고 응용되는 분야인지 알아보고 섹션 사이의 Keyword가 되는 것은 단어가 무엇인지 알아보기로 하였다.

따라서, 나는 후방인용수가 많은 자료와 기술 순환 주기가 비교적 긴 자료에 대해서 일차적으로 자료를 선별해 1차로 정제된 데이터를 얻은 후, IPC 분석과 특허 제목 분석을 통해 트랜디한 키워드를 찾아보기로 했다.

최종 목표 설정

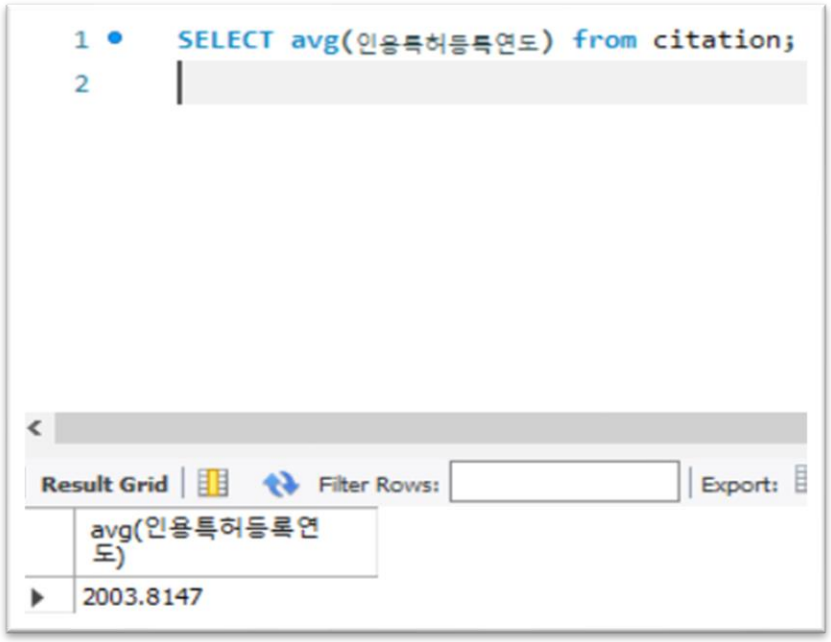
1. 데이터를 SQL을 통해서 1차적으로 정제한다.
2. 정제된 2017년 특허데이터 중 가장 기반이 되는 섹션을 파이썬을 활용하여 Network 분석을 통해 무엇인지 파악해본다.
3. 가장 중요한 Node를 알아보고 그 Node와 가장 Network model에서 동시 출현 빈도가 높은 node의 2차적인 IPC class 분석을 통해 가장 핵심적인 class 도출한다.
4. 어떤 섹션이 가장 기초가 되는 섹션인지 파악해본다.
5. 위의 두 섹션에서 나온 가장 동시 출현 빈도가 높은 Class도출 후, 가장 Hot했던 Keyword를 도출한다.

2. 목표에 따른 구체적 방법론과 과정

※ 각 목표가 순차적으로 진행되다 보니 (2. 목표에 따른 구체적 방법론과 과정)에 간단한 결과가 포함되어 있다. 자세한 해석은 (3. 결과 및 결과 해석)에서 다룬다.

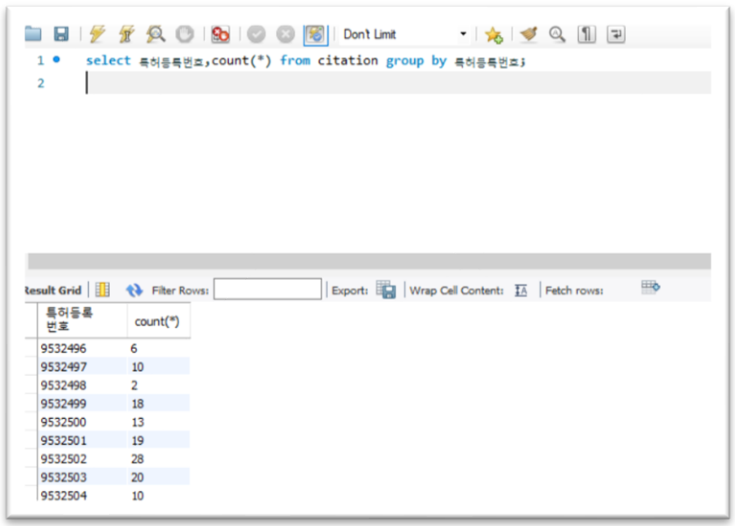
SQL을 통한 데이터 선별

후방 인용수의 평균, 후방 인용수의 특허등록연도의 평균을 계산해서 threshold를 설정해서 일차적으로 자료를 선별한다.



평균 2003.8147

인용 특허 등록연도 threshold = 2004로 설정했다.



	A	B	C
1	count(*)		
2	6		42.27438
3	10		
4	2		

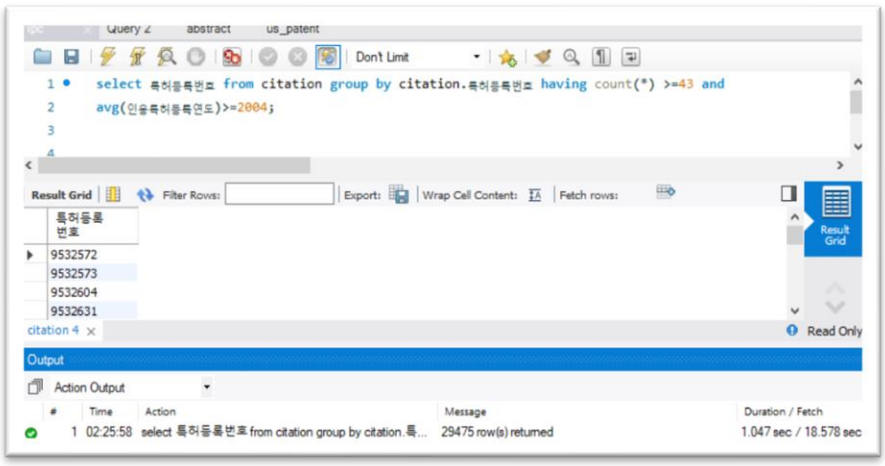
평균 42.27438

특허등록번호 threshold = 43로 설정했다.

✓	2	16:11:57	SELECT * FROM tp_class_da.citation	13260880 row(s) returned	0.000 sec / 34.094 sec
✓	3	16:15:27	SELECT * FROM tp_class_da.abstract	314791 row(s) returned	0.000 sec / 18.718 sec

Citation에 있는 전체 1300만개 자료 중

SQL 구문에 조건에 부합되는 자료를 abstract 테이블 약 31만개중 30000개 정도의 데이터를 활용하려고 한다.



'정제된 특허등록번호.csv'라는 파일로 추출 후 사용한다. 이 파일은 SQL 구문을 통해 걸러진 특허등록번호 들의 csv 파일이다. 특허등록번호가 DB에서 PK로 작용하기 때문에 이 데이터만 있으면 다른 TABLE 데이터와 연결하기가 용이하다.

#정제된 특허등록번호의 첫 행은 '특허등록번호'의 행의 이름이 들어가 있어서 수정을 했다.

파이썬과 Gephi를 통한 네트워크 모델링

정제된 데이터의 특허등록 번호를 PK로 삼고 그에 해당하는 IPC를 매칭시켜서 리스트에 넣는 과정을 파이썬으로 진행한다.

그 후, IPC의 첫번째 글자들만 뽑아서 Set로 만든 후 분석을 진행한다. 분석은 동시 출현 네트워크 기법을 사용한다. 행렬화 한 후 역행렬을 곱해주는 식을 써서 섹션과 섹션관의 관계를 나타내는 한 행렬을 도출한 후 그 행렬을 Gephi를 통해서 시각화 한다.

어떤 식으로 네트워크를 구성할지 설계에 대한 고민이 있었다. 하나의 특허에는 다수의 IPC가 존재할 수 있는데 그 섹션이 중복-될 수도 있고 하나 일 수도 있다. 예를 들어 [A, A, B]혹은 [A, A, B, B]와 같은 경우이다. 이때, [A, B]와 [A, B, B]를 같은 Network로 볼 것인지 아니면 가중치를 둘 것인지에 대해서 고민이었다. 같은 경우로 본다면 ARM분석에서 X가 1개인 경우만을 보면 된다. 하지만 이 모델은 적합하지 않다고 생각했다. 그 이유는 내가 가진 데이터의 행태를 보니 중복이 많은 데이터가 많이 존재했다.

```
39  for i in range(0,len(result)):
40
41      if len(result[i]) == len(set(result[i])):
42          a=a+1
43      else:
44          b=b+1
45  print("a",a)
46  print("b",b)
```

<중복검사.py 실행 결과>

```
a 2478
b 26996
```

result에는 IPC의 첫 글자들이 세트화 되어있다. Ex) [A,B,B]

중복이 없는 자료 2478개 중복이 있는 자료 26996라는 결과가 나왔다.

모든 데이터 계산 과정은 파이썬을 통해 진행했다.

C: > Users > minisong > Desktop > ipc 코딩.py > ...

```python
import csv
import pandas as pd
from tqdm import tqdm
from sklearn.feature_extraction.text import CountVectorizer
import scipy
import nltk
ipc=[]
f = open('./ipc.csv','r',encoding='utf-8')
rdr = csv.reader(f)
for i in rdr:
    ipc.append(i)
rawdata =[]
mid =[]
a=100
정제=[]
f = open('./정제된 특허등록번호.csv','r',encoding='utf-8')
lines=f.readlines()
#rdr = csv.reader(f)
for i in lines:
    정제.append(i.replace("\n",""))
for i in tqdm(range(1,len(ipc))):
    if ipc[i][0] in 정제:

        if ipc[a][0] == ipc[i][0]:
            mid.append(ipc[i][1])
        else:
            rawdata.append(mid)
            mid =[]
            mid.append(ipc[i][0])
            mid.append(ipc[i][1])
            a=i
rawdata.remove([])
new=[]
```

<ipc code_1.py -1>

```
for i in range(0,len(rawdata)):
    sia = ""
    for j in range(1,len(rawdata[i])):
        sia= sia + " " + rawdata[i][j][0]
    new.append(sia)
vocabulary = {}
data = []
row = []
col = []
tokenizer = nltk.tokenize.word_tokenize
def create_cooccurrence_matrix(sentences, window_size=2):
    for sentence in sentences:
        sentence = sentence.strip()
        tokens = [token for token in tokenizer(sentence) if token != u""]
        for pos, token in enumerate(tokens):
            i = vocabulary.setdefault(token, len(vocabulary))
            start = max(0, pos-window_size)
            end = min(len(tokens), pos+window_size+1)
            for pos2 in range(start, end):
                if pos2 == pos:
                    continue
                j = vocabulary.setdefault(tokens[pos2], len(vocabulary))
                data.append(1.)
                row.append(i)
                col.append(j)
    cooccurrence_matrix_sparse = scipy.sparse.coo_matrix((data, (row, col)))
    return vocabulary, cooccurrence_matrix_sparse
sentences = new
vocabs,co_occ = create_cooccurrence_matrix(sentences)
df_co_occ  = pd.DataFrame(co_occ.todense(),
                          index=vocabs.keys(),
                          columns = vocabs.keys())
df_co_occ = df_co_occ.sort_index()[sorted(vocabs.keys())]
df_co_occ.to_csv("Co-occurence.csv", mode='w')
print(df_co_occ)
```

<ipc code_1.py -2>

처음에 정제한 특허등록번호와 전체 ipc파일을 비교해서 특허등록번호와 그 특허등록번호가 가진 IPC를 리스트화하는 작업을 처음에 거친다.

ex) [9532496, A01B-063/00, A01B-071/02]

그후, 다시 IPC의 앞글자들만 추출해서 리스트화 시킨다.

ex) [A, B, B], [A, A, B, B]

그후 계산을 통해 섹션간의 관계를 볼 수 있는 8 X 8 Matrix를 도출한다.

기본적으로 계산에 사용하는 Method는 동시 출현(Co-occurrence Method)이다.

동시출현 Matrix를 구현해서 네트워크를 구성한다.

Co-occurrence.csv라는 행렬 csv 파일 형태로 추출한다.

이 데이터를 Gephi를 통해 시각화 한다.

이후, ipc code_2.py를 통해 IPC의 2,3번째 글자인 클래스를 분석한다.

C: > Users > minisong > Desktop > ipc code_2.py > ...

```python
import csv
import pandas as pd
from tqdm import tqdm
from sklearn.feature_extraction.text import CountVectorizer
import scipy
import nltk
ipc=[]
f = open('./ipc.csv','r',encoding='utf-8')
rdr = csv.reader(f)
for i in rdr:
    ipc.append(i)
rawdata =[]
mid =[]
a=100
정제=[]
f = open('./정제된 특허등록번호.csv','r',encoding='utf-8')
lines=f.readlines()
for i in lines:
    정제.append(i.replace("\n",""))
print(정제)
for i in tqdm(range(1,len(ipc))):
    if ipc[i][0] in 정제:

        if ipc[a][0] == ipc[i][0]:
            mid.append(ipc[i][1])
        else:
            rawdata.append(mid)
            mid =[]
            mid.append(ipc[i][0])
            mid.append(ipc[i][1])
            a=i
rawdata.remove([])
sp=[0,0]
new=[]
for i in tqdm(range(0,len(rawdata))):
    sia = ""
    for j in range(1,len(rawdata[i])):
```

<ipc code_2.py 1>

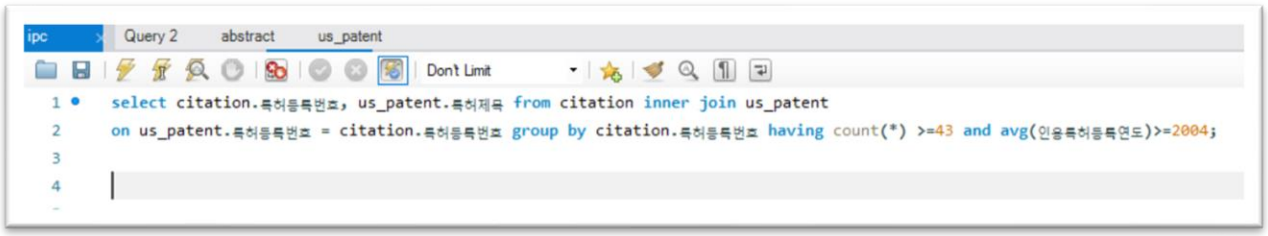
```
            if rawdata[i][j][0] =="H":
                sp[0] = 1
                sia= sia + " " + rawdata[i][j][0:3]
            if rawdata[i][j][0] =="G":
                sp[1] = 1
                sia= sia + " " + rawdata[i][j][0:3]
        if sp==[1,1]:
            new.append(sia)
            sp=[0,0]
    vocabulary = {}
    data = []
    row = []
    col = []
    tokenizer = nltk.tokenize.word_tokenize
    def create_cooccurrence_matrix(sentences, window_size=2):
        for sentence in sentences:
            sentence = sentence.strip()
            tokens = [token for token in tokenizer(sentence) if token != u""]
            for pos, token in enumerate(tokens):
                i = vocabulary.setdefault(token, len(vocabulary))
                start = max(0, pos-window_size)
                end = min(len(tokens), pos+window_size+1)
                for pos2 in range(start, end):
                    if pos2 == pos:
                        continue
                    j = vocabulary.setdefault(tokens[pos2], len(vocabulary))
                    data.append(1.)
                    row.append(i)
                    col.append(j)
        cooccurrence_matrix_sparse = scipy.sparse.coo_matrix((data, (row, col)))
        return vocabulary, cooccurrence_matrix_sparse
    sentences = new
    vocabs,co_occ = create_cooccurrence_matrix(sentences)
    df_co_occ  = pd.DataFrame(co_occ.todense(),
                              index=vocabs.keys(),
                              columns = vocabs.keys())
    df_co_occ = df_co_occ.sort_index()[sorted(vocabs.keys())]
    df_co_occ.to_csv("Co-occurrence HG.csv", mode='w')
```

<ipc code_2.py -2>

기본적으로 code_1 과 code_2는 유사하다. 하지만, 섹션의 구분을 위해 클래스 앞에 섹션까지 붙여서 네트워크 모델을 구성한다. ex: (H01, G06)

특허 제목을 통한 Text 분석

SQL구문을 활용해서 '특허제목추출.csv'를 추출해서 활용한다.



C: > Users > minisong > Desktop > Text code.py > ...

```python
import csv
import pandas as pd
from tqdm import tqdm
from sklearn.feature_extraction.text import CountVectorizer
import scipy
import nltk
pd.options.mode.chained_assignment = None
import numpy as np
np.random.seed(0)
from konlpy.tag import Twitter
twitter = Twitter()
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
import re
from nltk.corpus import stopwords
title=[]
f = open('./특허제목추출.csv','r',encoding='utf-8')
rdr = csv.reader(f)
for i in rdr:
    title.append(i)
title.pop(0)
rawdata=[]
f = open('./정제된 ipc.csv','r',encoding='utf-8')
rdr = csv.reader(f)
for i in rdr:
    rawdata.append(i)
sp=[0,0]
new=[]
for i in tqdm(range(0,len(rawdata))):
    for j in range(1,len(rawdata[i])):
        if rawdata[i][j][0:3] =="H04":
            sp[0] = 1
        if rawdata[i][j][0:3] =="G06":
            sp[1] = 1
    if sp==[1,1]:
        new.append(title[i][1])
        sp=[0,0]
    else:
```

<text code.py -1>

```
C: > Users > minisong > Desktop > Text code.py > ...
39          sp=[0,0]
40      def tokenizer(raw, pos=["Noun","Alpha","Number"], stopword=[]):
41          return [
42              word for word, tag in twitter.pos(
43                  raw,
44                  norm=True,
45                  stem=True
46                  )
47                  if len(word) > 1 and tag in pos and word not in stopword
48              ]
49      stop_words = set(stopwords.words('english'))
50      aaa = []
51      listlsit=""
52      rawdata = []
53      for i in new:
54          aa=[]
55          a=[]
56          a= re.sub('[^a-zA-Z]', ' ', i)
57          a.lower()
58          middle=""
59          aa=a.split(" ")
60          for w in aa:
61              if w  not in stop_words:
62                  middle = middle + w + " "
63          rawdata.append(middle)
64      vectorize = CountVectorizer(
65          tokenizer=tokenizer,
66          min_df=50
67      )
68      X = vectorize.fit_transform(rawdata)
69      features = vectorize.get_feature_names()
70      featuress = {}
71      a = np.array(X.toarray())
72      df = pd.DataFrame(np.dot(a.T,a))
73      df.to_csv("Text_code_result.csv",encoding="utf-8-sig",index = features,header = features)
```

<text code.py -2>

명사를 위주로 단어를 추출해서 분석한다. min_df는 50으로 설정해서 빈도수가 50미만인 단어는 제외시켰다. '정제된 ipc.csv'파일을 사용했다. IPC 분석할 때마다 코드를 진행하니 항상 모든 데이터에서 SQL 정제된 자료에 맞는 IPC를 추가하는 과정이 15분정도 걸려서 csv로 추출한 뒤 사용했다. **기본적인 빈도에 의거한 text 분석**을 하였다.

H04와 G04가 둘다 포함된 IPC를 가지고 있는 특허자료들에 대해서 TEXT 분석을 진행했다.

IPC의 앞글자 3개를 비교하여서 자료를 걸러낸 후, 그 특허등록번호들이 가진 특허의 이름을 추출했다. 그 후 명사를 추출하기 위해 불용처리를 해주었고 min 빈도는 50으로 설정했다.

가장 중요한 섹션 – 섹션관계에서 가장 많이 등장하는 클래스-클래스 관계를 파악한후, 분석을 진행했다.

3. 결과 및 결과 해석

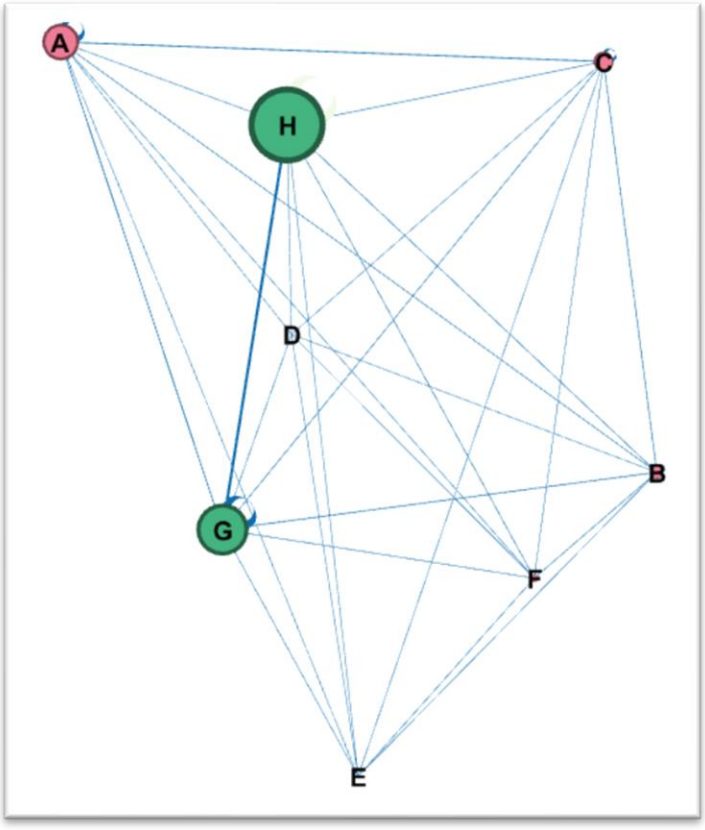
Network 시각화 및 분석

ipc code_1.py의 결과

ipc code_1.py 결과는 특허 섹션(A-H)간의 관계를 보여준다.

	A	B	C	D	E	F	G	H
A	81504.0	1118.0	3809.0	22.0	37.0	212.0	2886.0	505.0
B	1118.0	26880.0	1612.0	78.0	256.0	874.0	2189.0	990.0
C	3809.0	1612.0	43132.0	100.0	151.0	241.0	1133.0	1428.0
D	22.0	78.0	100.0	1010.0	2.0	4.0	33.0	19.0
E	37.0	256.0	151.0	2.0	2806.0	174.0	359.0	76.0
F	212.0	874.0	241.0	4.0	174.0	15306.0	912.0	829.0
G	2886.0	2189.0	1133.0	33.0	359.0	912.0	116538.0	15814.0
H	505.0	990.0	1428.0	19.0	76.0	829.0	15814.0	172786.0

A-H의 관계를 보여주는 Matrix Co-occurrence.csv 생성



<Gephi를 이용한 Co-occurrence.csv의 Network>

행렬의 대각원소는 섹션의 전체 출현빈도를 보여준다. 따라서 Node의 크기로 표현했다. 행렬간 빈도는 Edge의 진하기로 표현했다. G-H의 Edge가 가장 강하게 표현된다. 섹션의 개수가 몇 개 없어서 복잡하지 않은 네트워크가 형성되었다.

H가 가장 중요한 Node이고 G와 H가 가장 동시 출현 빈도가 높으므로 G와 H가 제일 긴밀하게 연결되어 있다고 할 수 있다.

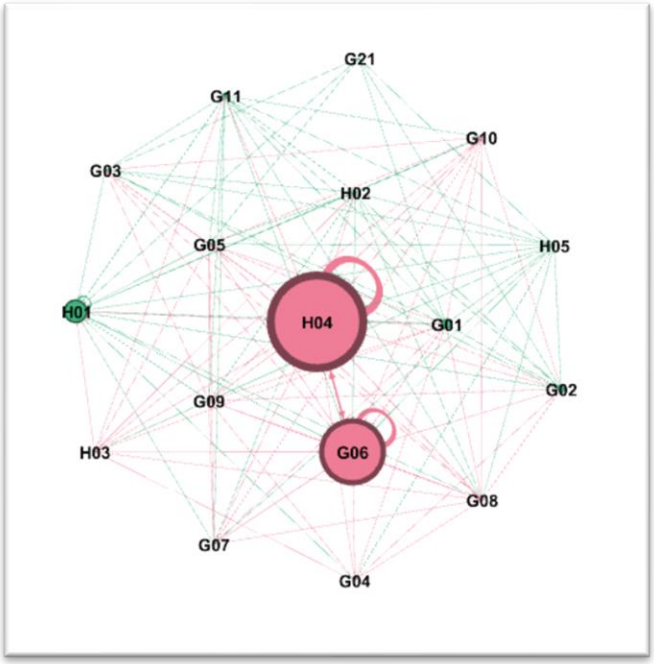
2017년 특허자료 분석 결과 H(전기) 섹션이 가장 핵심적인 Node인데 그것과 가장 밀접한 섹션이 G이다. 전기의 가장 기초적인 요소가 바로 물리이다. 이 결과를 토대로 2017년 미국 특허 시장에서 물리학이 가장 기초가 된다고 할 수 있다. 이 결과가 물리-전기가 아닌 물리-화학과 같은 결과가 나왔다면 섣불리 결론을 도출하지 못하였을 것이다. 하지만 결과가 물리-전기라는 보편적이고 근거가 있는 결론이어서 이러한 관계를 찾을 수 있었다.

ipc code_2.py의 결과

G-H의 섹션 관계에서 클래스의 분석을 진행해서 Network 시각화를 한다.

	G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G21	H01	H02	H03	H04	H05
G01	3980.0	79.0	20.0	6.0	129.0	549.0	18.0	82.0	9.0	10.0	18.0	3.0	411.0	92.0	34.0	474.0	59.0
G02	79.0	3046.0	188.0	1.0	5.0	289.0	0.0	1.0	112.0	6.0	8.0	1.0	440.0	6.0	0.0	246.0	50.0
G03	20.0	188.0	854.0	0.0	2.0	118.0	0.0	1.0	14.0	3.0	1.0	4.0	135.0	5.0	0.0	190.0	14.0
G04	6.0	1.0	0.0	74.0	0.0	29.0	0.0	1.0	1.0	0.0	0.0	0.0	10.0	0.0	2.0	2.0	3.0
G05	129.0	5.0	2.0	0.0	690.0	425.0	13.0	29.0	2.0	1.0	4.0	1.0	40.0	130.0	12.0	99.0	34.0
G06	549.0	289.0	118.0	29.0	425.0	30046.0	293.0	496.0	402.0	200.0	297.0	0.0	548.0	181.0	269.0	8685.0	165.0
G07	18.0	0.0	0.0	0.0	13.0	293.0	256.0	106.0	3.0	1.0	2.0	0.0	18.0	14.0	0.0	228.0	2.0
G08	82.0	1.0	1.0	1.0	29.0	496.0	106.0	1246.0	34.0	23.0	35.0	0.0	27.0	78.0	2.0	743.0	36.0
G09	9.0	112.0	14.0	1.0	2.0	402.0	3.0	34.0	838.0	16.0	29.0	0.0	200.0	5.0	8.0	413.0	36.0
G10	10.0	6.0	3.0	0.0	1.0	200.0	1.0	23.0	16.0	1500.0	34.0	0.0	20.0	31.0	29.0	472.0	0.0
G11	18.0	8.0	1.0	0.0	4.0	297.0	2.0	35.0	29.0	34.0	2088.0	0.0	564.0	6.0	113.0	388.0	9.0
G21	3.0	1.0	4.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	16.0	35.0	0.0	0.0	0.0	1.0
H01	411.0	440.0	135.0	10.0	40.0	548.0	18.0	27.0	200.0	20.0	564.0	35.0	10490.0	229.0	111.0	227.0	177.0
H02	92.0	6.0	5.0	0.0	130.0	181.0	14.0	78.0	5.0	31.0	6.0	0.0	229.0	1620.0	52.0	205.0	45.0
H03	34.0	0.0	0.0	2.0	12.0	269.0	0.0	2.0	8.0	29.0	113.0	0.0	111.0	52.0	788.0	236.0	3.0
H04	474.0	246.0	190.0	2.0	99.0	8685.0	228.0	743.0	413.0	472.0	388.0	0.0	227.0	205.0	236.0	45244.0	167.0
H05	59.0	50.0	14.0	3.0	34.0	165.0	2.0	36.0	36.0	0.0	9.0	1.0	177.0	45.0	3.0	167.0	1046.0

섹션-클래스 간의 관계를 보여주는 Matrix Co-occurrence_HG.csv 생성



<Gephi를 통한 Co-occurrence_HG.csv의 Network>

IPC 섹션과 클래스

H 섹션: 전기

H01: 기본 전기 요소
H02: 전력의 생성, 변환 또는 분배
H03:기본 전자 회로
H04: 전기 통신 기술
H05: 다른 분야에는 없는 고유한 전기 기술
G 섹션 : 물리
G01: 측정, 테스트
G02: 광학
G03: 사진, 시네마토그래피, 광파 이외의 파장을 이용한 유사기술, 전자 그래피, 홀로그래피
G04: 측시학
G05: 제어, 조절
G06: 컴퓨팅, 계산 또는 카운팅(counting or counting, accounting)
G07: 검사-장치
G08: 신호 전달
G09: 교육, 암호, 디스플레이, 광고, 씰
G10: 악기; 음향학
G11: 정보 저장
G21: 핵물리학; 핵공학

앞선 결과보다 훨씬 복잡한 네트워크가 구성되었다. 그 중에 압도적으로 출현빈도가 큰 node 가 두개 등장했고, 둘의 동시 출현 빈도가 가장 크다.(H04,G06)

결과 Matrix 를 통해 분석한 결과 H04(전기통신기술)과 G06(계산)의 동시 출현빈도가 컸고, G06, H04 의 Node 하나하나의 출현빈도 또한 컸다. 가장 Network 에서 중심이 되는 관계이고 Node 들이라고 판단했다.

계산 또는 컴퓨팅은 전기통신기술에 근간이 되므로 G 섹션이 H 섹션의 기초가 된다는 주장을 어느정도 뒷받침한다.

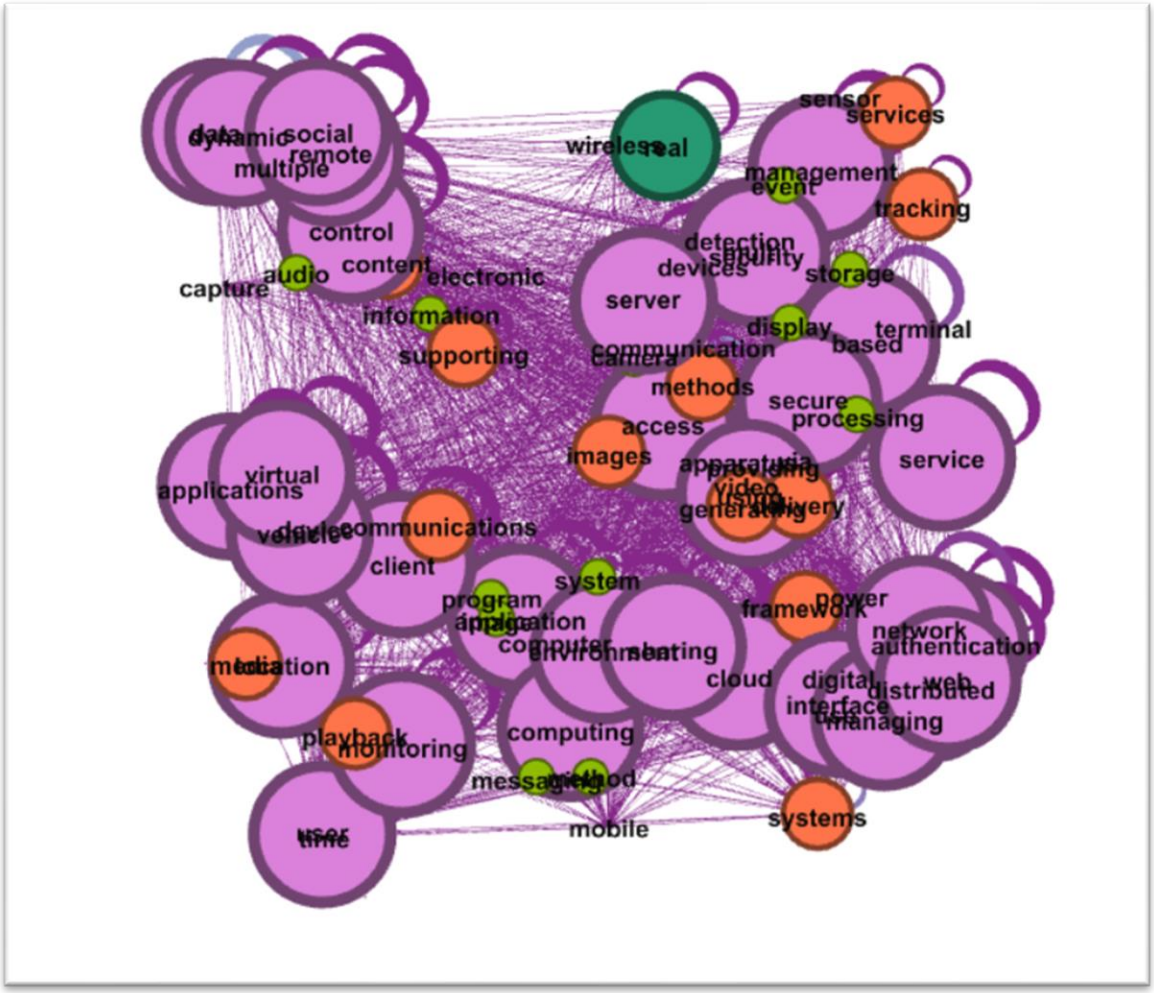
H04 와 G06 가 들어가는 특허 중 특허 Title Text 분석

Text code.py의 결과

빈도가 50이상인 단어들간의 관계를 보여주는 Text_code_result.csv를 생성한다.

	A	B	C	D	E	F	G	H	I	J	K
1		access	apparatus	applicatior	applicatior	audio	authentica	based	camera	capture	client
2	access	165	5	15	6	0	3	16	1	0	4
3	apparatus	5	448	11	5	7	2	25	6	4	2
4	applicatior	15	11	168	3	1	2	20	0	0	13
5	applicatior	6	5	3	74	0	1	10	0	0	2
6	audio	0	7	1	0	61	0	5	1	1	2
7	authentica	3	2	2	1	0	105	19	0	0	1
8	based	16	25	20	10	5	19	464	6	1	16
9	camera	1	6	0	0	1	0	6	69	0	0
10	capture	0	4	0	0	1	0	1	0	56	0
11	client	4	2	13	2	2	1	16	0	0	82
12	cloud	12	2	6	4	0	2	39	0	0	1
13	communic	10	25	4	5	4	1	21	2	3	4
14	communic	1	4	2	0	0	3	9	0	0	2
15	computer	7	32	7	5	4	1	13	1	0	2
16	computinc	7	6	6	2	0	4	12	1	0	2
17	content	12	56	9	2	4	3	56	1	1	5
18	control	35	17	3	3	1	0	10	2	0	6
19	data	32	49	9	3	3	5	49	8	23	9
20	delivery	0	8	4	1	2	0	8	0	0	2
21	detection	0	4	0	3	0	1	18	7	16	1

Index가 숫자로 나와서 csv파일을 변경 후 Gephi를 이용했다. (A열)



<Gephi를 통한 Text_code_result.csv의 Network>

결과를 보니까 Systems, service 같은 의미가 없는 단어가 있다고 생각했다. Stop_word를 추가해서 다시한번 불용처리를 진행했다. 또한 코드에 nltk.WordNetLemmatizer()를 추가해서 복수명사와 단수 명사를 통일 해주기로 하였다.

Min_df 를 5로 낮춰서 진행해서 더 큰 네트워크를 구성하기로 했다.

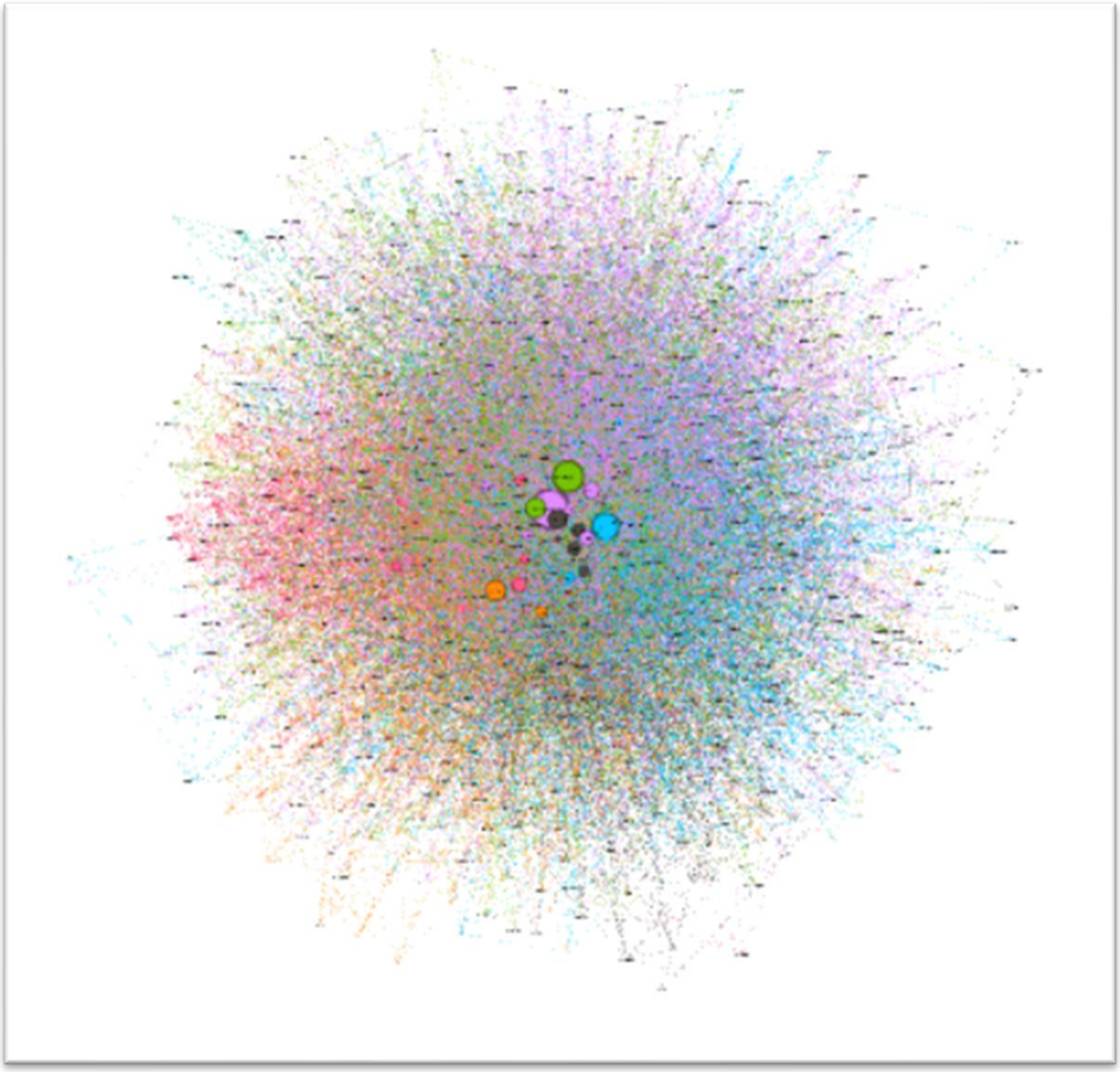
Text code.py를 약간 고쳐 Text code_2.py를 활용했다.

50이라는 높은 빈도에서 네트워크를 구성한다는 것보다 낮춰서 적게 나타나는 단어에서의 네트워크를 다시 구성했다.

Text_code_result_2.csv 생성

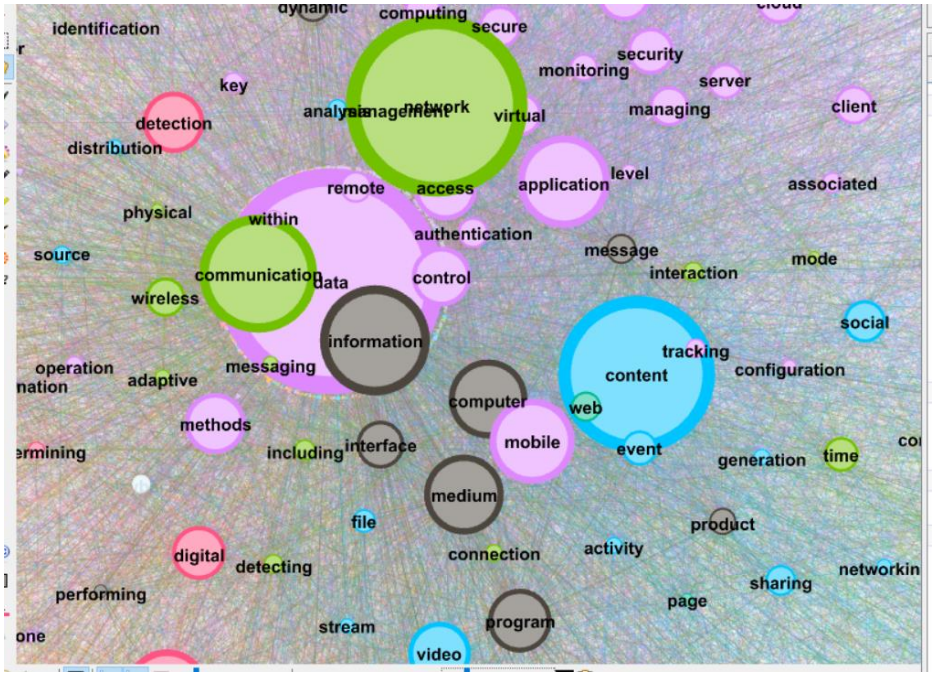
Index가 숫자로 나와서 임의로 변경하여서 Gephi를 이용했다. (A열)

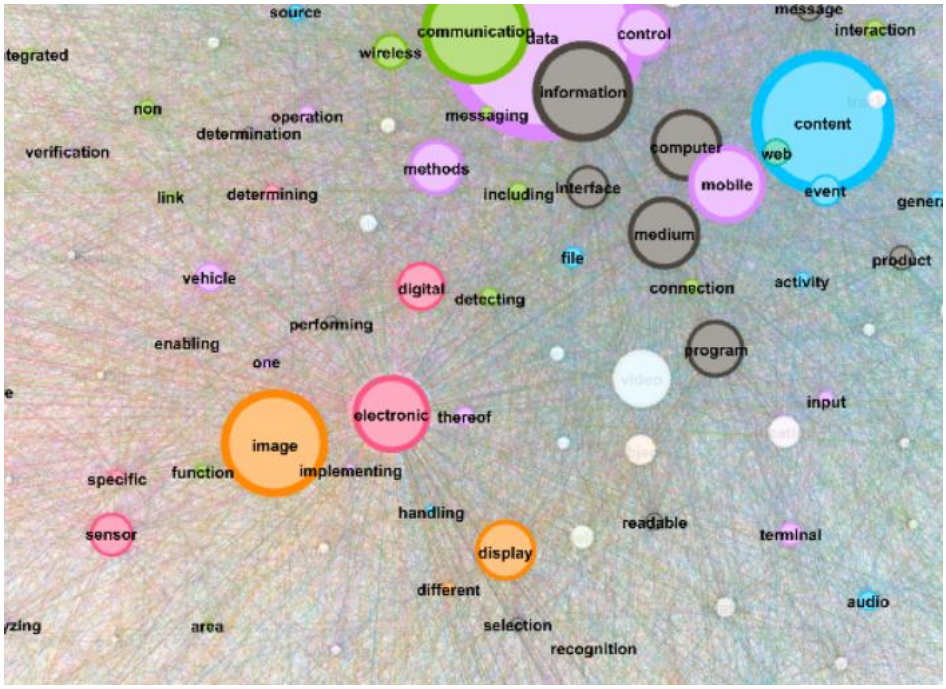
이전의 결과 보다 훨씬 많은 feature들이 나와서 훨씬 더 복잡한 Network가 구성되었다.

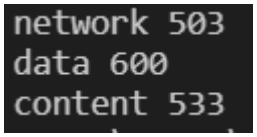


<결과.gephi>

빈도의 최소값을 5로 설정함으로써 아주 큰 Network가 형성된다. 이중에 중심에 있는 NODE들을 분석한다.







(분석결과 data, content, network) 순으로 Text 출현 빈도결과가 나왔다.

Network, mobile, information, application, image, display, electronic 등 많은 키워드들이 분포하고 군집을 이루고 있었다. 가장 중심에 있는 Node는 'data'이다. 'data'는 전체적인 네트워크 모델크기에서도 어디 있는지 보일 정도로 Node의 크기가 크고 중심에 있다. 따라서 H04, G06사이에 가장 중심에 있는 특허 제목 키워드는 'data'이다.

4. 결론 및 느낀점

결론 및 한계

2017년 미국 특허 자료 중 좋은 특허자료라고 생각되는 자료 중에 가장 중심이 되는 섹션은 H(전기)이다. 또한, H와 가장 동시 출현 빈도가 높은 섹션은 G(물리)이다. 이 결과를 토대로 2017년 미국 특허의 가장 기초가 되는 부문은 물리라고 할 수 있다.

H-G 사이의 관계에서 동시 출현 빈도가 가장 높은 클래스는 H04(전기 통신 기술), G06(계산 및 컴퓨팅) 클래스이다. IPC가 G06, H04를 포함하는 특허 자료들을 찾아서 특허의 title을 추출한 후 Title을 Text 분석한 결과 가장 Node의 중심에 있는 Keyword는 "data"라는 결과를 도출했다.

하지만 'data'라는 것은 모호하다. 어떤 'data'인지 알기 힘들다. 따라서 Text Set를 구성할 때 한 단어가 아닌 두 단어, 세 단어를 구성해서 분석을 하면 더 정교한 결과를 얻을 수 있을 것이라고 생각했다. 'data'뿐만 아니라 가장 text 출현 빈도가 높은 세 단어 ['data, content, network']는 모호하다. 구체적으로 어떤 data인지 어떤 network인지 알 수 없다. 처음부터 text set을 2,3개로 설정하는 것도 하나의 방법이겠지만 우선 나와 같은 분석을 진행한후, data나 network와 같은 특정 단어에만 집중해서 분석을 진행하는 것도 괜찮은 방법이라고 생각한다.

오히려 이렇게 모호한 단어여서 출현의 빈도가 높았을 수도 있다고 생각한다. 이 또한 역시 불용처리를 정교하게 하거나 wort set을 잘 구성한다면 해결할 수 있을 것 같다.

H(전기) – G(물리)가 아닌 다른 섹션이었으면 어떠한 섹션이 기초가 되는 부분인지 섣불리 결론을 도출할 수 없었을 것이다. 예를 들어 물리-화학과 같은 부분의 선, 후 관계는 내가 진행한 분석 방법으로는 한계가 분명히 존재한다. 더 정교한 분석 방법이 있다면 물리-화학 같은 애매한 선후 관계도 어떤 것이 기초가 되는지 찾을 수 있을 것이다.

"data"와 지금은 link가 끊어져 있지만 가장 이어질 수 있는 단어가 무엇인지 Link Prediction을 통해 예측할 수 있다는 점을 발견했다. Link Prediction이라는 방향으로 갔다면 가장 유망한 Keyword을 도출하고 그에 따라서 앞으로 연결될 수 있는 word에 대해 도출하는 방법을 택했을 것 같다.

많은 데이터속에서 예측이라는 결과를 도출하는 것은 현대사회에 필요한 부분이고 값진 결과라고 생각한다. 나의 프로젝트 방향은 단순히 과거의 자료를 보고 분석을 해서 2018년을 보는데 도움이 되는 자료라고 생각한다. 하지만, Link Prediction이라는 방법론을 택했다면 단순히 도움을 주는 것이 아니라 명확한 지표를 줄 수 있을 것 같다고 생각한다.

느낀점 및 프로젝트 소감

가장 큰 소감은 마지막 결과가 전기통신기술(H04)와 계산 및 컴퓨팅(G06)과 'data'와 관련이 있다고 생각해서 충분히 생각해서 나름대로 뿌듯했다. 관련이 있다는 것은 어떻게 보면 당연한 결과이지만, 프로젝트가 올바른 방향으로 나갔다는 것에 안도하고 뿌듯했다.

왜 데이터 정제과정과 단어의 불용처리에 대해서 중요하게 다루는지 알게 되었다. 데이터가 많은 경우 For 문 1개와 For 문 2개의 차이가 심각하게 컸다. 데이터를 어떻게 정제해서 사용할 것인지 어떻게 코딩을 하는지에 따라 데이터 처리속도의 차이가 엄청 나다는 것을 알았다.

학교에서 배우는 것은 나와는 좀 거리가 멀고 세상과는 약간 동떨어진 느낌이었는데 이번 프로젝트를 진행하면서 'DA라는 과목은 나와는 동떨어져 있고 관련 없는 과목은 아니다'라는 생각이 들었다.

실제로 존재하는 데이터들을 다루고 정제하는 과정이 생각보다 즐거웠다고 생각한다. 처음에는 데이터 분석이라는 것이 너무 어렵고 보이지도 않는 무형의 과정이라고 생각했는데 이번 분석을 진행하고 또 시각화를 통해서 한 눈에 데이터 분석 결과를 보니 전과는 생각이 많이 달라진 것 같다.

DA라는 과목을 처음 들었을 때는 코딩하는 능력이 가장 요구되는 과목이라고 생각했는데 지금은 데이터 분석을 어떻게 설계하고 진행할 것인지 구상하는 것이 가장 큰 능력이라고 생각한다. 코딩은 부수적인 수단이고 가장 요구되는 능력은 내가 가진 자료에서 어떤 것을 도출할지, 어떤 방법론을 사용할 것인지 또는 통계적으로 이 결과가 유의 한지 이러한 것들을 잘 알고 설계하는 능력인 것 같다.

5. 참고 문헌

통계청 - 통계로 보는 특허동향(2017).pdf

한국 지식 재단 정부 - R&D 특허기술동향조사사업 기본 분석 실무 자료집.pdf

WIPO- IPC PUBLICION

https://www.wipo.int/classifications/ipc/ipcpub/?notion=scheme&version=20200101&symbol=D&menulang=en&lang=en&viewmode=f&fipcpc=no&showdeleted=yes&indexes=no&headings=yes&notes=yes&direction=o2n&initial=A&cwid=none&tree=no&searchmode=smart