

정보문화학 여름 워크샵

파이썬을 활용한 데이터 과학 맛보기

이태영

서울대학교 융합과학기술대학원
인간중심컴퓨팅 연구실

강사소개

이태영

- 개발자/연구자
 - 서울대학교 융합과학기술대학원 석사과정 재학 중
 - 컴퓨터공학부 졸업
 - 인턴 @Nexon, Netmarble
- 연구 관심분야
 - Deep Learning / Natural Language Generation
 - Big Data Stream Processing & Visualization
- 교내 앱 개발 동아리 Appy Road 프로그래밍 강의(2~4기)

워크샵 프로그램 안내

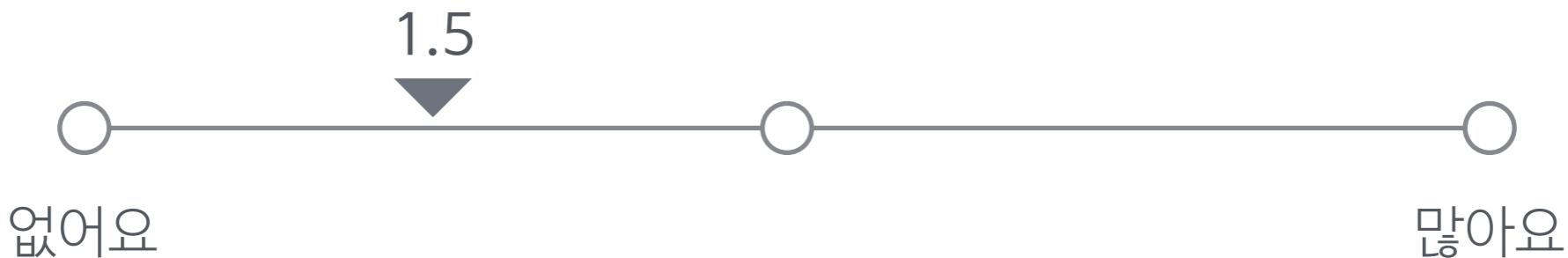
워크샵 프로그램 안내 를 하기 전에...

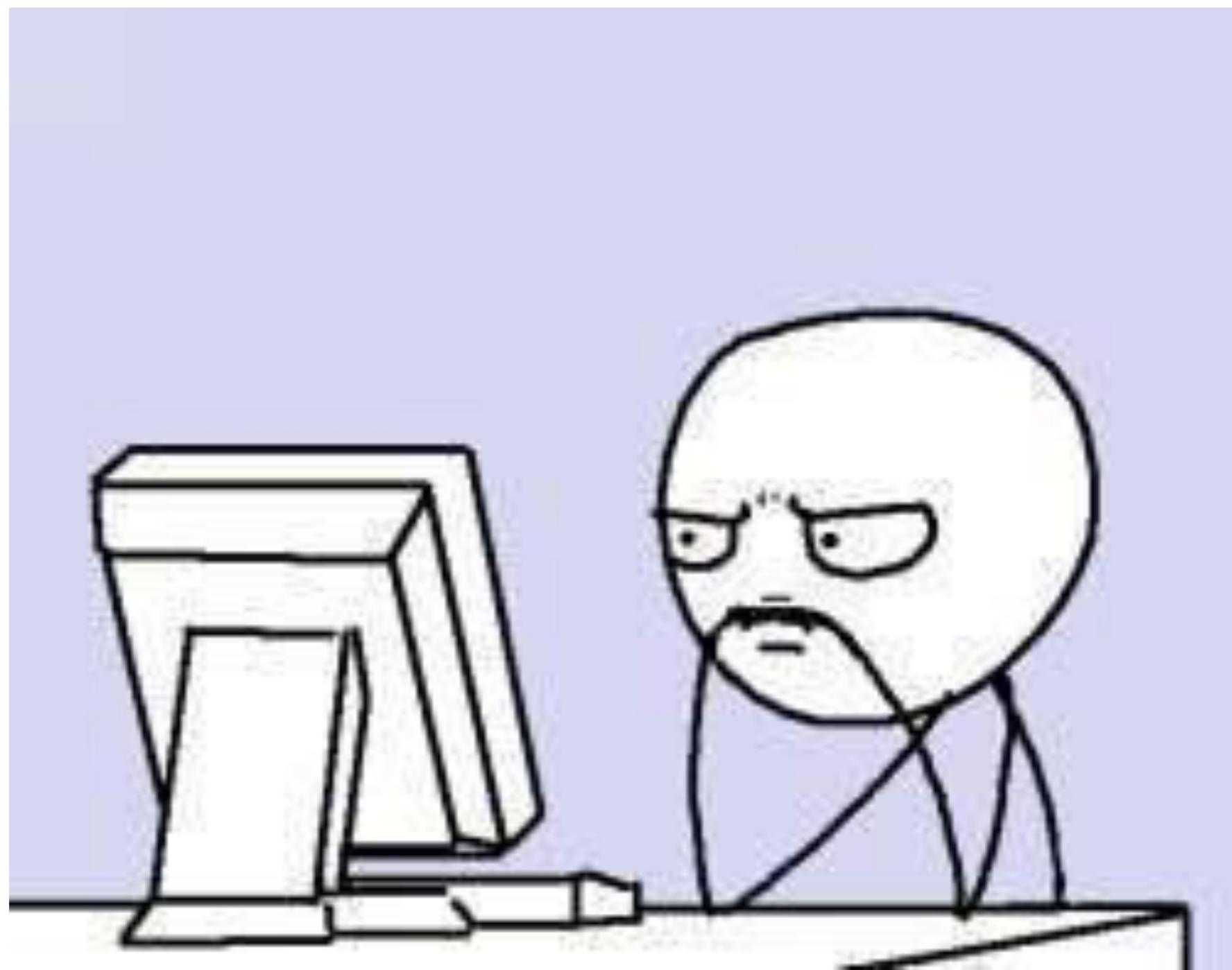
참여자 사전조사

- “파이썬 프로그래밍 경험이 있나요?”

참여자 사전조사

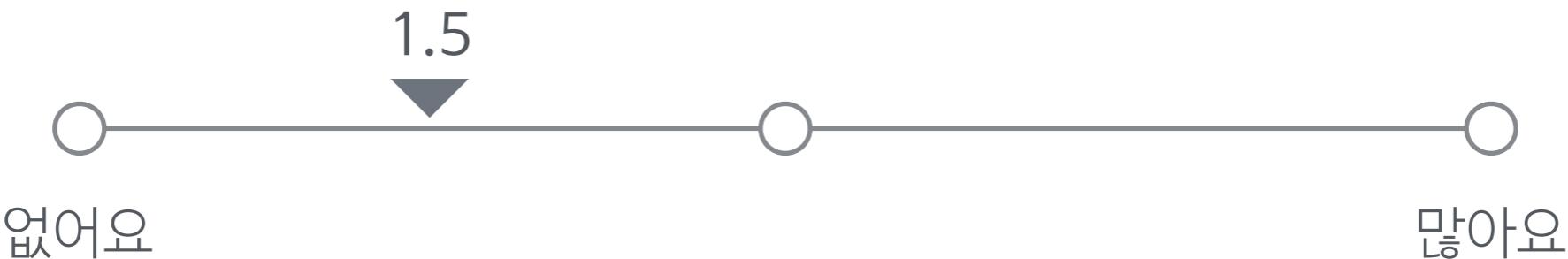
- “파이썬 프로그래밍 경험이 있나요?”





참여자 사전조사

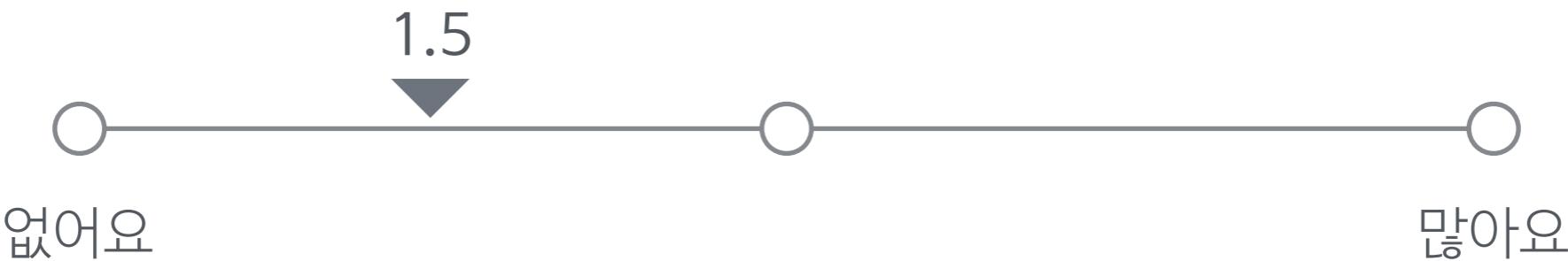
- “파이썬 프로그래밍 경험이 있나요?”



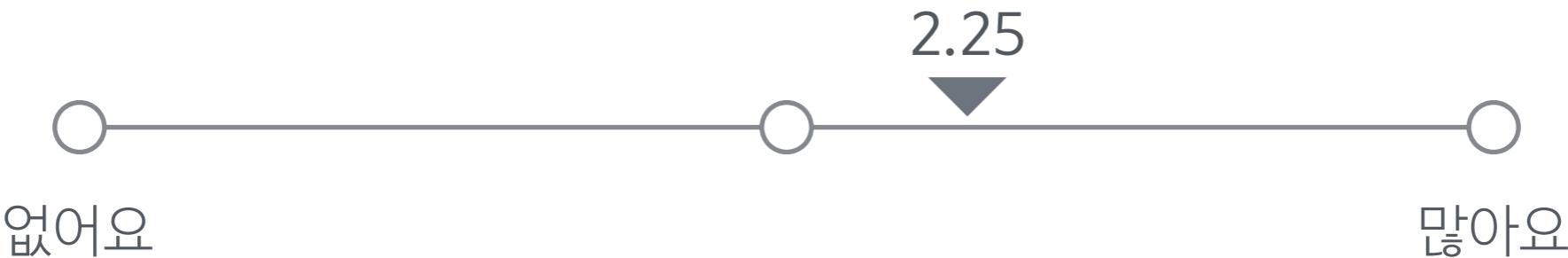
- “파이썬 외에 다른 프로그래밍 경험이 있나요?”

참여자 사전조사

- “파이썬 프로그래밍 경험이 있나요?”



- “파이썬 외에 다른 프로그래밍 경험이 있나요?”



여러분들의 경험과 실력을 믿고..

워크샵 프로그램 안내

워크샵 프로그램

일정	8/22(월)	8/23(화)	8/24(수)	8/25(목)	8/26(금)
주제	텍스트 마이닝	데이터 핸들링 및 시각화	기계학습		
내용	형태소 분석, 워드클라우드	데이터 전처리, 분석, 시각화	지도학습, 비지도학습	데이터 분석 프로젝트 기획	데이터 분석 프로젝트 발표
Python	KoNLPy Seaborn Wordcloud	BeautifulSoup4 pandas Matplotlib Seaborn	pandas scikit-learn statsmodels		

프로젝트

- 개인별 프로젝트로 진행
- 각자에게 의미있는/재미있는 데이터 + 강의 시간에 배운 각종 분석 기법
- 목요일 - 데이터, 분석 목표, 아이디어 등 발표
- 금요일 - 최종 결과 발표, 투표 및 시상
- 수요일 강의 후 다시 안내

데이터 과학

데이터 시대

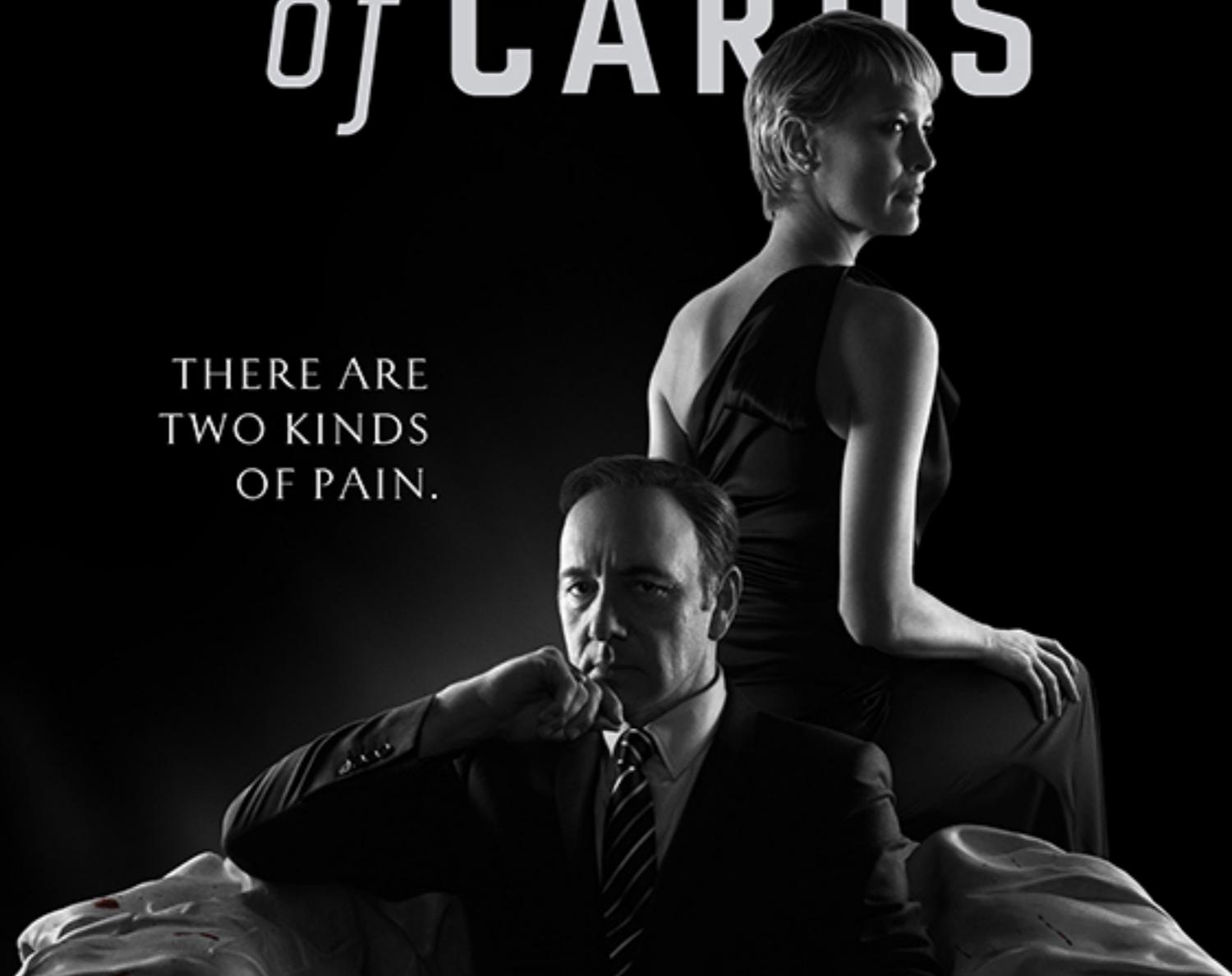
- 2018년 미국에서는 19만 명의 데이터 과학자가 부족해질 것이라 전망.
- IT 기술의 발전으로 점점 많은 산업계, 회사들에서 데이터 분석이 가능해지고 있다.
- IoT의 성장 → 데이터 양의 폭발적 성장
- 각종 기업/단체에서 데이터를 분석해 의사 결정에 활용한다.



HOUSE *of* CARDS



THERE ARE
TWO KINDS
OF PAIN.



빅데이터가 아니면 데이터 과학이 아닌가?

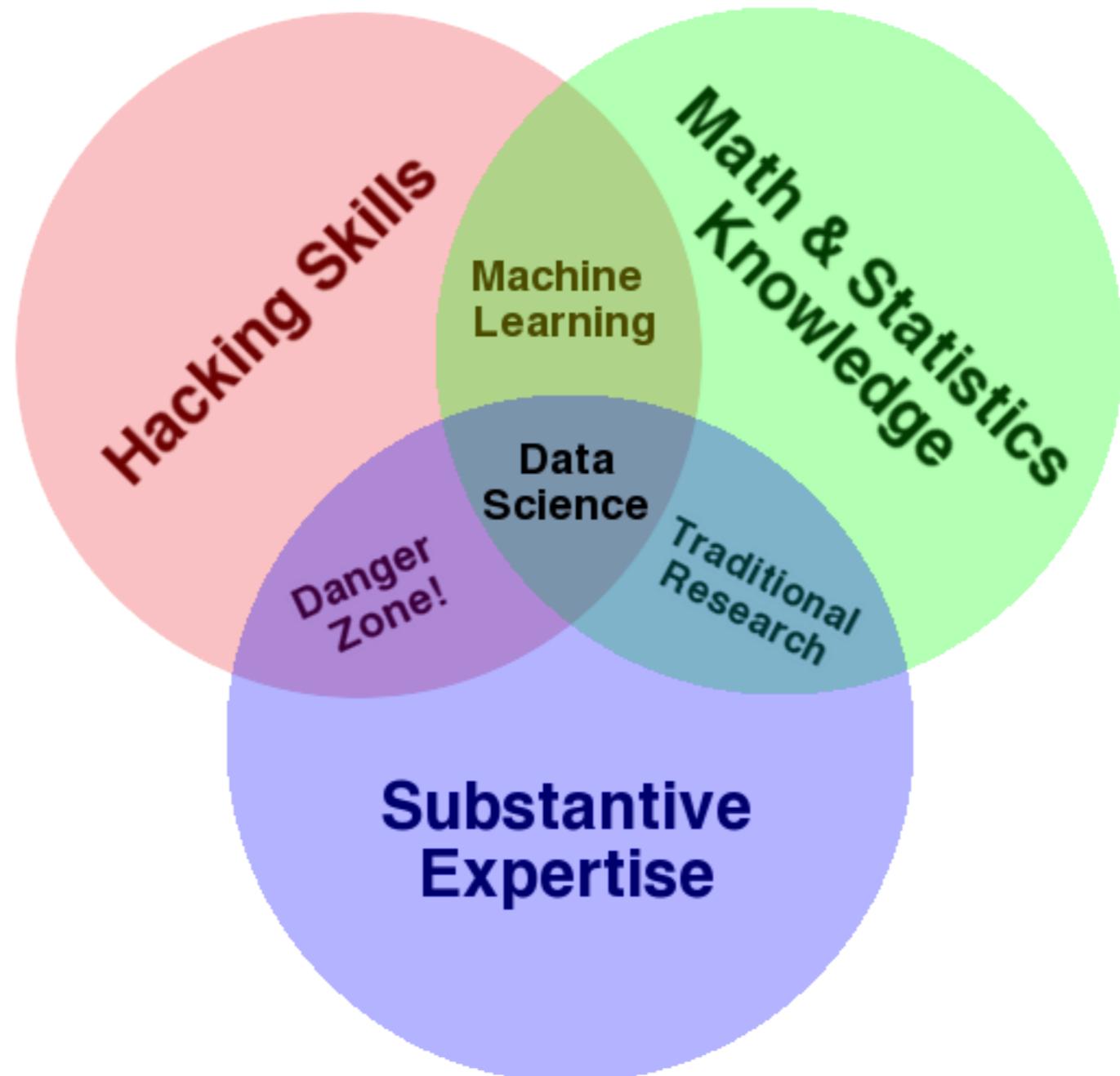
- 데이터 폭발 → 데이터 과학의 부흥
- 데이터가 많아야 하고, 분석 기법이 복잡해야 하고..
- ‘많은 양의 데이터’가 데이터 과학의 본질인가?

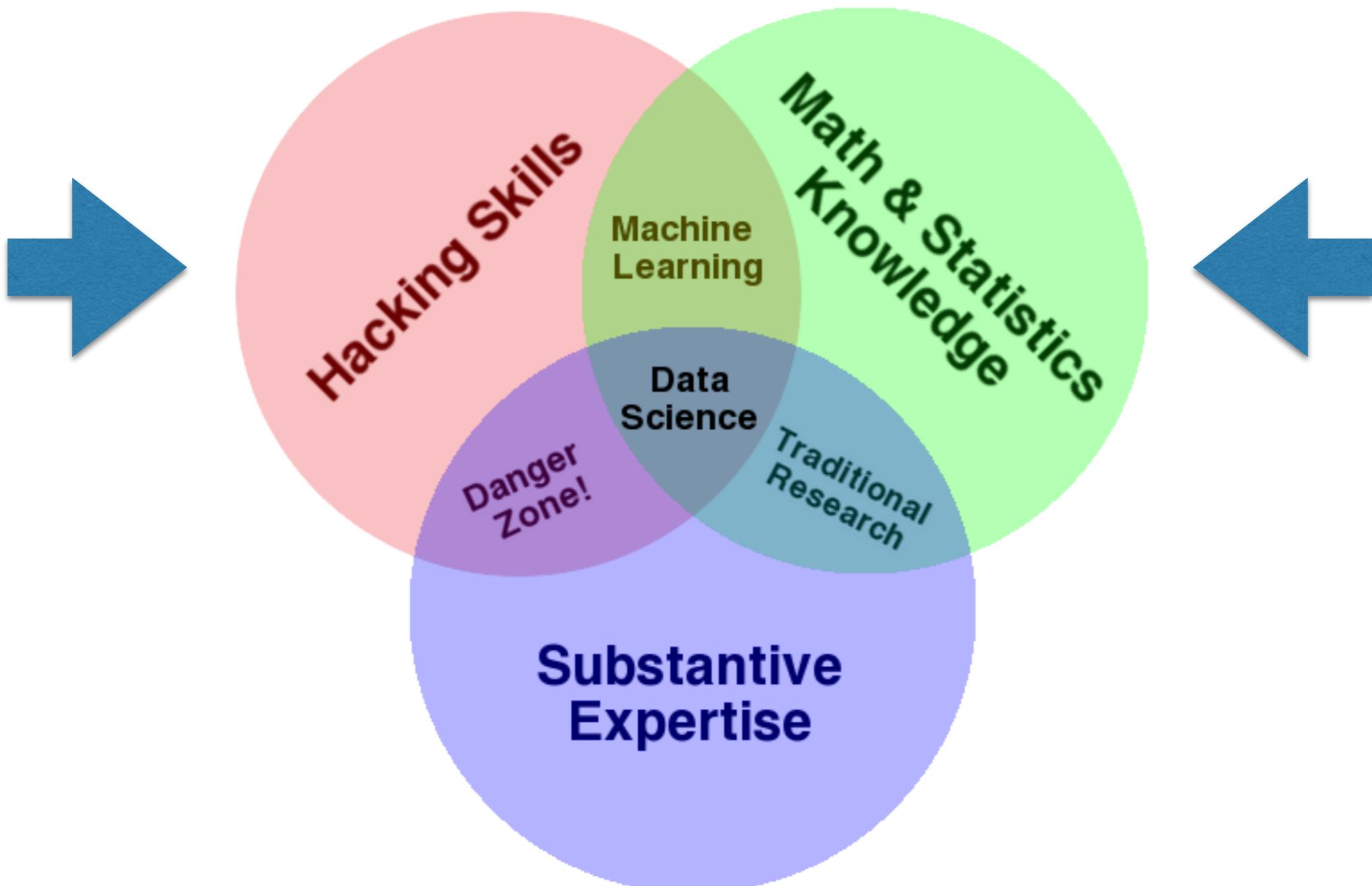
모두를 위한 데이터 과학

- 빅데이터, 데이터 과학... 쉽게 접근하기 힘들어 보이는 단어.
- **기술**로서의 데이터 과학 ↔ **사고방식**으로서의 데이터 과학
- 주변에서 일어나는 **현상에 대한 이해**를 돋고, 효과적인 **의사결정을 도와주는** 수단.

Data Thinking

- 현상에서 데이터를 찾고 수집한다.
- 데이터를 분석해 현상을 바르고 정확하게 이해한다.
- (Optional) 현상을 개선하거나 이해한 현상을 의사결정에 활용하고 실천한다.





왜 파이썬이죠?

파이썬의 장점

- 인터랙티브한 언어
- 타입에 신경 쓸 필요가 없다 → 빠르게 코딩할 수 있다
- 문법이 쉽다
- 학계에서 많이 사용 → **풍부한 라이브러리** 보유

오늘의 목표

텍스트 마이닝

이번 시간이 끝나면

- 텍스트 뭉치들을 분석하기 좋은 형태로 가공할 수 있습니다.
- 텍스트 뭉치에서 자주 등장하는 단어들을 한 눈에 보기 좋게 시각화할 수 있습니다.
- 텍스트 뭉치에서 같이 묶여 돌아다니는 단어들이 무엇인지 알아볼 수 있습니다.

텍스트 마이닝

텍스트 마이닝

- 깊고 심오한 세계..
- 정형 데이터 vs 비정형 데이터
- 텍스트 문서들 속에서 패턴을 발견하고 응용
- 세부 분야
 - 문서 군집
 - 특성 추출
 - 문서 요약
 - 필터링
 - 추천
 - 질의응답 시스템
 - ...

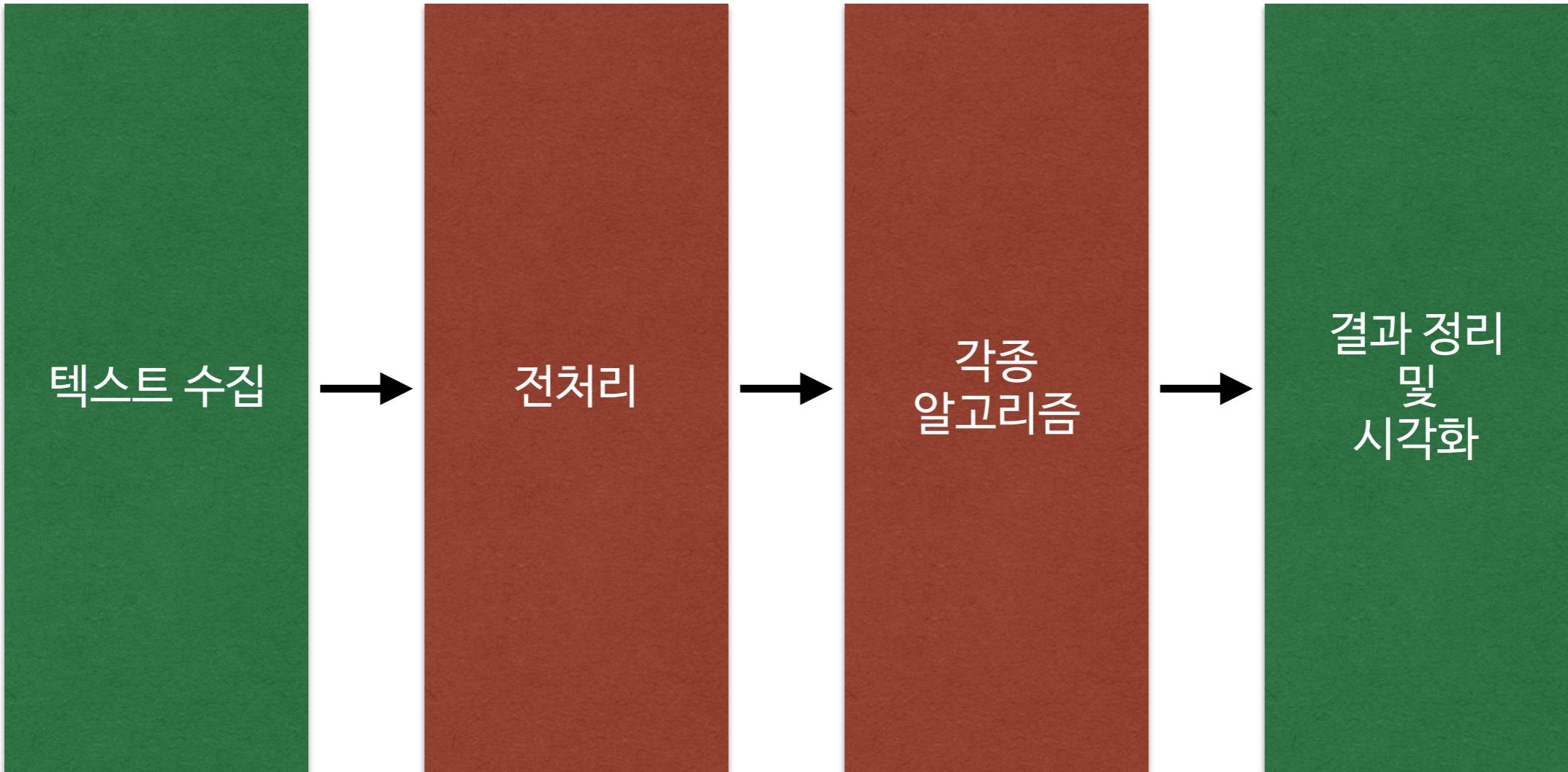
텍스트 마이닝

- 컴퓨터는 표(=테이블)를 좋아한다.
- “텍스트 마이닝은 참 쉬워요”

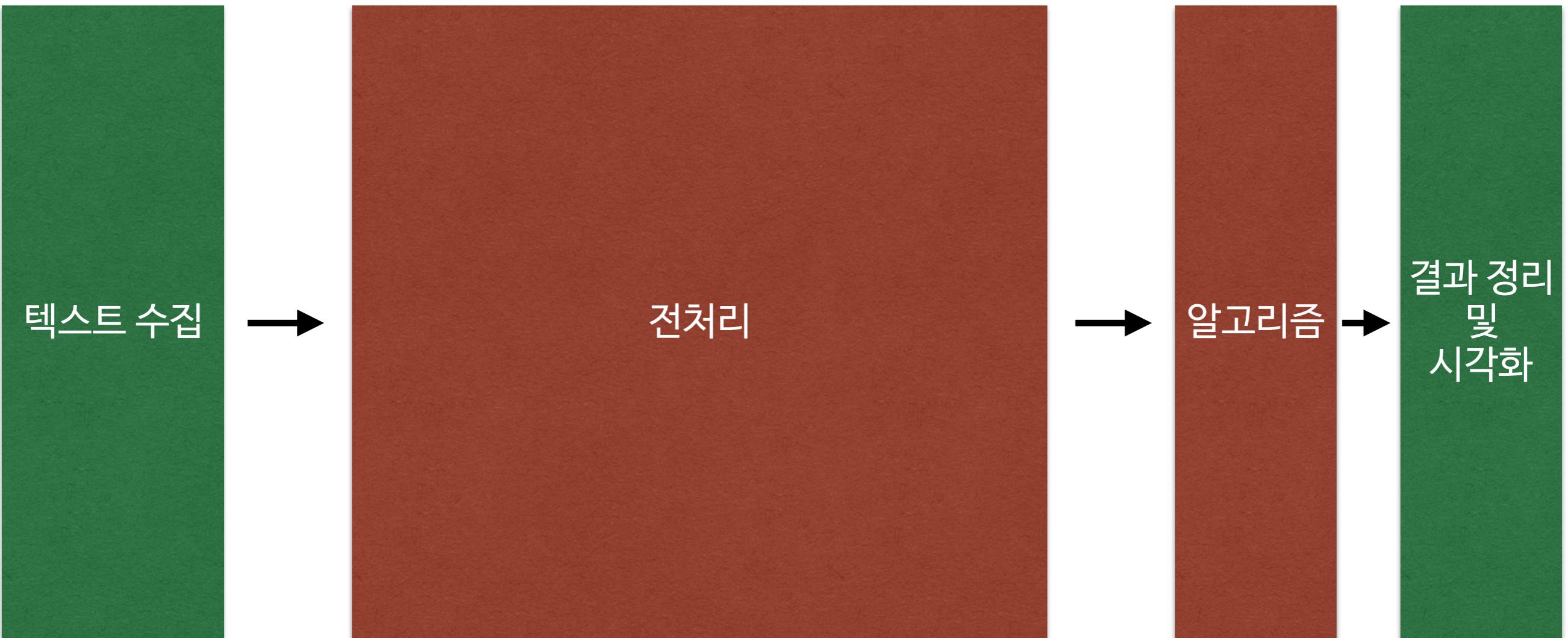


	텍스트	데이터	마이닝	도구	쉽다	알다	어렵다
문장1	1	0	1	0	1	0	0
문장2	0	1	1	1	0	1	0
문장3	1	1	1	1	0	0	1
문장4	0	1	1	0	1	0	0

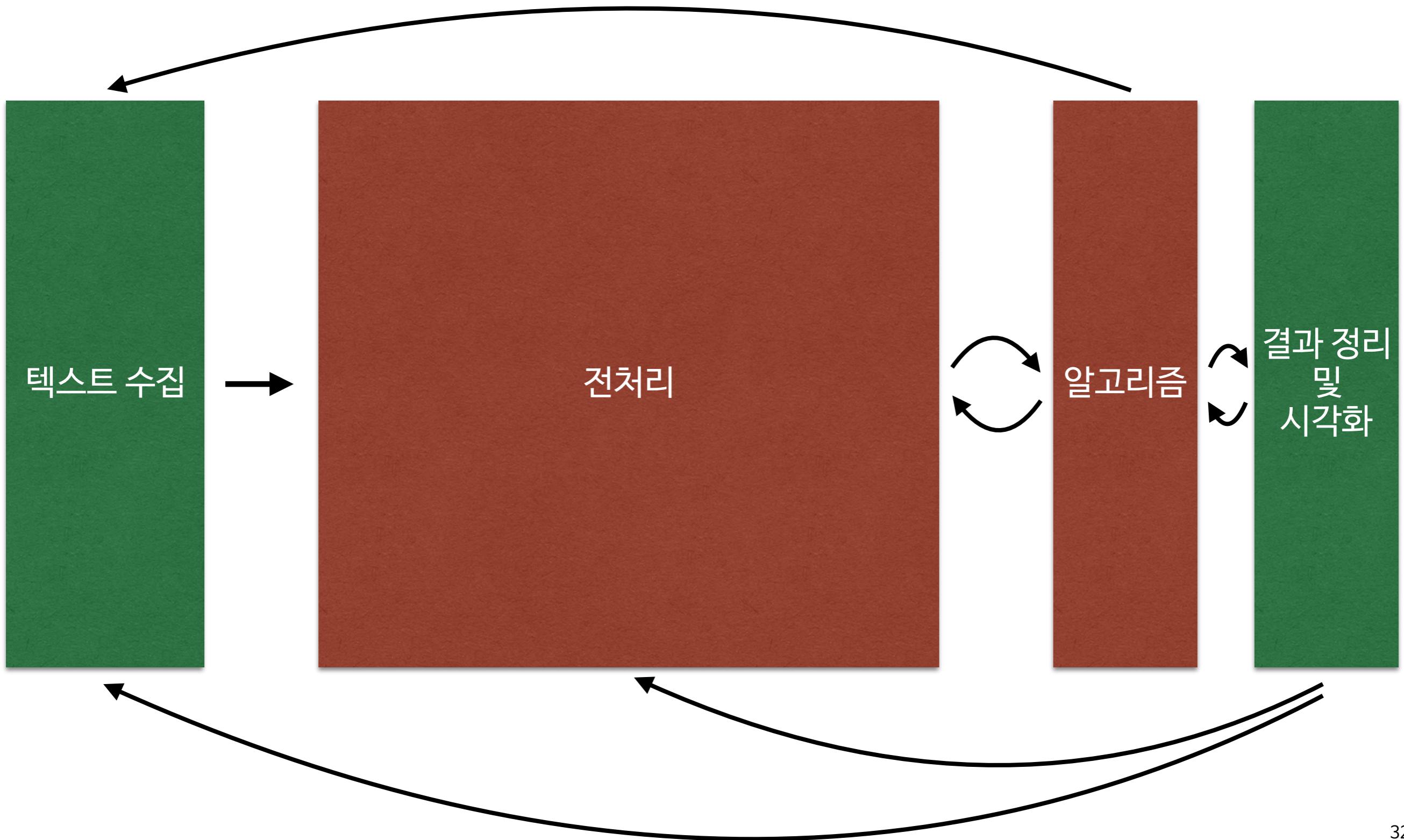
텍스트 마이닝의 단계



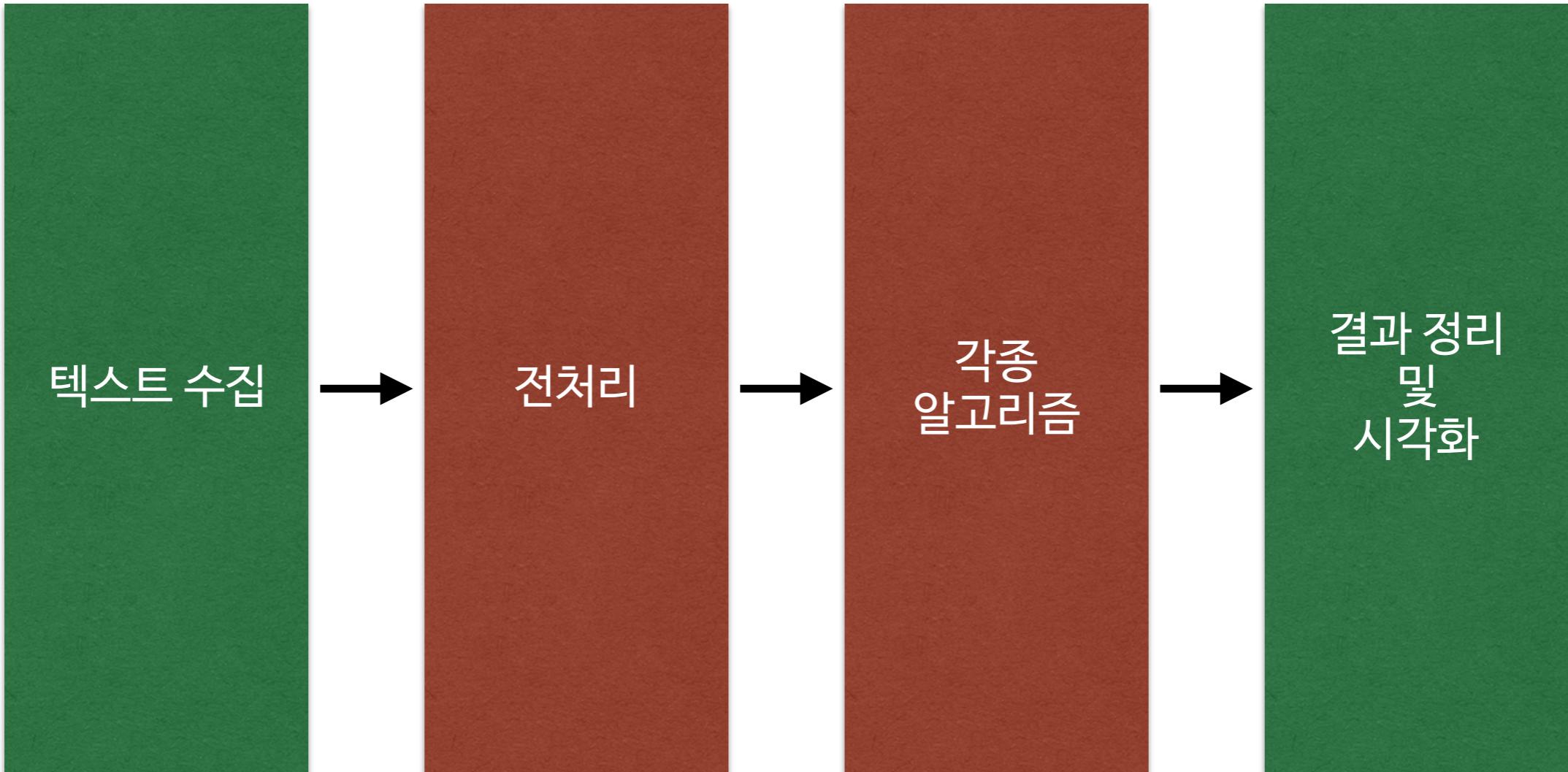
텍스트 마이닝의 단계



텍스트 마이닝의 단계



텍스트 마이닝의 단계



오늘 해볼 것

- 수집
 - 오늘은 패스합니다.
- 전처리
 - 문장 단위로 자르기
 - 형태소 분석하기
 - 분석에 사용할 형태소 필터링하기
 - 불용어(stopwords) 걸러내기
- 자주 등장하는 단어 시각화하기
- 함께 나타나는 단어 시각화하기

텍스트 마이닝

각종 라이브러리 소개

오늘 해볼 것

- 전처리
 - 문장 단위로 자르기
 - 형태소 분석하기
 - 분석에 사용할 형태소 필터링하기
 - 불용어(stopwords) 걸러내기
- 자주 등장하는 단어 시각화하기
- 함께 나타나는 단어 시각화하기

오늘 알아볼 라이브러리들

■ 텍스트 전처리, 형태소 분석

- KoNLPy

■ 함께 나타나는 단어 시각화

- NetworkX
- pynetviz

■ 시각화

- matplotlib
- seaborn
- wordcloud



전처리

Preprocessing

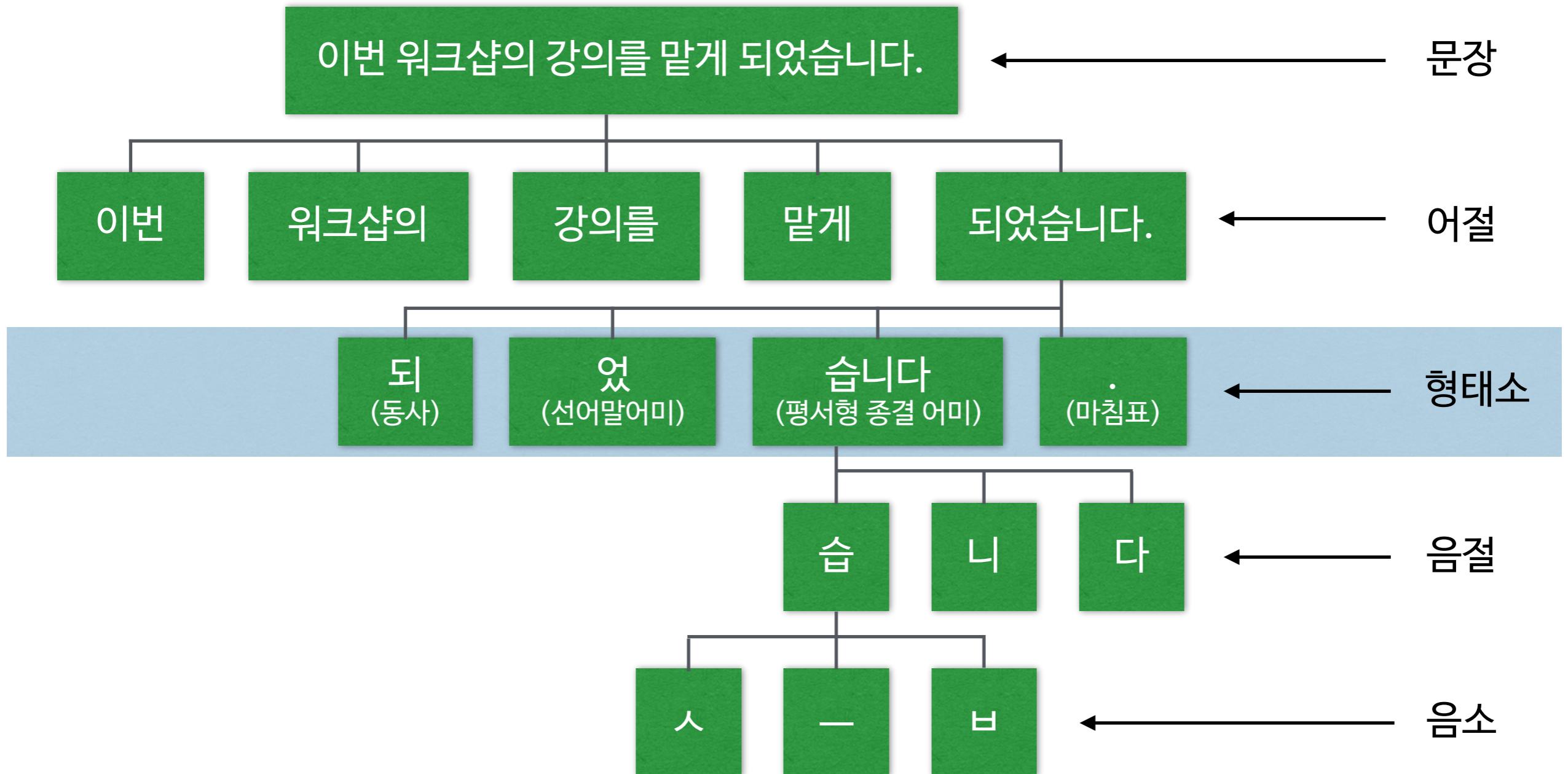
문장 분리

- “갤럭시노트7 열풍은 국내에 한정된 것이 아니다. 1차 출시국인 미국에서도 사전예약에서 이미 초기 공급 물량이 바닥이 났다. 미국 시장에서는 이미 추가물량 확보 요청이 잇따르고 있다는게 삼성전자 관계자의 설명이다.”

- 먼저 통짜 글을 문장 단위로 분해해줘야 합니다.
 - “갤럭시노트7 열풍은 국내에 한정된 것이 아니다.”
 - “1차 출시국인 미국에서도 사전예약에서 이미 초기 공급 물량이 바닥이 났다.”
 - “미국 시장에서는 이미 추가물량 확보 요청이 잇따르고 있다는게 삼성전자 관계자의 설명이다.”

형태소 분석

- 형태소 : 언어의 최소 의미 단위



형태소 분석

- 형태소를 분석해서 명사, 동사, 형용사 등을 추려냅니다.
- KoNLPy 패키지의 형태소 분석기를 사용합니다.
 - 다양한 형태소 분석기를 파이썬에서 사용할 수 있도록 도와줍니다.
 - 한나눔(카이스트, 1999)
 - 꼬꼬마(서울대, 2004)
 - 코모란(Shineware, 2014)
 - 은전한닢 프로젝트
 - Twitter Korean Text

형태소 분석

- “갤럭시노트7 열풍은 국내에 한정된 것이 아니다.”

갤럭시노트	고유 명사
7	숫자
열풍	일반 명사
은	보조사
국내	일반 명사
에	부사격 조사
한정	일반 명사
되	동사 파생 접미사
ㄴ	관형형 전성 어미
것	의존 명사
이	주격 조사
아니	부정 지정사
다	종결 어미
.	마침표

명사 추출

갤럭시노트

7

열풍

은

국내

에

한정

되

ㄴ

것

이

아니

다

.

고유 명사

숫자

일반 명사

보조사

일반 명사

부사격 조사

일반 명사

동사 파생 접미사

관형형 전성 어미

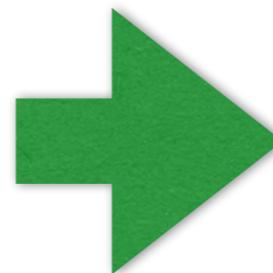
의존 명사

주격 조사

부정 지정사

종결 어미

마침표



갤럭시노트

고유 명사

열풍

일반 명사

국내

일반 명사

한정

일반 명사

같이 등장하는 단어 찾기

Co-occurrence matrix

	단어1	단어2	단어3	단어4	단어5
문서1	1	0	1	1	0
문서2	1	0	1	0	1
문서3	0	1	0	1	1
문서4	1	0	0	1	1
문서5	0	1	0	1	0
문서6	1	0	1	0	1



	단어1	단어2	단어3	단어4	단어5
단어1	0	0	3	2	3
단어2	0	0	0	2	1
단어3	3	0	0	1	2
단어4	2	2	1	0	2
단어5	3	0	2	2	0

같이 등장하는 단어 찾기

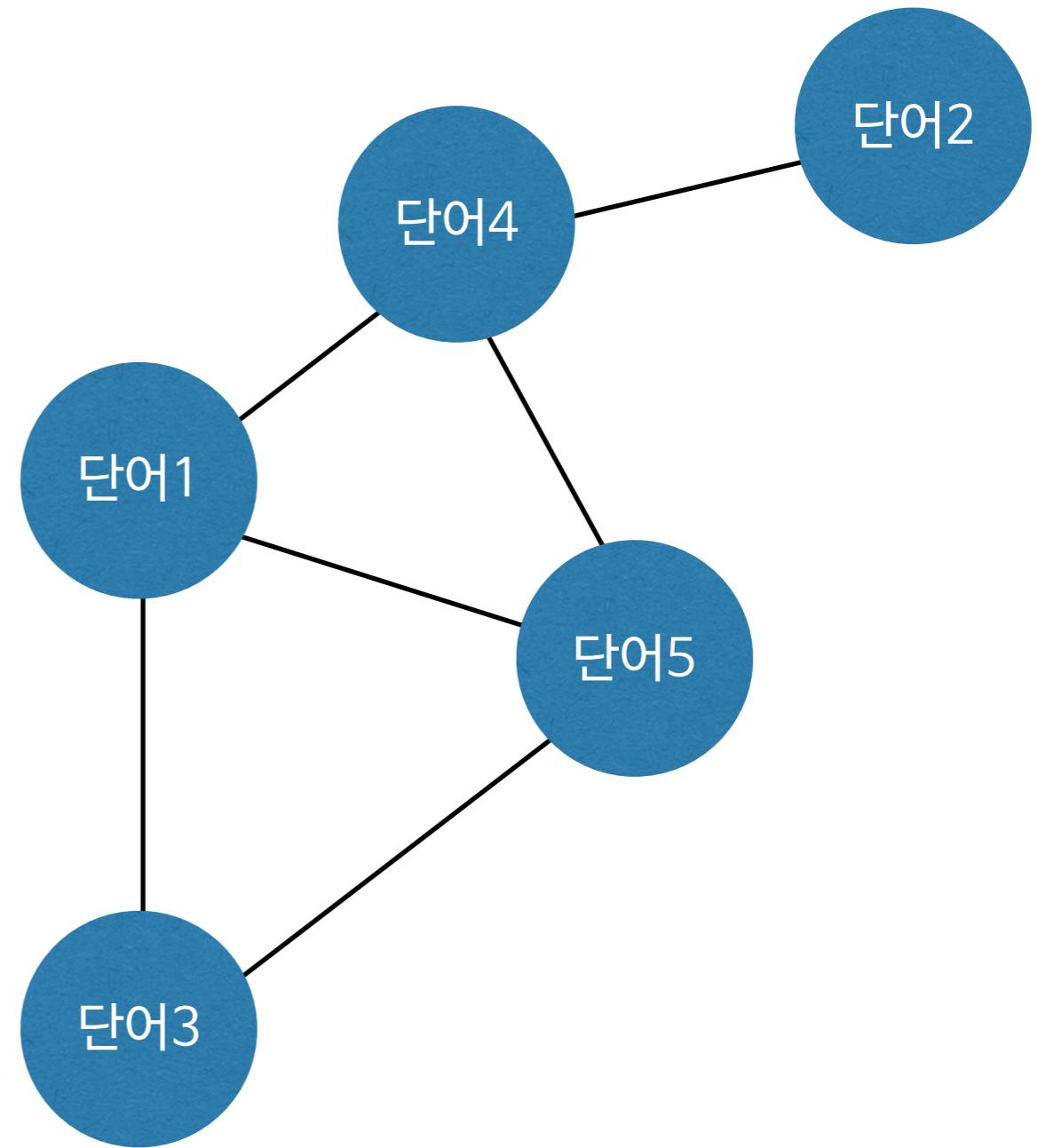
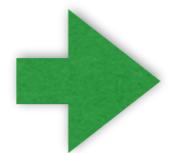
	단어1	단어2	단어3	단어4	단어5
단어1	0	0	3	2	3
단어2	0	0	0	2	1
단어3	3	0	0	1	2
단어4	2	2	1	0	2
단어5	3	0	2	2	0



단어1 - 단어3
단어1 - 단어4
단어1 - 단어5
단어2 - 단어4
단어3 - 단어5
단어4 - 단어5

같이 등장하는 단어 시각화하기

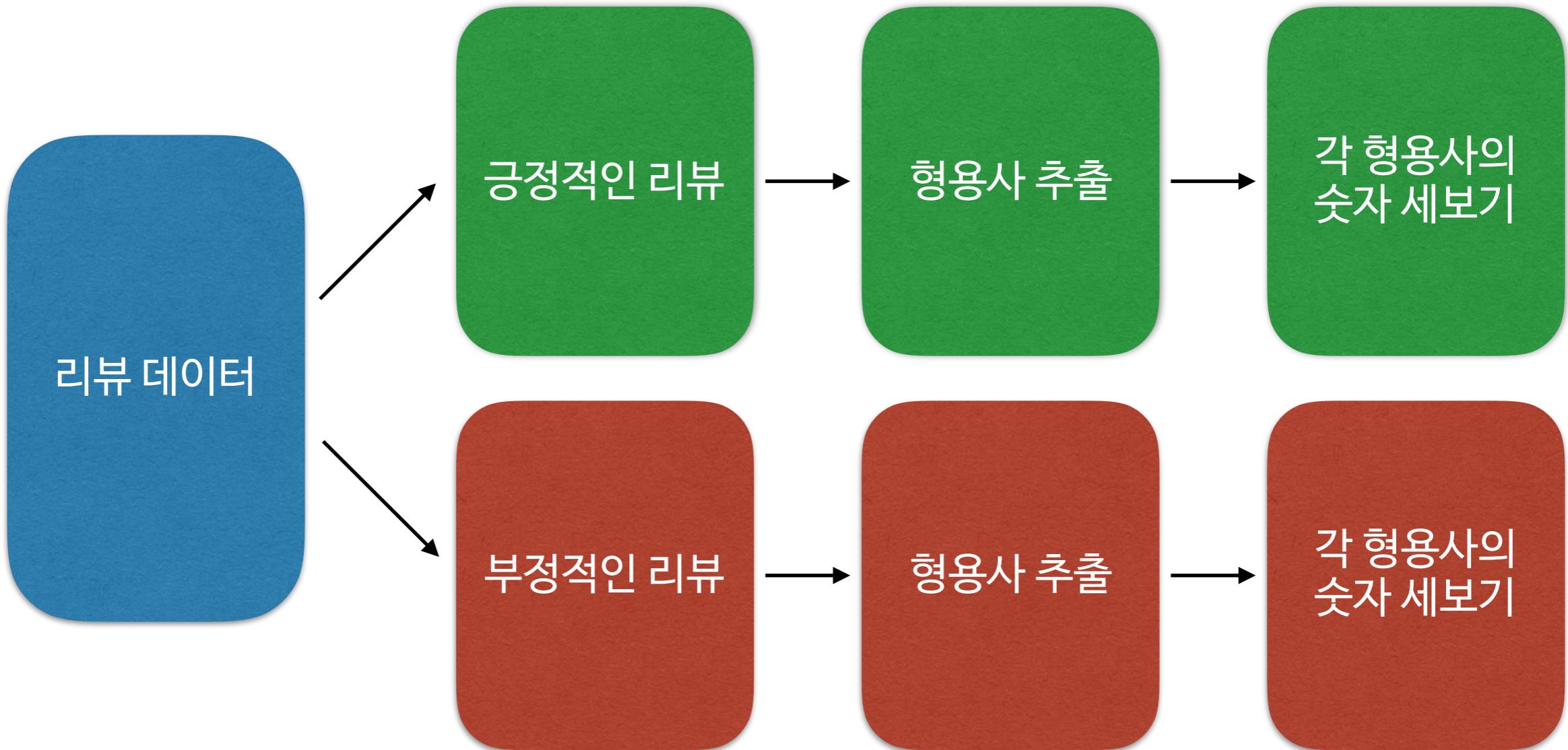
단어1 - 단어3
단어1 - 단어4
단어1 - 단어5
단어2 - 단어4
단어3 - 단어5
단어4 - 단어5



많이 등장하는 단어 보여주기

■ 네이버 영화 리뷰

- 높은 별점 / 낮은 별점 리뷰는 어떤 점이 다를까?
- 다른 감정 → 형용사를 보자!

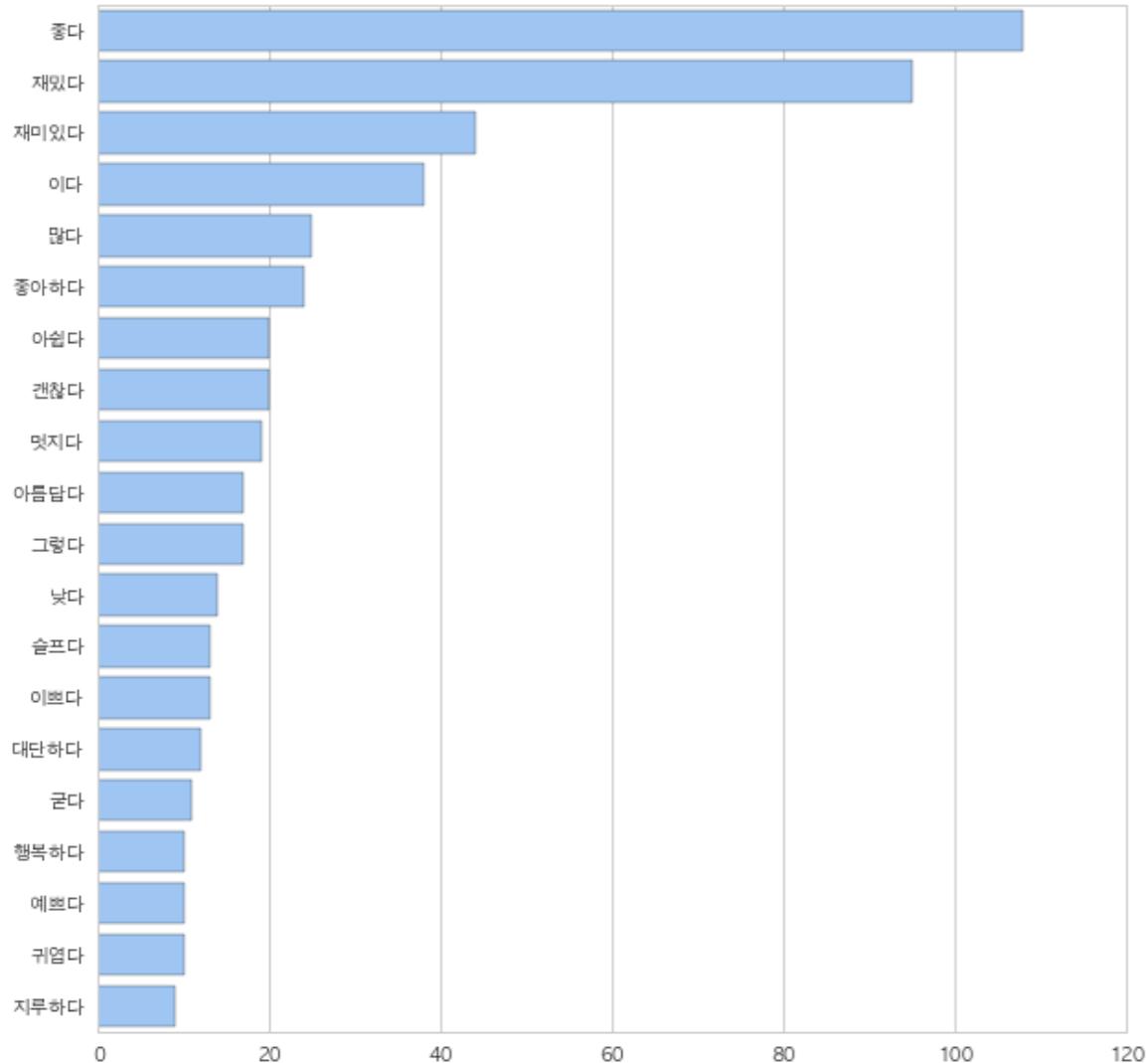


많이 등장하는 단어 보여주기

- 형용사가 등장한 횟수는 세봤다. 그러면?
- 그려보자
 - Bar Plot
 - 워드클라우드

Bar Plot & Word Cloud

긍정적인 리뷰의 형용사들



Tutorial

본격적으로 시작하기 전에

빠르게 파이썬 문법 정리!

파이썬 버전

- 2 vs 3
- 우리는 **파이썬 3**을 씁니다.
- 파이썬 2와 3은 약간 호환 안 되는 문법이 있어요.
 - 구글링하거나 라이브러리 설치 시 주의!

값 출력

```
print("hello")
```

숫자

`print(83)`

변수

```
num = 83  
print(num) # 83
```

참/거짓

```
print(True)      # True  
print(False)    # False
```

문자열

```
message = "문자열은 큰따옴표로 감싸줍니다."  
print(message)
```

```
message = '작은따옴표도 쓸 수 있습니다.'  
print(message)
```

리스트

```
mylist = [1, 2, 3]
mylist.append(5)

print(mylist[0])      # 1
print(mylist[3])      # 5
print(mylist[4])      # Error!
```

딕셔너리

```
mydict = {'a': 1, 'b': 24, 'c': 'hi'}
mydict['d'] = 100
mydict['e'] = [1, 2, 3, 4]
mydict[1] = {'name': 'John', 'age': 25}

print(mydict['d'])          # 100
print(mydict['e'][2])       # 3
```

길이

```
s = '안녕하세요'  
print(len(s))    # 5  
  
l = [4, 5, 2, 1]  
print(len(l))    # 4
```

for

```
for i in range(5):  
    print(i)      # 0, 1, 2, 3, 4
```

```
mylist = ['a', 'b', 'c']  
for item in mylist:  
    print(item) # a, b, c
```

List comprehension

```
mylist = [1, 2, 3, 4, 5]
squared = [x ** 2 for x in mylist]

print(squared)      # 1, 4, 9, 16, 25
```

if

```
if 2 < 3:  
    print("2는 3보다 작습니다.")
```

```
cond1 = False  
cond2 = True
```

```
if cond1:  
    print('cond1이 참이다.')  
elif cond2:  
    print('cond2가 참이다.')  
else:  
    print('둘 다 거짓이다.')
```

while

```
counter = 0
while counter < 10:
    counter += 1
```

함수

```
def sum(x, y):  
    return x + y
```

```
z = sum(2, 5)  
print(z)      # 7
```

```
def square_values(values):  
    results = []  
    for value in values:  
        results.append(value ** 2)  
    return results
```

```
l = [1, 2, 3]  
l = square_values(l)  
print(l)      # 1, 4, 9
```

클래스

```
class User:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def print_info(self):  
        print("name:", name)  
        print("age:", age)  
  
u = User('Hana', 20)  
u.print_info()      # name: Hana  
                  # age: 20
```

라이브러리 불러오기

```
import numpy  
x = numpy.array([1, 2, 3], dtype=numpy.int64)
```

```
import numpy as np  
x = np.array([1, 2, 3], dtype=np.int64)
```

```
from collections import defaultdict  
counts = defaultdict(int)  
counts['John'] += 1
```

라이브러리 설치

KoNLPy, Wordcloud, Seaborn, NLTK

KoNLPy 설치

```
$ pip install jpyper1-py3 konlpy
```

Wordcloud 설치

```
$ pip install wordcloud
```

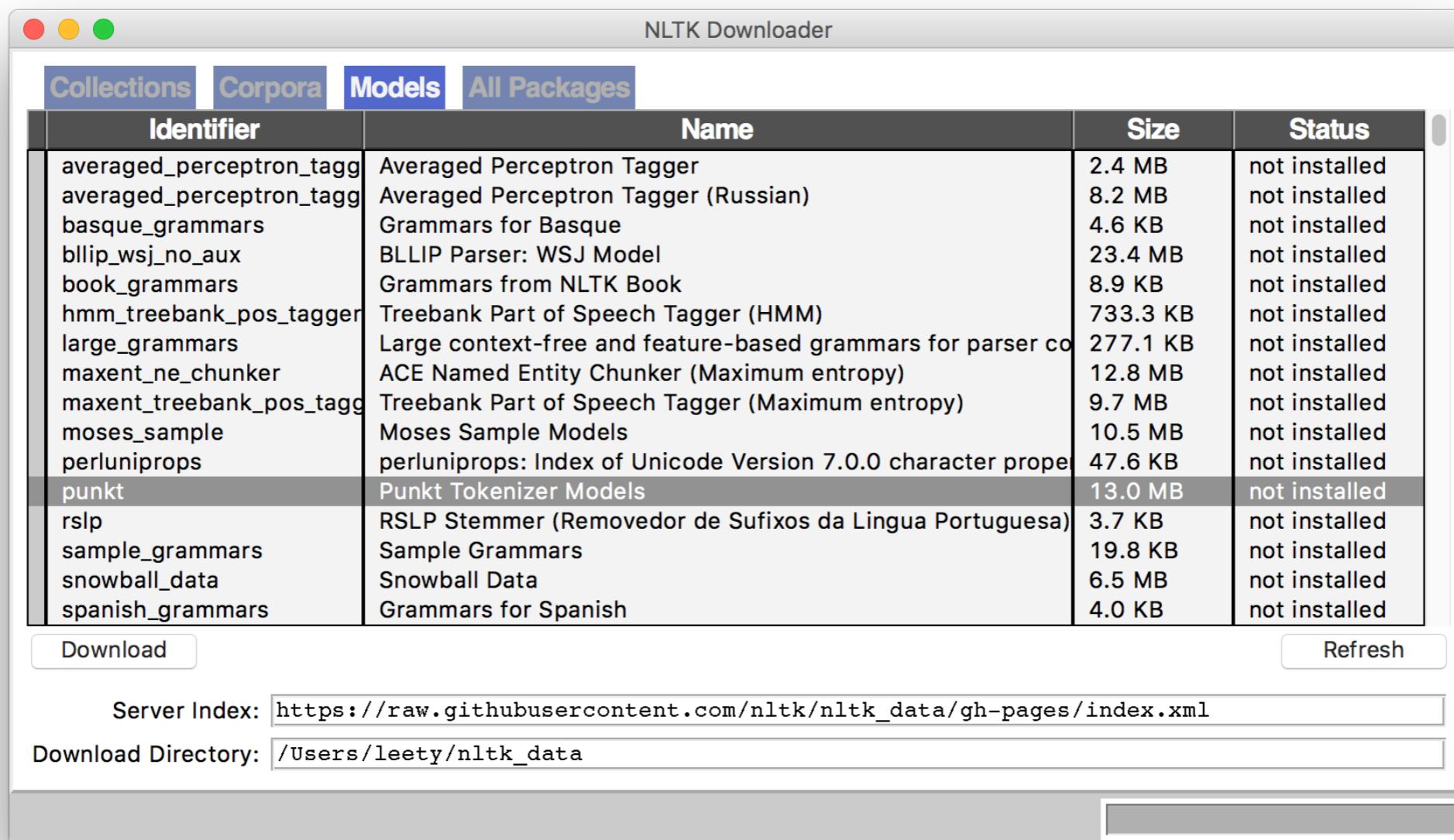
Seaborn 설치

```
$ pip install seaborn
```

NLTK 모듈 설치

```
$ python3 -c 'import nltk; nltk.download()'
```

■ Models 탭의 punkt 다운로드



Jupyter Notebook

실행 & 사용법 익히기

파이썬으로 텍스트 마이닝 해보기

데이터 내려받기

- <https://github.com/leety/itct-workshop>
- git clone 혹은 zip download
 - **data/** — 사용할 데이터
 - **ratings.txt** — 네이버 영화 리뷰 샘플
 - **statement.txt** — 세월호 대통령 담화문
 - **statement_next.txt** — 세월호 1주기 담화문
 - **08서울남산체 B.ttf** — 워드클라우드에 사용할 폰트
 - **notebooks/** — 샘플 노트북
 - **slides/** — 강의자료

1. 네이버 영화 리뷰 분석

워드클라우드 그리기

파일 내용 구경하기

```
!head ../data/ratings.txt
```

CSV 파일

- CSV(Comma Separated Values)
- 여러 variation
 - 다른 delimiter ('|', 탭 등등) 쓰기도 함.
- 우리의 예제 파일은 TSV 형식.

ratings.txt

```
In [1]: !head ../data/ratings.txt
```

id	document	label
8112052	어릴때보고 지금다시봐도 재밌어요ㅋㅋ	1
8132799	디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산업이 부러웠는데. 사실 우리나라에서도 그 어려운시절에 끝까지 열정을 지킨 노라노 같은 전통이있어 저와 같은 사람들이 꿈을 꾸고 이뤄나갈 수 있다는 것에 감사합니다.	1
4655635	폴리스스토리 시리즈는 1부터 뉴까지 버릴께 하나도 없음.. 최고.	1
9251303	와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이런게 진짜 영화지	1
10067386	안개 자욱한 밤하늘에 떠 있는 초승달 같은 영화.	1
2190435	사랑을 해본사람이라면 처음부터 끝까지 웃을수 있는영화	1
9279041	완전 감동입니다 다시봐도 감동	1
7865729	개들의 전쟁2 나오나요? 나오면 1빠로 보고 싶음	1
7477618	굿	1

- 첫 줄은 header
- label 값이 1이면 긍정, 0이면 부정.

CSV 파일 불러와서 긍/부정 분류하기

```
import csv

pos_review = []
neg_review = []

with open('../data/ratings.txt', 'r') as data_file:
    reader = csv.reader(data_file, delimiter='\t')
    header = next(reader)
    for row in reader:
        if int(row[2]) == 1:
            pos_review.append(row[1])
        else:
            neg_review.append(row[1])
```

리뷰 데이터를 부분적으로 사용하기

- 주어진 데이터는 긍정/부정 리뷰가 각각 10만 건
- 데이터가 꽤 많아서 형태소 분석하는 데에 시간이 많이 듈다.
- 1000개씩 랜덤 샘플링

1000개씩 랜덤 샘플링

```
import numpy as np

samples = np.random.choice(len(pos_review), 1000,
                           replace=False)
samples.sort()

pos_review = [pos_review[i] for i in samples]
neg_review = [neg_review[i] for i in samples]
```

형태소 분석해서 형용사 숫자 세기

```
from konlpy.tag import Twitter
from collections import Counter

def count_adjs(reviews):
    stopwords = ['있다', '없다', '이렇다', '같다', '아니다', '스럽다', '이다']

    twt = Twitter()
    counts = Counter()

    for review in reviews:
        tags = twt.pos(review, stem=True)
        for morph, tag in tags:
            if tag == 'Adjective' and morph not in stopwords:
                counts[morph] += 1

    return counts
```

형태소 분석해서 형용사 숫자 세기

```
pos_counts = count_adjs(pos_review)  
neg_counts = count_adjs(neg_review)
```

워드클라우드 그리기

```
%matplotlib inline
import matplotlib.pyplot as plt
from wordcloud import WordCloud

cloud = WordCloud(width=900, height=600,
                  font_path='..../data/08서울남산체_B.ttf',
                  background_color='white')
cloud = cloud.fit_words(pos_counts.items())

plt.figure(figsize=(15, 20))
plt.axis('off')
plt.imshow(cloud)
plt.show()
```

공정 리뷰



부정 리뷰



2. 세월호 대통령 담화문 분석

연관 단어 네트워크 그리기

담화문 내용 살펴보기

```
!cat ../data/statement.txt
```

데이터 로드하기

```
with open('../data/statement.txt', 'r') as data_file:  
    lines = data_file.read().splitlines()
```

데이터 로드하기

```
In [1]: with open('../data/statement.txt', 'r') as f:  
    lines = f.read().splitlines()  
lines
```

Out[1]: ['존경하는 국민 여러분, 세월호 침몰사고가 발생한지 오늘로 34일째가 되었습니다.',
 '.',
 '온 국민이 소중한 가족을 잃은 유가족들의 아픔과 비통함을 함께 하고 있습니다.',
 '.',
 '국민의 생명과 안전을 책임져야 하는 대통령으로서 국민 여러분께서 겪으신 고통에 진심으로 사과드립니다.',
 '.',
 '국민 여러분, 지난 한 달여 동안 국민 여러분이 같이 아파하고, 같이 분노하신 이유를 잘 알고 있습니다.',
 '.',
 '살릴 수도 있었던 학생들을 살리지 못했고, 초등대응 미숙으로 많은 혼란이 있었고, 불법 과적 등으로 이미 안전에 많은 문제가 예견되었는데도 바로 잡지 못한 것에 안타까워하고 분노하신 것이라 생각합니다.',
 '.',
 '채 피지도 못한 많은 학생들과 마지막 가족여행이 되어 버린 혼자 남은 아이, 그 밖에 눈물로 이어지는 희생자들의 안타까움을 생각하며 저도 번민으로 잠을 이루지 못한 나날이었습니다.',
 '.',
 '그들을 지켜주지 못하고, 그 가족들의 여행길을 지켜 주지 못해 대통령으로서 비애감이 듭니다.',
 '.',
 '이제 여기에 제대로 대처하겠다 민족 치즈 채이오 대통령이 되시게 이습니다.]

빈 줄 정리하기

```
sents = []
for line in lines:
    if line != '':
        sents.append(line)

# Or use list comprehension
sents = [line for line in lines if line != '']
```

형태소 분석기 연습

```
sent = sents[0]  
print(sent)
```

```
from konlpy.tag import Komoran  
tagger = Komoran()  
tagger.pos(sent)
```

형태소 분석 결과 보기

```
Out[3]: [ ('존경', 'NNG'),  
          ('하', 'XSV'),  
          ('는', 'ETM'),  
          ('국민', 'NNG'),  
          ('여러분', 'NP'),  
          ('', 'SP'),  
          ('세월호 침몰사고', 'NNP'),  
          ('가', 'JKS'),  
          ('발생', 'NNG'),  
          ('하', 'XSV'),  
          ('ㄴ지', 'EC'),  
          ('오늘', 'NNG'),  
          ('로', 'JKB'),  
          ('34', 'SN'),  
          ('일', 'NNB'),  
          ('째', 'XSN'),  
          ('가', 'JKS'),  
          ('되', 'VV'),  
          ('었', 'EP'),  
          ('습니다', 'EF'),  
          ('.', 'SF') ]
```

(형태소, 태그)로 이루어진 Tuple(튜플)의 리스트

Tuple의 각 성분에 접근하기

```
tags = tagger.pos(sent)
```

리스트의 성분처럼 접근할 수 있다.

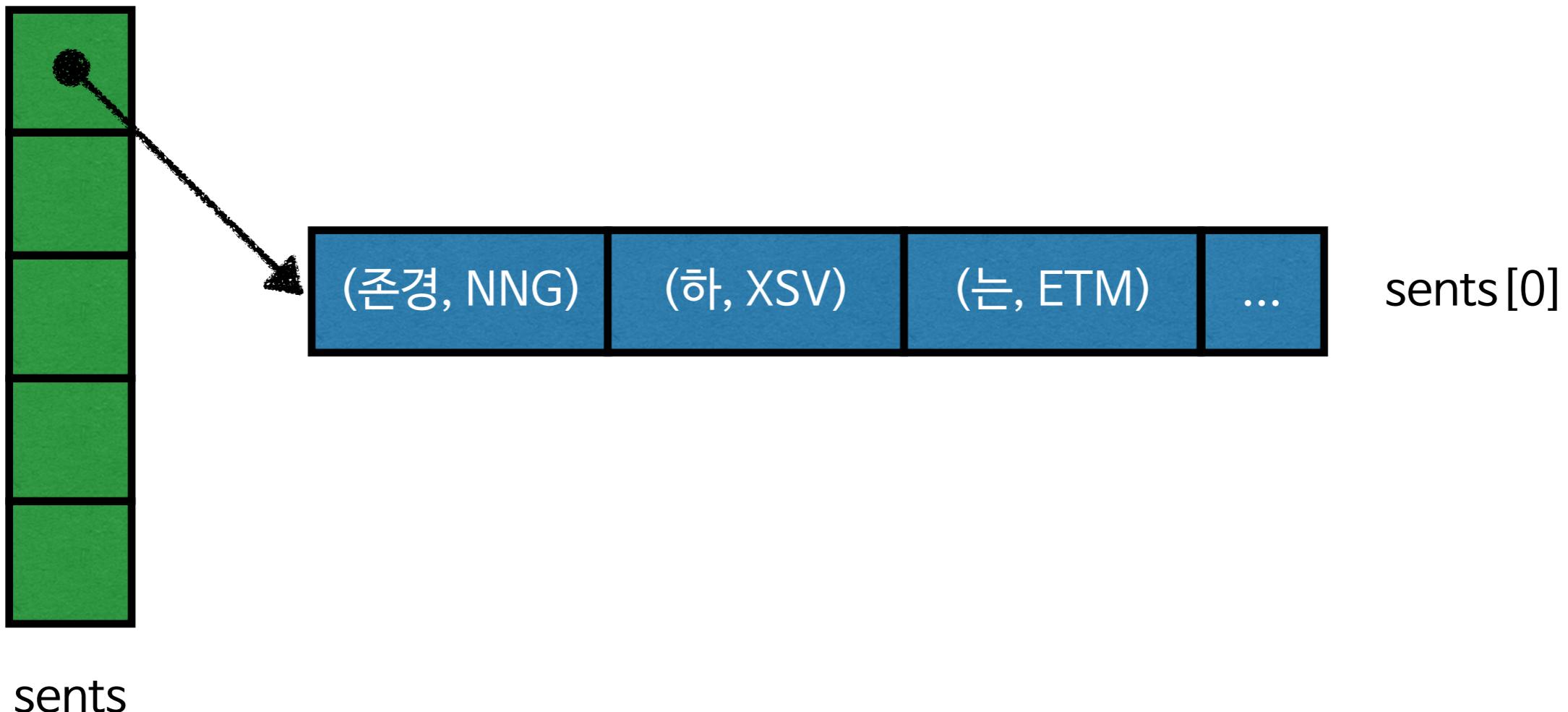
```
pair = tags[0]  
print(pair[0], pair[1])
```

각 변수에 곧바로 성분들을 할당할 수도 있다.

```
morph, tag = tags[0]  
print(morph, tag)
```

전체 문장들을 각각 형태소 분석하기

```
sents = [tagger.pos(sent) for sent in sents]
```



Sentence-term matrix

- 이런 테이블을 만들어야 합니다.

	세월호	재난	적극	활동	희망	인명	해경
문장1	1	0	1	0	1	0	0
문장2	0	1	1	1	0	1	0
문장3	1	1	1	1	0	0	1
문장4	0	1	1	0	1	0	0

Sentence-term matrix

- 이런 테이블을 만들어야 합니다.
 - 단어 중에서는 **명사만** 사용합니다.
 - **명사의 목록**이 필요합니다.

	세월호	재난	적극	활동	희망	인명	해경
문장1	1	0	1	0	1	0	0
문장2	0	1	1	1	0	1	0
문장3	1	1	1	1	0	0	1
문장4	0	1	1	0	1	0	0

명사 리스트 만들기

```
nouns = set()  
  
for sent in sents:  
    for morph, tag in sent:  
        if tag in ['NNP', 'NNG']:  
            nouns.add(morph)  
  
nouns = list(nouns)  
noun_index = {noun: i for i, noun in enumerate(nouns)}
```

문장-단어 행렬 만들기

- (전체 문장 수) X (등장하는 명사의 가짓수) 크기의 빈 행렬을 만든다.
- 형태소로 쪼개진 각 문장에 대해서,
 - 각 형태소들 중 일반명사 또는 고유명사가 있다면
 - 행렬의 (문장 번호, 명사 번호) 칸에 1을 넣는다.

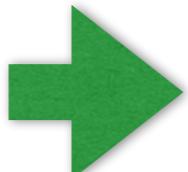
문장-단어 행렬 만들기

```
import numpy as np
occurs = np.zeros([len(sents), len(nouns)])

for i, sent in enumerate(sents):
    for morph, tag in sent:
        if tag in ['NNP', 'NNG']:
            index = noun_index[morph]
            occurs[i][index] = 1
```

공존 단어 행렬 만들기

	세월호	재난	활동	희망
문장1	1	0	0	1
문장2	0	1	1	0
문장3	1	1	1	0
문장4	0	1	0	1



	세월호	재난	활동	희망
세월호	0	1	1	1
재난	1	0	2	1
활동	1	2	0	0
희망	1	1	0	0

공존 단어 행렬 만들기

	세월호	재난	활동	희망		세월호	재난	활동	희망
문장1	1	0	0	1		0	1	1	1
문장2	0	1	1	0		1	0	2	1
문장3	1	1	1	0		1	2	0	0
문장4	0	1	0	1		1	1	0	0

공존 단어 행렬 만들기

	세월호	재난	활동	희망		세월호	재난	활동	희망
문장1	1	0	0	1			0	1	1
문장2	0	1	1	0		재난	1	0	2
문장3	1	1	1	0		활동	1	2	0
문장4	0	1	0	1		희망	1	1	0

공존 단어 행렬 만들기

The diagram illustrates the process of creating a co-occurrence matrix from a document-term matrix. It shows two tables side-by-side, connected by a large green arrow pointing from left to right.

Original Document-Term Matrix:

	세월호	재난	활동	희망
문장1	1	0	0	1
문장2	0	1	1	0
문장3	1	1	1	0
문장4	0	1	0	1

Transformed Co-Occurrence Matrix:

	세월호	재난	활동	희망
세월호	0	1	1	1
재난	1	0	2	1
활동	1	2	0	0
희망	1	1	0	0

The transformation is achieved by calculating the co-occurrence counts between terms across all documents. For example, the term '세월호' co-occurred with '재난' (1), '활동' (1), and '희망' (1) in the original matrix, which becomes the value '1' in the '세월호' row of the second table. The diagonal elements in the second table represent the self-co-occurrence count for each term.

공존 단어 행렬 만들기

	세월호	재난	활동	희망		세월호	재난	활동	희망
문장1	1	0	0	1		세월호	0	1	1
문장2	0	1	1	0	→	재난	1	0	2
문장3	1	1	1	0		활동	1	2	0
문장4	0	1	0	1		희망	1	1	0

공존 단어 행렬 만들기

	세월호	재난	활동	희망		세월호	재난	활동	희망
문장1	1	0	0	1			0	1	1
문장2	0	1	1	0		재난	1	0	2
문장3	1	1	1	0		활동	1	2	0
문장4	0	1	0	1		희망	1	1	0

공존 단어 행렬 만들기

The diagram illustrates the process of creating a co-occurrence matrix (document-term matrix) by transposing it. It shows two tables side-by-side, connected by a large green arrow pointing from left to right.

Left Table (Row-Oriented Format):

	세월호	재난	활동	희망
문장1	1	0	0	1
문장2	0	1	1	0
문장3	1	1	1	0
문장4	0	1	0	1

Right Table (Column-Oriented Format):

	세월호	재난	활동	희망
세월호	0	1	1	1
재난	1	0	2	1
활동	1	2	0	0
희망	1	1	0	0

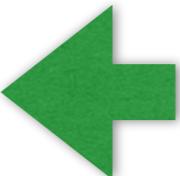
공존 단어 행렬 만들기

	세월호	재난	활동	희망
문장1	1	0	0	1
문장2	0	1	1	0
문장3	1	1	1	0
문장4	0	1	0	1

공존 단어 행렬 만들기

transpose

	문장1	문장2	문장3	문장4
세월호	1	0	1	0
재난	0	1	1	1
활동	0	1	1	0
희망	1	0	0	1



	세월호	재난	활동	희망
문장1	1	0	0	1
문장2	0	1	1	0
문장3	1	1	1	0
문장4	0	1	0	1

공존 단어 행렬 만들기

	문장1	문장2	문장3	문장4		세월호	재난	활동	희망	
세월호	1	0	1	0	.dot (문장1	1	0	0	1
재난	0	1	1	1		문장2	0	1	1	0
활동	0	1	1	0		문장3	1	1	1	0
희망	1	0	0	1		문장4	0	1	0	1
	세월호	재난	활동	희망		세월호	1	1	1	1
세월호	2	1	1	1	=	재난	1	3	2	1
재난	1	3	2	1		활동	1	2	2	0
활동	1	2	2	0		희망	1	1	0	2
희망	1	1	0	2						

공존 단어 행렬 만들기

```
co_occurs = occurs.T.dot(occurs)
```

연관 단어 그래프 만들기

- 그래프 = 노드(node)와 엣지(edge)의 집합
 - 노드: 그래프의 점. 등장하는 명사.
 - 엣지: 노드 사이를 잇는 선. 단어가 연관돼있으면 긋는다.
- NetworkX 라이브러리를 사용해 그래프를 표현한다.

연관 단어 그래프 만들기

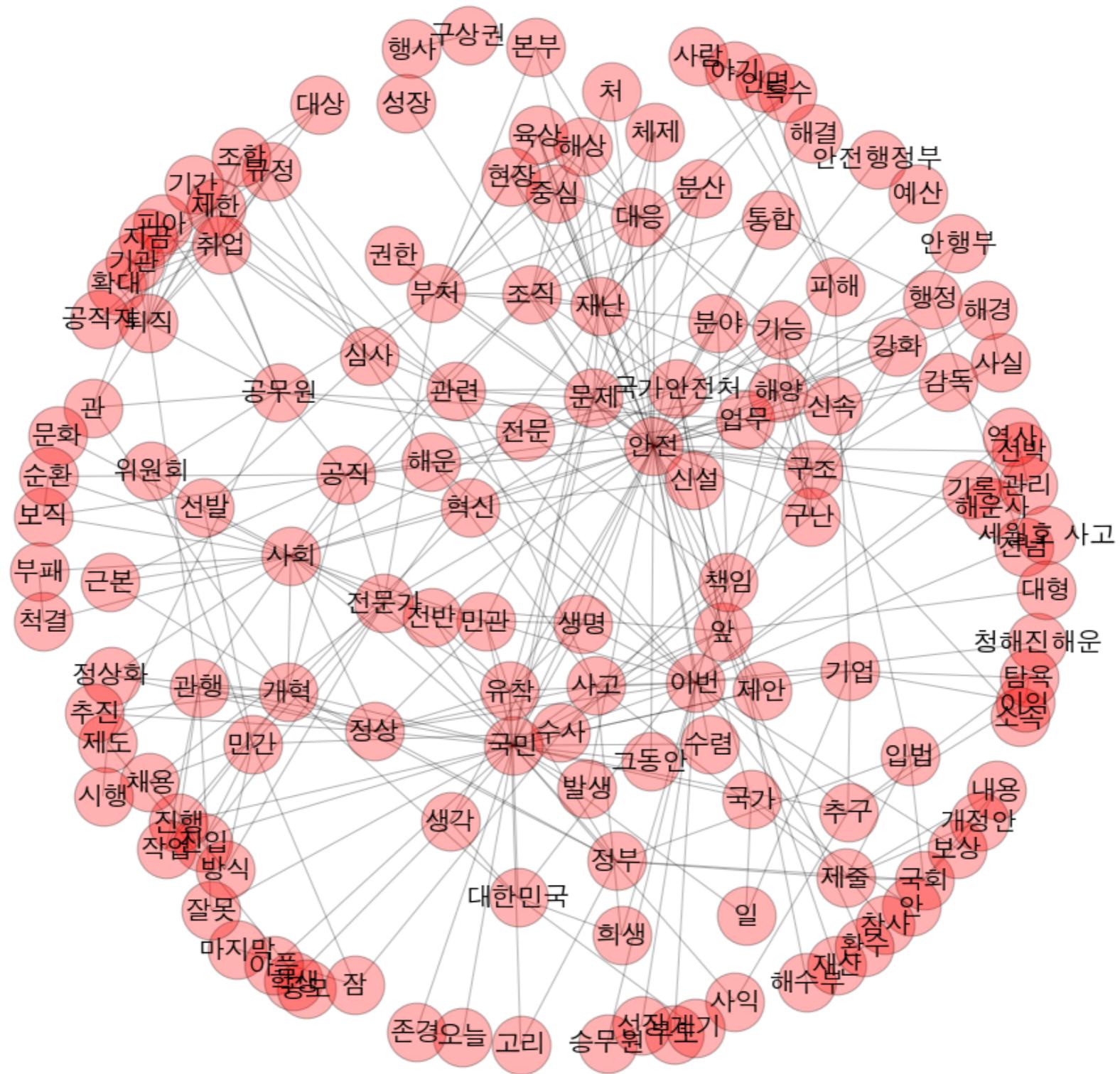
```
import networkx as nx
G = nx.Graph()

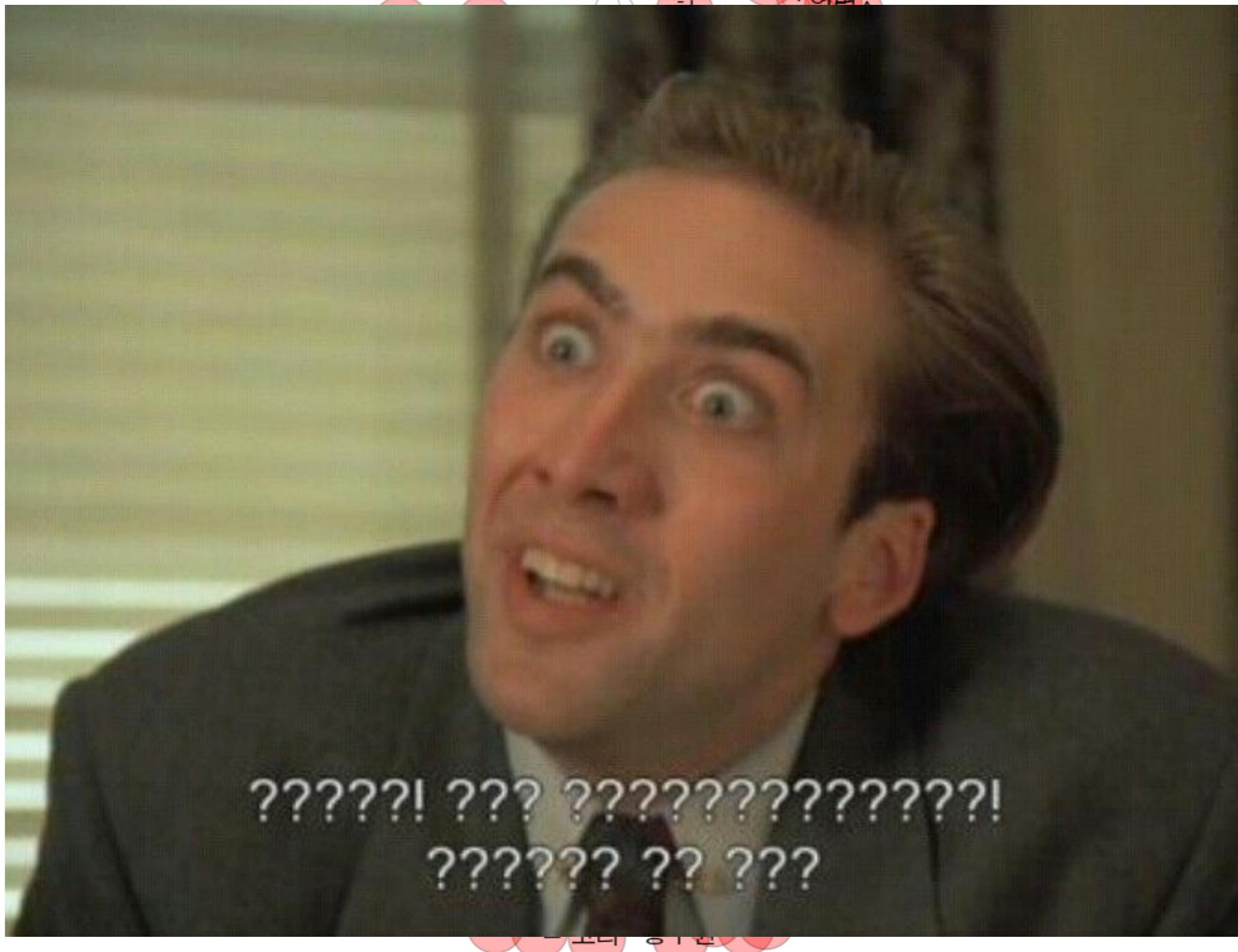
for i in range(len(nouns)):
    for j in range(i + 1, len(nouns)):
        if co_occurs[i][j] > 1:
            G.add_edge(nouns[i], nouns[j])
```

연관 단어 그래프 시각화

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 15))
layout = nx.spring_layout(G, k=.1)
nx.draw(G, pos=layout, with_labels=True, \
        font_size=20, font_family="AppleGothic", \
        alpha=0.3, node_size=2000)
plt.show()
```



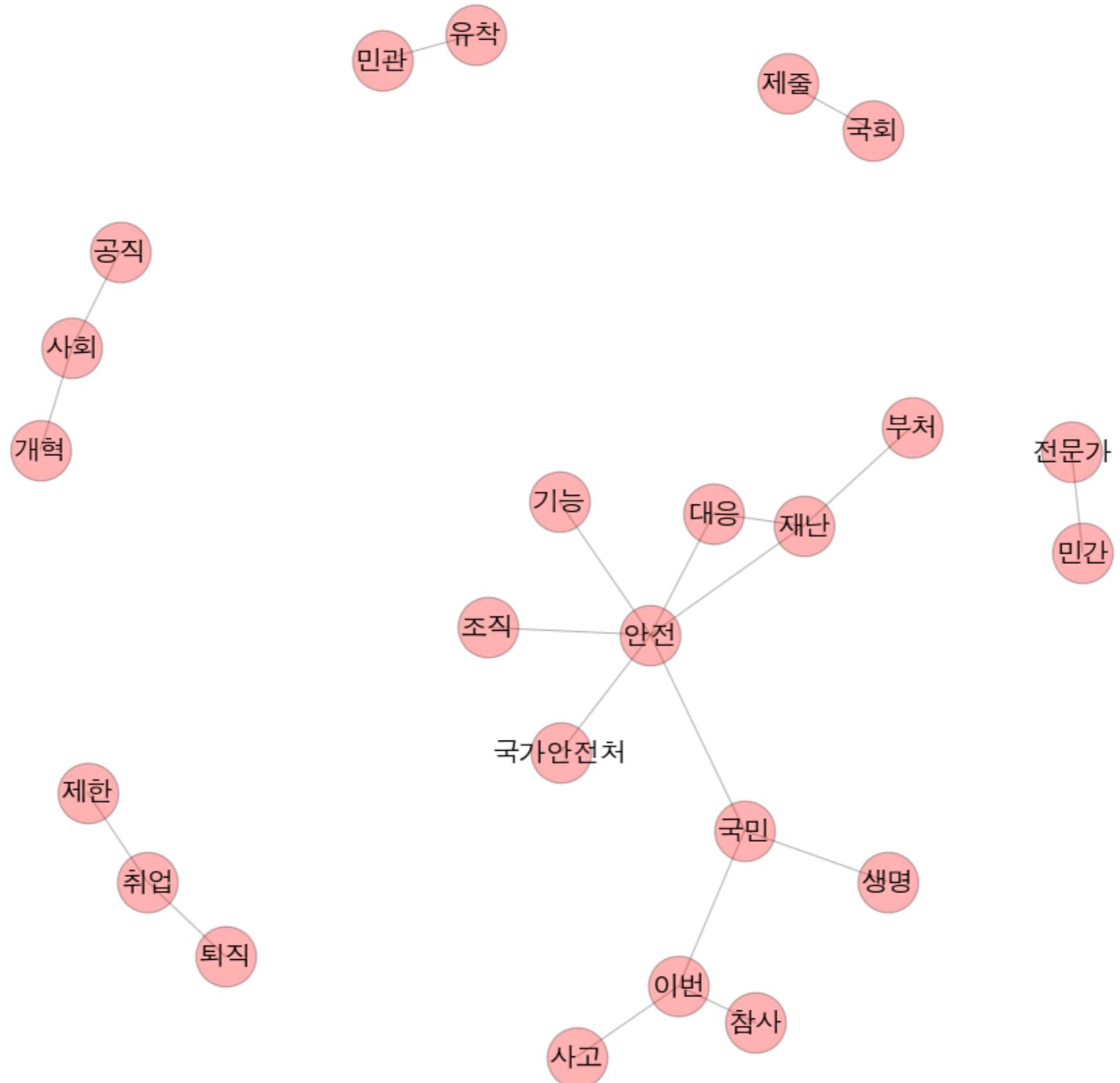


— 고나 융기준

연관 단어 그래프 만들기

```
import networkx as nx
G = nx.Graph()

for i in range(len(nouns)):
    for j in range(i + 1, len(nouns)):
        if co_occurs[i][j] > 3:
            G.add_edge(nouns[i], nouns[j])
```



1주기 담화문 분석

- 직접 해보세요!
- data/statement_next.txt
- 주의할 점
 - 한 줄에 문장이 여러 개 있습니다. 문장을 쪼개줘야 합니다.
 - nltk 패키지의 문장 쪼개기 함수를 사용하세요.
 - **from nltk.tokenize import sent_tokenize**
 - **sent_tokenize(sentences)**