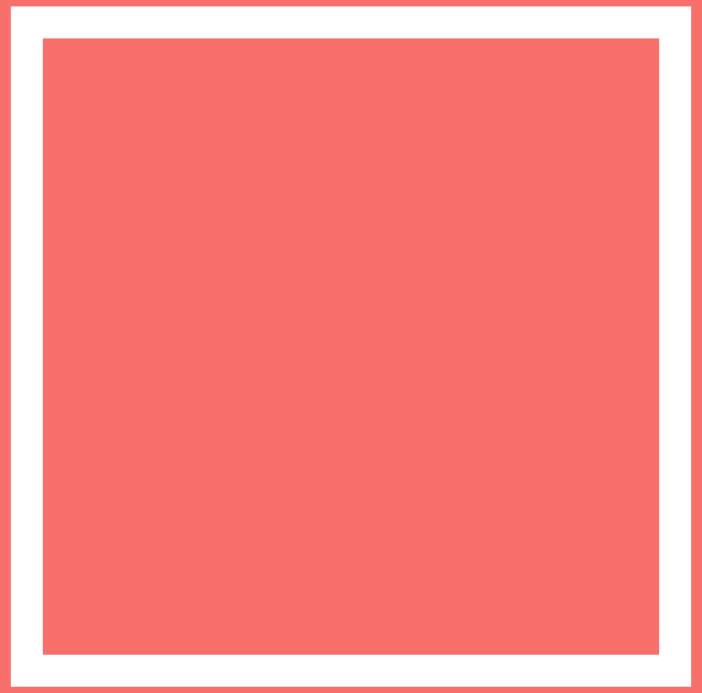


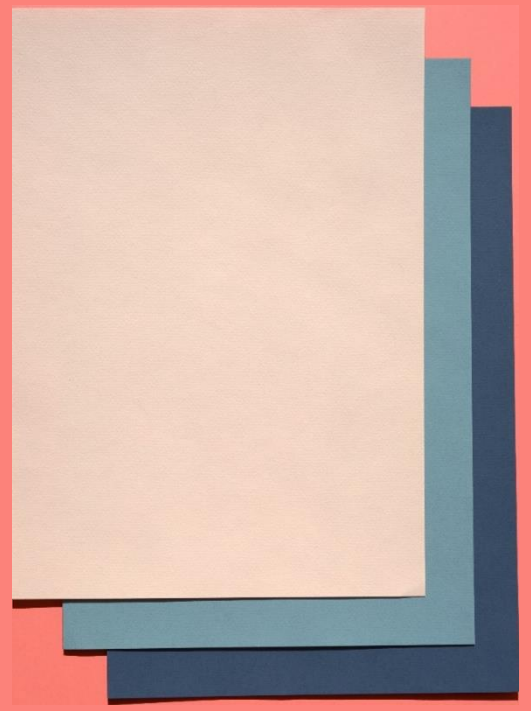
워드클라우드와 레이더차트를 이용한 대선공약 시각화

시각화 목표

01 전체 공약의 분야별 워드클라우드

02 4명의 주요후보의 공약 워드클라우드

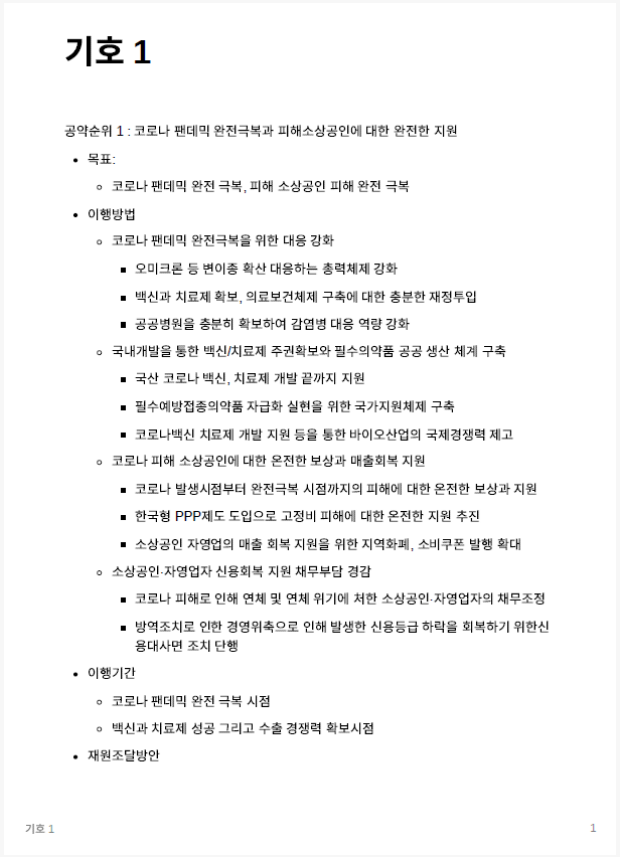
03 3명의 주요후보의 공약레이더 차트



사용한 라이브러리

```
# 전처리
import pdfplumber
import numpy as np
import pandas as pd
from konlpy.tag import Okt
from ckonlpy.tag import Twitter
from pykospacing import Spacing
from collections import Counter

# 크롤링
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException,StaleElementReferenceException
import time

# 워드클라우드
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns

# 레이더차트, Count, TF_IDF Vectorazizer
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from math import log
from math import pi
from matplotlib import rc, font_manager

# 이미지 처리
from PIL import Image, ImageDraw, ImageFont

# 기타
import warnings
warnings.filterwarnings('ignore')
```

사용한 데이터



기호 1

공약순위 1 : 코로나 팬데믹 완전극복과 피해소상공인에 대한 완전한 지원

- 목표:
  - 코로나 팬데믹 완전 극복, 피해 소상공인 피해 완전 극복
- 이행방법
  - 코로나 팬데믹 완전극복을 위한 대응 강화
    - 오미크론 등 변이종 확산 대응하는 총력체제 강화
    - 백신과 치료제 확보, 의료보건체제 구축에 대한 충분한 재정투입
    - 공공병원을 충분히 확보하여 감염병 대응 역량 강화
  - 국내개발을 통한 백신/치료제 주권확보와 필수의약품 공공 생산 체계 구축
    - 국산 코로나 백신, 치료제 개발 끝까지 지원
    - 필수예방접종의약품 자급화 실현을 위한 국가지원체제 구축
    - 코로나백신 치료제 개발 지원 등을 통한 바이오산업의 국제경쟁력 제고
  - 코로나 피해 소상공인에 대한 온전한 보상과 매출회복 지원
    - 코로나 발생시점부터 완전극복 시점까지의 피해에 대한 온전한 보상과 지원
    - 한국형 PPP제도 도입으로 고정비 피해에 대한 온전한 지원 추진
    - 소상공인 자영업의 매출 회복 지원을 위한 지역화폐, 소비쿠폰 발행 확대
  - 소상공인·자영업자 신용회복 지원 채무부담 경감
    - 코로나 피해로 인해 연체 및 연체 위기에 처한 소상공인·자영업자의 채무조정
    - 방역조치로 인한 경영위축으로 인해 발생한 신용등급 하락을 회복하기 위한신용대사면 조치 단행
- 이행기간
  - 코로나 팬데믹 완전 극복 시점
  - 백신과 치료제 성공 그리고 수출 경쟁력 확보시점
- 재원조달방안

기호 1 1

2022년 3월 2일 기준 수집한 대선 후보들의 공약 PDF 파일

A	B	C	D	E
	1번후보	2번후보	3번후보	4번후보
0	보건의료/환경, 재정/경제/복지	재정/경제/복지	보건의료/환경	재정/경제/복지, 과학기술/정보통신
1	재정/경제/복지, 산업자원/건설교통	재정/경제/복지	교육/인적자원, 산업자원/건설교통	국방/통일/외교통상
2	재정/경제/복지, 교육/인적자원	재정/경제/복지	교육/인적자원	재정/경제/복지
3	재정/경제/복지	정치/행정/사법	재정/경제/복지, 산업자원/건설교통	보건의료/환경, 재정/경제/복지
4	재정/경제/복지	과학기술/정보통신	재정/경제/복지	재정/경제/복지
5	교육/인적자원	재정/경제/복지	보건의료/환경, 재정/경제/복지	교육/인적자원

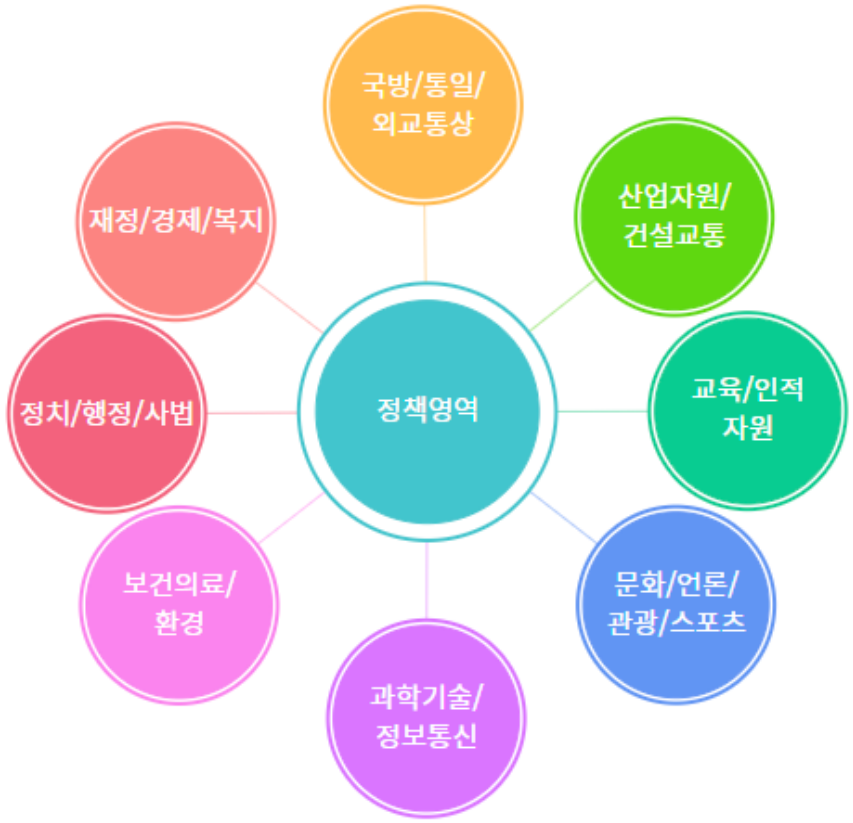
크롤링을 통해 수집한 공약별 카테고리 CSV 파일

PART1

크롤링

PART 1

크롤링



중앙선거관리위원회 정책·공약마당의
공약이슈트리를 따라 대선후보들의 공약을
분류하기 위해 우선 크롤링을 통해
대선후보들의 공약 카테고리 데이터를 수집

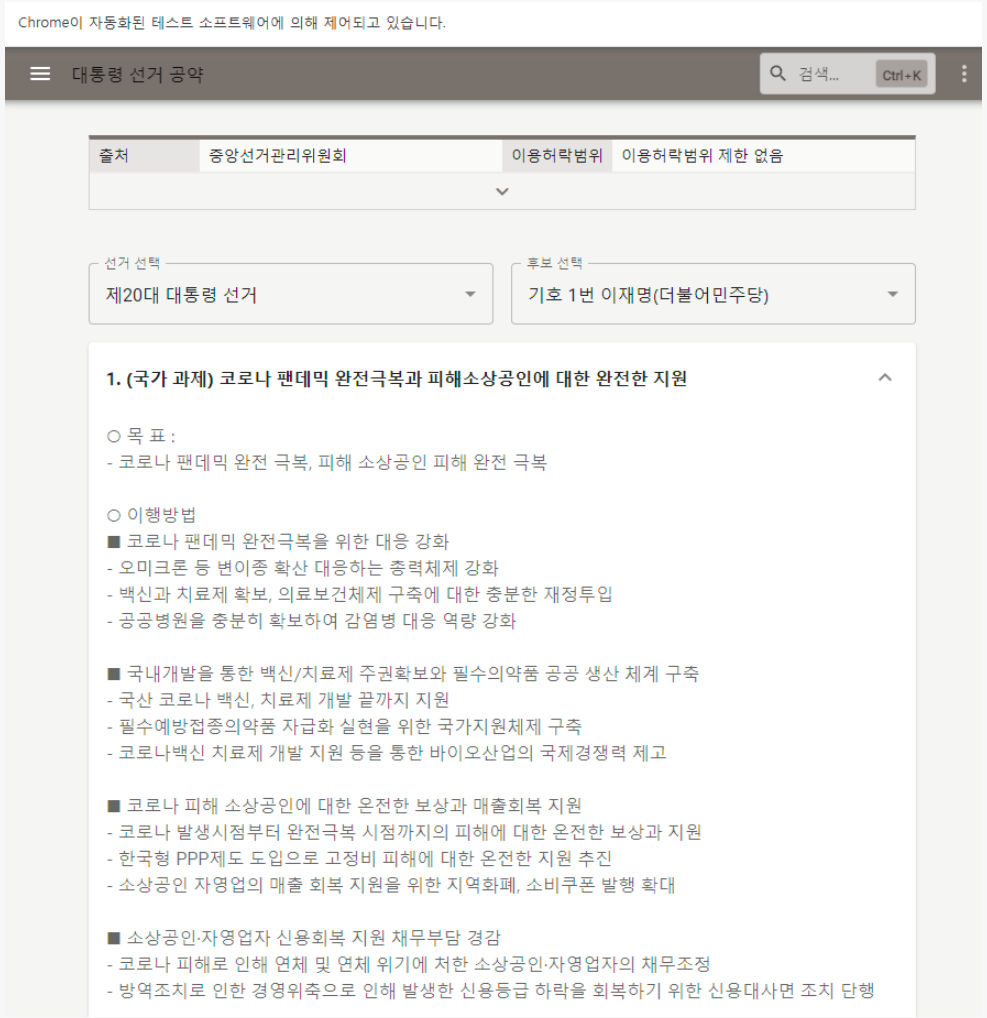
크롤링

```
name_list = []
for i in range(1, 15):
    name_list.append(f'기호 {i}번')

for i in range(1, 15):
    globals()[f'list_{i}'] = []

url = 'https://www.dploor.com/election/20/president'
driver = webdriver.Chrome('chromedriver.exe')
def chrome_get(k, l):
    driver.get(url)
    time.sleep(0.01)

    write = driver.find_element_by_xpath('//*[@id=":R2kudel6:"]')
    write.send_keys(k)
    write.send_keys(Keys.ARROW_DOWN)
    write.send_keys(Keys.RETURN)

    clk = driver.find_elements_by_xpath('//*[@id="__next"]/div[2]/div/main/div[2]/div[2]/div/div/div/div/button[1]').clear
    time.sleep(0.7)

    for i in range(1, 11):
        try:
            l.append(driver.find_element_by_xpath('//*[@id="__next"]/div[2]/div/main/div[3]/div[{0}]/div[1]/div[1]/p'.format(i)).text)
        except NoSuchElementException as e:
            print(e)
            l.append(driver.find_element_by_xpath('//*[@id="__next"]/div[2]/div/main/div[4]/div[{0}]/div[1]/div[1]/p'.format(i)).text)
```

ChromeDriver, Selenium을 이용한 동적 크롤링을 통해 후보들의 공약별 카테고리 데이터 수집

PART 1

크롤링




```python
for i in range(1, 15):
    globals()[f'list_{i}'] = []

get_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
for i in get_list:
    chrome_get(i, globals()[f'list_{i}'])
```

카테고리가 포함되어 있는
공약 데이터를 수집하여
리스트에 저장

PART2

전처리

전처리-PDF파일

```
for i in range(1, 15):
    globals()[f'path_{i}'] = f'open/기호_{i} 공약.pdf'
```

```
no_list = []
for i in range(1, 15):
    no_list.append(globals()[f'path_{i}'])

def pre_pro_pdf(temp): # pdf 불러오는 함수
    for i in range(len(temp.pages)):
        globals()[f'a_{i}_{i}'] = globals()[f'a_{i}'].rfind('\n') + 1
        globals()[f'a_{i}'] = globals()[f'a_{i}'][:globals()[f'a_{i}_{i}']]
    globals()[f'a_0'] = globals()[f'a_0'][globals()[f'a_0'].find('\n',3)+1:]
    for i in range(1, len(temp.pages)):
        globals()[f'a_0'] += globals()[f'a_{i}']
    return a_0

for i,j in zip(no_list, range(1, 15)):
    with pdfplumber.open(i) as temp:
        for k in range(len(temp.pages)):
            globals()[f'{k}_page'] = temp.pages[k]
            globals()[f'a_{k}'] = globals()[f'{k}_page'].extract_text()
    globals()[f'no_{j}'] = pre_pro_pdf(temp) # no_1 ~ no_14 까지 각 후보들의 공약임
```

주어진 자료의 형식이 PDF 형식이기 때문에

PDF에서 text를 추출하여 시각화 진행

전처리-PDF파일

```python
for i in range(1, 15):
    globals()[f'no_{i}'] = globals()[f'no_{i}'].split('공약순위')
```

```python
for i in range(1, 15):
    del globals()[f'no_{i}'][0]
df = pd.DataFrame()

for i in range(1, 15):
    df[f'{i}번후보'] = globals()[f'no_{i}']

def del_space(df):
    cand_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13]
    for i in cand_list:
        for j in range(len(df)):
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('이행 방법', '이행방법')
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('이행 기간', '이행기간')
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('목 표', '목표')
    return df
def change_contents(df):
    cand_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13]
    for i in cand_list:
        for j in range(len(df)):
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j][:df[f'{i}번후보'][j].find('목표')] + \
            df[f'{i}번후보'][j][df[f'{i}번후보'][j].find('이행방법'):df[f'{i}번후보'][j].find('이행기간')]
    return df

del_space(df)
change_contents(df)
```

각 후보의 통합되어 있는 1~10번 공약을 공약 순위 기준으로 나누어 배열에 저장

각 후보별 용어 띄어쓰기가 다른 문제를 replace를 통해 용어 통합

후보들의 공약에서 분석에 불필요한 ‘이행기간’, ‘재원조달방안’ 제거

전처리-PDF파일

```python
def del_first_num(df):
    cand_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
    for i in cand_list:
        for j in range(len(df)):
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j][3:]
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j].strip(':')
            df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('\n', ' ')
    return df

del_first_num(df)
```

```python
cand_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
for i in cand_list:
    df[f'{i}번후보'][9] = df[f'{i}번후보'][9].strip(' :')
```

```python
for i in cand_list:
    for j in range(len(df)):
        df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('이행방법', ' ')
```

분석에 필요없는 공약순위의

숫자, 이스케이프문자, 이행방법 텍스트 제거

전처리-PDF파일

```
for i in cand_list:
    for j in range(len(df)):
        df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace(' ', '')
```

```
spacing = Spacing()
```

```
for i in cand_list:
    for j in range(len(df)):
        df[f'{i}번후보'][j] = spacing(df[f'{i}번후보'][j])

for i in cand_list:
    for j in range(len(df)):
        df[f'{i}번후보'][j] = df[f'{i}번후보'][j].replace('•', '·')
```

```
df_1 = pd.DataFrame()
df_1 = df.copy()
```

PDF 불러오는 과정에서
줄바꿈으로 인한
띄어쓰기 오류 존재

모든 띄어쓰기를 제거 후
PyKoSpacing 라이브러리를 사용해
다시 띄어쓰기를 하는 과정을 진행

전처리-PDF파일

```
twitter.add_dictionary(['포스트코로나','긴급구조','긴급구조특별본부','가동','감염병','건강보험법','시행령','지속가능','창출', ...],'Noun')
stop_words = '''
위
과
·
/
고
,
및
.
-
(
)
.
을
'
'
...

for i in range (1,15):
    df[f'{i}번후보'] = df[f'{i}번후보'].apply(twitter.nouns)


for i in cand_list:
    for j in range(len(df)):
        df[f'{i}번후보'][j] = [word for word in df[f'{i}번후보'][j] if not word in stop_words]
```

단어사전에 등록되어 있지 않은
단어의 경우 제대로 토큰화
되지않는 오류 존재

623개의 단어를 dictionary에 추가
72개의 단어를 stop_words에 추가

이후 토큰화 및 불용어 제거

전처리- 크롤링데이터

```
for i in range(1, 15):
    globals()[f'list_{i}'] = []

get_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]
for i in get_list:
    chrome_get(i, globals()[f'list_{i}'])

for i in range(1, 15):
    for j in range(0, 10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j][globals()[f'list_{i}'][j].find('(')+1:globals()[f'list_{i}'][j].rfind(')')]

globals()[f'list_{6}'][2] = globals()[f'list_{6}'][2][:globals()[f'list_{6}'][2].rfind(')')]
globals()[f'list_{6}'][3] = globals()[f'list_{6}'][3][:globals()[f'list_{6}'][3].rfind(')')]
globals()[f'list_{6}'][9] = globals()[f'list_{6}'][9][:globals()[f'list_{6}'][9].rfind(')')]
globals()[f'list_{10}'][8] = globals()[f'list_{10}'][8][:globals()[f'list_{10}'][8].rfind(')')]
globals()[f'list_{14}'][7] = globals()[f'list_{14}'][7][:globals()[f'list_{14}'][7].rfind(')')]

for i in range(10):
    list_12[i] = list_12[i][list_12[i].find('(')+1:list_12[i].rfind(')')]

list_12[6] = list_12[6].replace(')(', ', ')

list_14[7] = list_14[7].replace('재검', '재정')
```

크롤링된 데이터에서
( ) 괄호 안에 있는
데이터만 필요
find를 통해 카테고리 추출

특정 후보의 경우
공약 제목에 괄호가
여러개 존재

필요한 카테고리 이름만
추출 할 수 있도록 처리

전처리- 크롤링데이터

```
for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('노동', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('여성', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('인권', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('선거공영제', '정치')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('판결문 공개', '사법')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('백신접종', '보건의료')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('국가 과제', '보건의료, 경제')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('부동산, 균형발전', '경제')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('노동, 일자리', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('청년', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('선거', '행정')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('일자리', '인적자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('안전', '복지')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('산업', '산업자원')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('외교', '외교')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('사업', '사법')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('지방자치', '행정')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('사회', '경제')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('통상', '외교')

for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace(' ', '')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('·', '')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace(',', '')
```

후보별 카테고리 정의가
공약이슈트리와 동일하지않음

각 후보가 정한 임의의 카테고리를
공약 이슈트리와 동일하게 변경

불필요한 특수문자 제거

전처리- 크롤링데이터

```
for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('환경', '1')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('보건의료', '1')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('정보통신', '2')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('과학기술', '2')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('스포츠', '3')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('관광', '3')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('언론', '3')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('문화', '3')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('인적자원', '4')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('교육', '4')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('건설교통', '5')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('산업자원', '5')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('외교', '6')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('통상', '6')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('통일', '6')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('국방', '6')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('복지', '7')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('경제', '7')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('재정', '7')

        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('사법', '8')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('행정', '8')
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('정치', '8')

for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('자원', '')
```

공약별로 1~8 인코딩

환경 > 보건의료/환경 로 변경시
‘환경/환경/보건의료’ 로
중복되어 수정되는 문제 발생

숫자로 인코딩 해준뒤
다시 디코딩 해주는 방식으로 진행

ex) 1 > 보건의료/환경

전처리- 크롤링데이터

```
for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = set(globals()[f'list_{i}'][j])

cat_list = ['보건의료/환경', '과학기술/정보통신', '문화/언론/관광/스포츠', '교육/인적자원', '산업자원/건설교통', '국방/통일/외교통상', '재정/경제/복지', '정치/행정/사법']

for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = str(globals()[f'list_{i}'][j]).strip('{',).strip('}',).strip('\'').replace('\'', '')

for i in range(1, 15):
    print(globals()[f'list_{i}'])
```

```
['7, 1', '7, 5', '7, 4', '7', '7', '4', '2, 4', '3, 2', '8', '6']
['7', '7', '7', '8', '2', '7', '8, 4', '6', '1, 5', '3, 4']
['1', '5, 4', '4', '7, 5', '7', '7, 1', '3, 4', '8, 7', '8', '6']
['7, 2', '6', '7', '7, 1', '7', '4', '7', '7', '8', '7, 1']
['7', '7', '8', '8', '1', '7', '2', '4', '1', '7']
['7', '7, 8', '8', '6', '4', '8, 5', '7', '5', '7', '5']
['7', '4', '1', '4', '7', '7', '4', '4', '6', '8']
['8', '8', '8', '8', '8', '8', '8', '8', '1', '8']
['8', '8', '8', '7', '7', '6', '5', '4', '4', '2']
['6', '3', '8', '5', '4', '7', '7', '8', '8', '5']
['8', '7', '7', '1', '8', '5', '4', '4', '8', '8']
['4', '4', '4', '4', '5', '1', '1, 4', '7', '8', '6']
['7', '7', '7, 1, 2', '7', '7, 4', '6, 4', '6', '6', '6', '6']
['7, 3, 6', '8', '8', '8, 4', '8, 6, 7', '8, 4', '7, 8, 4', '7', '8', '7, 1']
```

정상적으로 인코딩 되었는지

출력을 통한 확인

전처리- 크롤링데이터

```
for i in range(1, 15):
    for j in range(10):
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('1', cat_list[0])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('2', cat_list[1])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('3', cat_list[2])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('4', cat_list[3])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('5', cat_list[4])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('6', cat_list[5])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('7', cat_list[6])
        globals()[f'list_{i}'][j] = globals()[f'list_{i}'][j].replace('8', cat_list[7])

cat_df = pd.DataFrame()

for i in range(1, 15):
    cat_df[f'{i}번후보'] = globals()[f'list_{i}']

cat_df.to_csv('공약별_카테고리.csv', encoding='utf-8-sig')
```

```
col_list = []
for i in range(1, 15):
    col_list.append(i)

col_list
cat_df.columns = col_list
df.columns = col_list
```

디코딩을 통해

후보별 정책 카테고리

통합 후 CSV 파일로 저장

전처리- 통합데이터

```python
def get_contents(cl, cat):
    for i in range(1, 15):
        for j in range(10):
            if cat in cat_df[i][j]:
                cl.append((i,j))
    return cl
```

```python
def get_contents_2(df, l, l2):
    for i in range(len(l)):
        l2.append(df[l[i][0]][l[i][1]])
    return l2

for i in range(1, 9):
    globals()[f'cat_list_{i}'] = []

for i in range(8):
    get_contents(globals()[f'cat_list_{i+1}'], cat_list[i])

for i in range(1, 9):
    globals()[f'word_list_{i}'] = []

for i in range(8):
    globals()[f'word_list_{i+1}'] = get_contents_2(df, globals()[f'cat_list_{i+1}'], globals()[f'word_list_{i+1}'])

for i in range(8):
    globals()[f'word_list_{i+1}'] = sum(globals()[f'word_list_{i+1}'], [])

for i in range(8):
    globals()[f'common_list_{i+1}'] = Counter(globals()[f'word_list_{i+1}']).most_common(75)
```

전처리 완료된 각 후보들의 공약을 크롤링을 통해 수집한 카테고리 데이터와 조합하여 각 후보별 카테고리 토큰 저장

이후 분야별로 빈도수가 가장 높은 75개 단어를 이용해 워드클라우드 생성

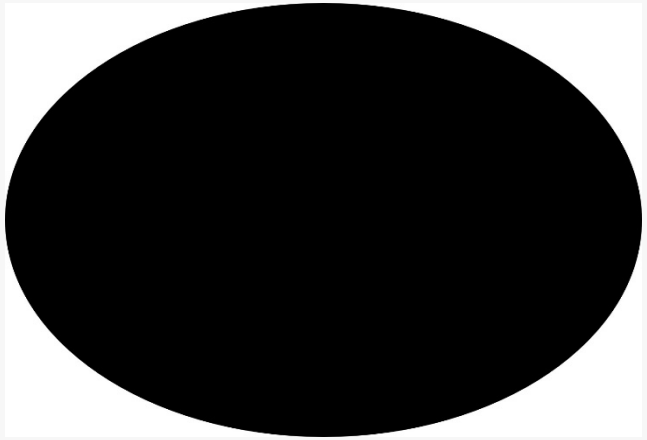
PART3

워드클라우드

워드클라우드

```
circle_jpg = Image.open('black_circle.jpg')
circle_jpg = circle_jpg.resize((560,400))
img_array = np.array(circle_jpg)

for i in range(1, 5): # 1~4 숫자 이미지 가져오기
    globals()[f'img_{i}'] = Image.open(f'img_{i}.png')
    globals()[f'img_{i}'] = globals()[f'img_{i}'].resize((800, 1120))
    globals()[f'img_{i}_array'] = np.array(globals()[f'img_{i}'])
```

워드클라우드의 모양을 설정하기 위해
1~4까지의 숫자와 타원형의 그림을 배열로 저장



워드클라우드

```
def make_wc(l, color):
    wc = WordCloud(font_path='malgunbd.ttf',background_color="grey", random_state=1,
                   max_font_size=120, mask = img_array, prefer_horizontal = 1, colormap=color)
    cloud = wc.generate_from_frequencies(dict(l))
    plt.figure(figsize=(10, 8))
    plt.axis('off')
    plt.imshow(cloud)
    plt.show()
    return cloud

color_list = ['Greens','Reds', 'Blues', 'Oranges', 'Purples', 'PuRd', 'YlGn', 'YlOrRd' ]

for i in range(1, 9):
    globals()[f'wc_{i}'] = make_wc(globals()[f'common_list_{i}'], color_list[i-1])
    globals()[f'wc_{i}'].to_file(f'wc_{i}.jpg')
```

seaborn의 color_palette에 있는 색배열로 분야별 카테고리

워드클라우드를 생성 후 이미지 파일로 저장

PART 3

워드클라우드

생성된 워드클라우드

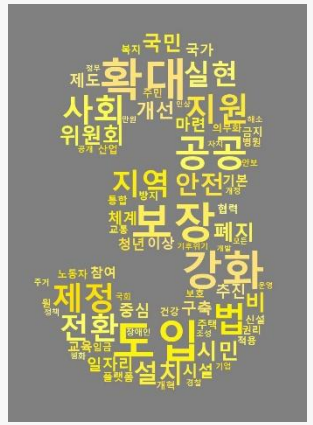
워드클라우드

```
for i in range(1, 5):
    globals()[f'cand_list_{i}'] = []
    globals()[f'cand_list_{i}'] = sum(df[i], [])
    globals()[f'cand_list_{i}'] = Counter(globals()[f'cand_list_{i}']).most_common(75)

def make_wc_num(l,array):
    wc = WordCloud(font_path='malgunbd.ttf',background_color="grey", random_state =1,
                   max_font_size=120, mask = array, prefer_horizontal = 1)
    cloud = wc.generate_from_frequencies(dict(l))
    recolor = cloud.recolor(color_func=make_colors, random_state=True) #추가

    plt.figure(figsize=(10, 8))
    plt.axis('off')
    plt.imshow(recolor, interpolation='bilinear') #추가
    plt.show()
    return cloud

def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = random_state.randint(100,255)
    g = random_state.randint(0,50)
    b = random_state.randint(0,50)

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color
wc_cand_1 = make_wc_num(cand_list_1, img_1_array)
```

1번 ~ 4번 후보의 공약을 토큰화한
데이터를 기반으로 빈도수가 가장 많은 75개의
단어를 이용해각 후보 번호의 모양과
소속 정당 색상의 워드클라우드 생성

워드클라우드

```
def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = random_state.randint(100,255)
    g = random_state.randint(0,50)
    b = random_state.randint(0,50)

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color
wc_cand_2 = make_wc_num(cand_list_2, img_2_array)

def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = random_state.randint(255,255)
    g = random_state.randint(225, 255)
    b = random_state.randint(0,130)

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color
wc_cand_3 = make_wc_num(cand_list_3, img_3_array)

def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = random_state.randint(255,255)
    g = random_state.randint(70, 170)
    b = random_state.randint(0,110)

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color
wc_cand_4 = make_wc_num(cand_list_4, img_4_array)
```




```
for i in range(1, 5):
    globals()[f'wc_cand_{i}'].to_file(f'wc_cand_{i}.jpg')
```

1번 ~ 4번후보 워드클라우드를

이미지 파일로 저장

PART4

레이더차트

레이더차트

```
font_path = "C:/Windows/Fonts/NGULIM.TTF"
font = font_manager.FontProperties(fname=font_path).get_name()

rc('font', family=font)
plt.rcParams['axes.unicode_minus'] = False


def string_conversion(cond):
    cond = list(df['{0}번후보'.format(i)])
    cond = str(cond)
    cond = cond.replace(', ', "' '")
    cond = cond.replace("' '", "' '")
    cond = cond.replace('[', '')
    cond = cond.replace(']', '')
    cond = cond.replace("'", '')

    return cond

for i in range(1,len(df.columns)+1):
    globals()[f'cond_{i}'] = df['{0}번후보'.format(i)]

    globals()[f'cond_{i}'] = string_conversion(df['{0}번후보'.format(i)])
```

시각화를 진행할 때 한글을
표기하기 위해 font 설정
디코딩 된 각 후보별 데이터 프레임을
list형태의 string으로 변환
특수기호를 제거하는 전처리 실행

레이더차트

```python
def doc(*args):
    doc_list = []
    tf = pd.DataFrame()
    idf = pd.DataFrame()
    tf_idf = pd.DataFrame({'재정/경제/복지':[0], '정치/행정/사법':[0], '보건의료/환경':[0], '과학기술/정보통신':[0],
                           '문화/언론/관광/스포츠':[0], '교육/인적자원':[0], '산업자원/건설교통':[0],
                           '국방/통일/외교통상':[0]})

    # 단어 리스트 생성
    for i in args:
        # 단어 분해
        tmp_list = i.split(' ')
        # 리스트 결합
        doc_list += tmp_list
    doc_list = list(set(doc_list))

    # DF
    df = []
    for i in doc_list:
        tmp = 0
        for j in args:
            # 단어 분해
            tmp_list = list(set(j.split(' ')))
            if i in tmp_list:
                tmp += 1
        df.append(tmp)

    # TF(DTM), IDF, TF-IDF
    for i in range(len(doc_list)):
        tmp = []
        tmp2 = []
        tmp3 = []
        for j in args:
            # 단어 분해
            tmp_list = j.split(' ')
            # 단어 세기
            tmp.append(tmp_list.count(doc_list[i]))
            tmp2.append(log(len(args) / (df[i] + 1)))
            tmp3.append((tmp_list.count(doc_list[i])) * (log(len(args) / (df[i] + 1))))

        # 데이터 프레임 추가
        tf[doc_list[i]] = tmp
        idf[doc_list[i]] = tmp2
        tf_idf[doc_list[i]] = tmp3

    return tf_idf
```

데이터 프레임을 생성하고

각 공약별 8구분에 대해 컬럼 생성

각 단어 중 8개의 구분된 공약 기준

단어를 분해하여 DF

다시 단어를 분해하여 카운트 후 TF

DF와 TF를 곱하여 TF_IDF

레이더차트

```
tf_idf = pd.DataFrame({'재정/경제/복지':[0], '정치/행정/사법':[0], '보건의료/환경':[0], '과학기술/정보통신':[0],
                       '문화/언론/관광/스포츠':[0], '교육/인적자원':[0], '산업자원/건설교통':[0],
                       '국방/통일/외교통상':[0]})
tf_idf.drop(0,axis=0,inplace=True)

for i in range(1,15):
    tf_idf = tf_idf.append(doc(globals()[f'cond_{i}']))

tf_idf = abs(tf_idf).round(2)
tf_idf = tf_idf+1

tf_idf['cond'] = ['1번후보','2번후보','3번후보','4번후보','5번후보','6번후보','7번후보','8번후보','9번후보','10번후보','11번후보','12번후보','13번후보','14번후보']
tf_idf = tf_idf[['cond','재정/경제/복지','정치/행정/사법', '보건의료/환경', '과학기술/정보통신', '문화/언론/관광/스포츠',
                 '교육/인적자원', '산업자원/건설교통', '국방/통일/외교통상']]
```

앞의 함수로 TF_IDF 진행

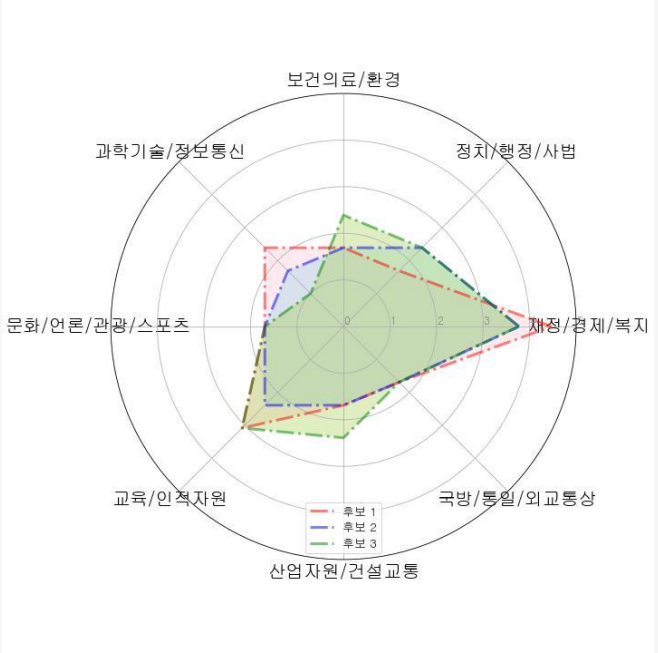
출력값들을 해당 데이터프레임의 값으로 삽입 후

각 컬럼의 이름 및 후보(cond)를 생성하여 데이터 프레임에 적용

PART 4

레이더차트

```python
def plot_radar_chart(df, group_id):
    # 변수 수
    cls_df_sel = df.drop(group_id, axis=1).copy()
    categories=list(cls_df_sel)
    N = len(categories)

    # value 계산
    cls_df_sel = cls_df_sel[cls_df_sel.columns.tolist() + [cls_df_sel.columns.tolist()[0]]].copy()
    values = cls_df_sel.values

    # 변수의 수에 따른 angle 계산
    angles = [n / float(N) * 2 * pi for n in range(N)]
    angles += angles[:1]

    # 그래프 창
    plt.figure(figsize=(10,10))
    ax = plt.subplot(111, polar=True)

    # x축 라벨링
    plt.xticks(angles[:-1], categories, color='black', size=20)

    #y축 라벨링
    ax.set_rlabel_position(0)
    plt.yticks([0, 1, 2, 3, 4, 5], ["0","1","2","3","4","5"], color="grey", size=12)
    plt.ylim(0,5)

    # 데이터 plotting
    color_ = ['green','blue','red']
    color_2 = ['yellowgreen', 'skyblue', 'pink']
    for cl_n in [2, 0, 1]:
        ax.plot(angles, values[cl_n], linewidth=3, linestyle='dashdot',color= color_[cl_n], label='후보 '+str(cl_n+1), alpha=0.5)
        ax.fill(angles, values[cl_n],color_2[cl_n], alpha=0.3)
    plt.legend(loc='lower center', fontsize=13)
    plt.tight_layout()
    plt.savefig('radar_chart.jpg')
    plt.show()

    return

r_chart_1 = plot_radar_chart(df[:4],'cond')
```



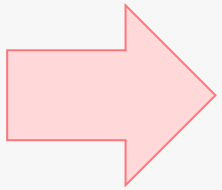
위의 진행된 TF_IDF 값을 이용하여
레이더 차트를 시각화
이후 이미지 파일로 저장

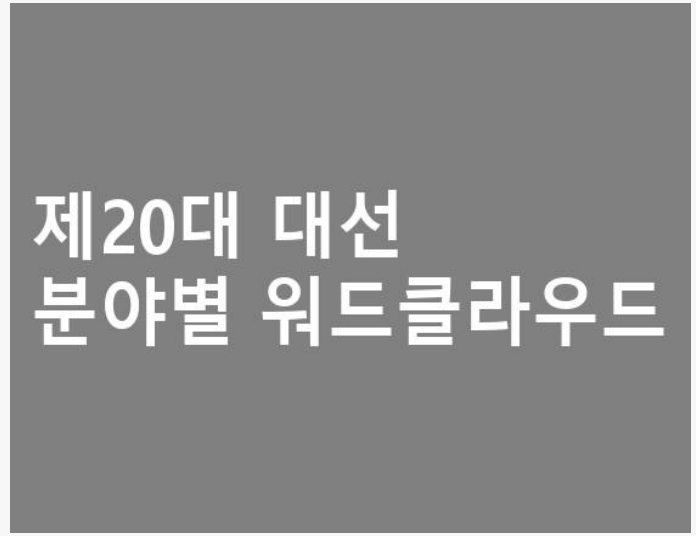
PART5

이미지처리

이미지처리

```
def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = 255
    g = 255
    b = 255

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color

def make_wc(l):
    wc = WordCloud(font_path='malgunbd.ttf',background_color="grey", random_state =1,
                   max_font_size=180, mask = img_array, prefer_horizontal = 1,
                  color_func= make_colors)
    cloud = wc.generate_from_frequencies(dict(l))
    plt.figure(figsize=(10, 8))
    plt.axis('off')
    plt.imshow(cloud)
    plt.show()
    return cloud

text_20 = [('''제20대 대선
분야별 워드클라우드''', 1)]

text_20_wc = make_wc(text_20)
```

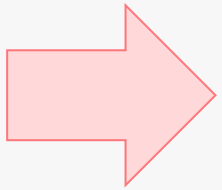




분야별 워드클라우드의 제목을 만들어
이미지 파일로 저장

이미지처리

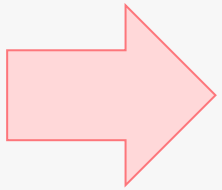
```
def make_colors(word, font_size, position, orientation, random_state, **kwargs):
    r = 255
    g = 255
    b = 255

    color = "rgb(%d, %d, %d)" % (r, g, b)

    return color

def make_wc(l):
    wc = WordCloud(font_path='malgunbd.ttf',background_color="grey", random_state =5,
                   max_font_size=180, mask = img_array, prefer_horizontal = 1,
                  color_func=make_colors)
    cloud = wc.generate_from_frequencies(dict(l))
    plt.figure(figsize=(10, 8))
    plt.axis('off')
    plt.imshow(cloud)
    plt.show()
    return cloud

text_4_cand = [('''4대후보
대선 공약
워드클라우드''', 1)]

text_4_cand_wc = make_wc(text_4_cand)
```



1번 ~ 4번후보의 워드클라우드의 제목을 만들어 이미지 파일로 저장

이미지처리

```
def make_wc(l):
    wc = WordCloud(font_path='malgunbd.ttf',background_color="grey", random_state =5,
                   max_font_size=180, mask = img_array, prefer_horizontal = 1,
                  color_func=make_colors)
    cloud = wc.generate_from_frequencies(dict(l))
    plt.figure(figsize=(10, 8))
    plt.axis('off')
    plt.imshow(cloud)
    plt.show()
    return cloud

text_rc = [('''주요후보 레이더 차트''', 1)]

text_rc_wc = make_wc(text_rc)
```



1번 ~ 3번 후보의 레이더 차트의
제목을 만들어 이미지 파일로 저장

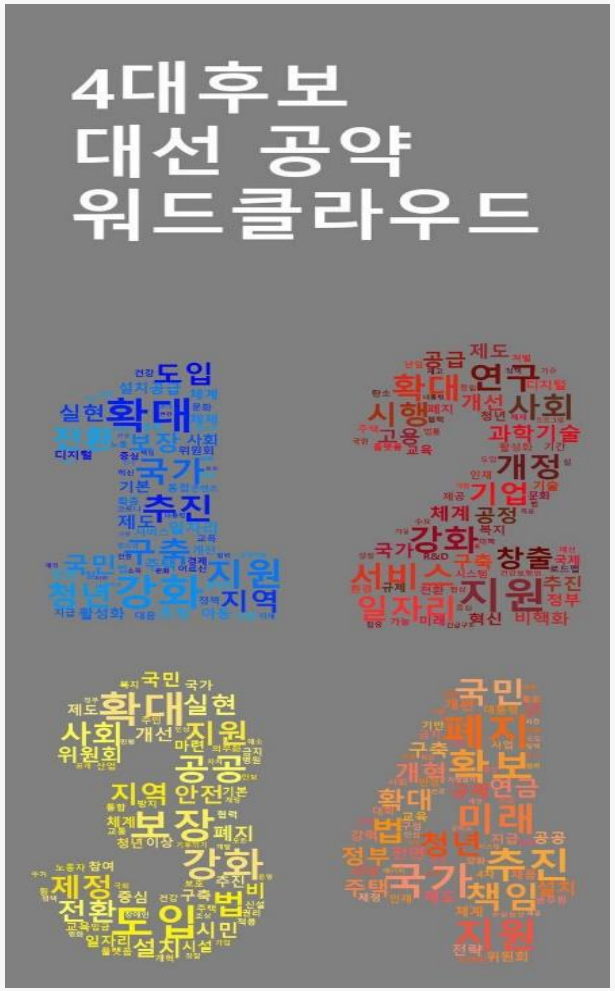
이미지처리

```
cat_list = ['<보건의료/환경>', '<과학기술/정보통신>', '<문화/언론/관광/스포츠>',
            '<교육/인적자원>', '<산업자원/건설교통>', '<국방/통일/외교통상>', '<재정/경제/복지>', '<정치/행정/사법>']

for i in range(1, 9):
    globals()[f'img_{i}'] = Image.open(f'wc_{i}.jpg')
    globals()[f'img_{i}'] = globals()[f'img_{i}'].resize((560, 400))

def set_name(img, txt):
    new_img = Image.new(size = (560, 500), color='grey', mode = 'RGB')
    new_img.paste(img)
    draw=ImageDraw.Draw(new_img)
    draw.text(((560 - len(txt) * 25)/2, 430), txt, font=ImageFont.truetype("malgunbd.ttf", 30), fill=(255,255,255))
    return new_img

for i in range(0, 8):
    globals()[f'img_{i+1}'] = set_name(globals()[f'img_{i+1}'], cat_list[i])

img_text_20 = Image.open('text_20.jpg')

img_text_20 = img_text_20.resize((560,500))

new_img = Image.new("RGB", (1880, 1500), color='grey')

new_img.paste(img_1)
new_img.paste(img_2, (img_1.width + 100, 0))
new_img.paste(img_3, (img_1.width*2 + 200, 0))
new_img.paste(img_4, (0, img_1.height))
new_img.paste(img_5, (img_1.width*2 + 200, img_1.height))
new_img.paste(img_6, (0, img_1.height*2))
new_img.paste(img_7, (img_1.width + 100, img_1.height*2))
new_img.paste(img_8, (img_1.width*2+200, img_1.height*2))
new_img.paste(img_text_20, (img_1.width+100, img_1.height))

wordcloud_by_category = new_img
```

분야별 공약 워드클라우드와 제목을 좌표를 설정하여 붙여 하나의 이미지 파일 생성

이미지처리

```
new_img_2 = Image.new("RGB", (800, 1620), 10000) # 워드클라우드 400 * 560 4개, 제목 800*500

for i in range(1, 5):
    globals()[f'cand_{i}'] = Image.open(f'wc_cand_{i}.jpg')
    globals()[f'cand_{i}'] = globals()[f'cand_{i}'].resize((400, 560))

img_text_4대 = Image.open('text_4_cand.jpg').resize((800, 500))

new_img_2.paste(img_text_4대)
new_img_2.paste(cand_1, ((0, img_text_4대.height)))
new_img_2.paste(cand_2, ((cand_1.width, img_text_4대.height)))
new_img_2.paste(cand_3, ((0, img_text_4대.height+cand_1.height)))
new_img_2.paste(cand_4, ((cand_1.width, img_text_4대.height+cand_1.height)))

wordclod_4cands = new_img_2
```



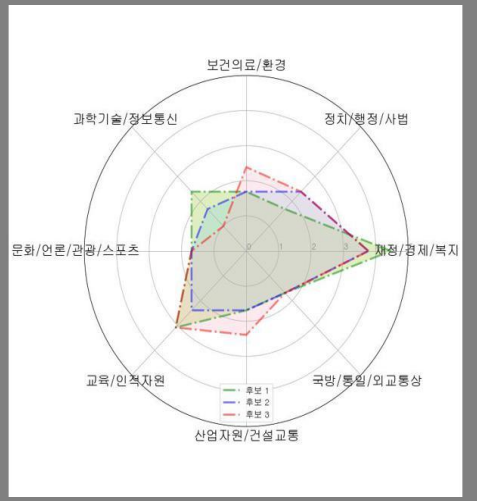
1번 ~ 4번 후보의 워드클라우드와
제목을 좌표를 설정하여 붙여 하나의
이미지 파일 생성

이미지처리

```
chart_img = Image.open('radar_chart.jpg')
chart_img = chart_img.resize((600, 600))
text_img = Image.open('text_rc.jpg')
text_img = text_img.resize((600, 400))
new_img_3 = Image.new(size = (800, 1200), color='grey', mode = 'RGB')
new_img_3.paste(text_img, (100, 0))
new_img_3.paste(chart_img, (100, 400))
new_img_3.save('radar.jpg')
radar_chart = Image.open('radar.jpg')
```

1번 ~ 3번 후보의 레이더차트와
제목을 좌표를 설정하여 붙여 하나의
이미지 파일 생성

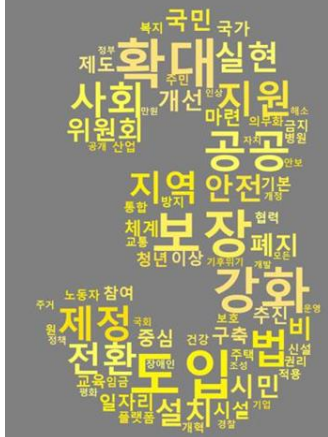
주요후보 레이더 차트



#워드클라우드

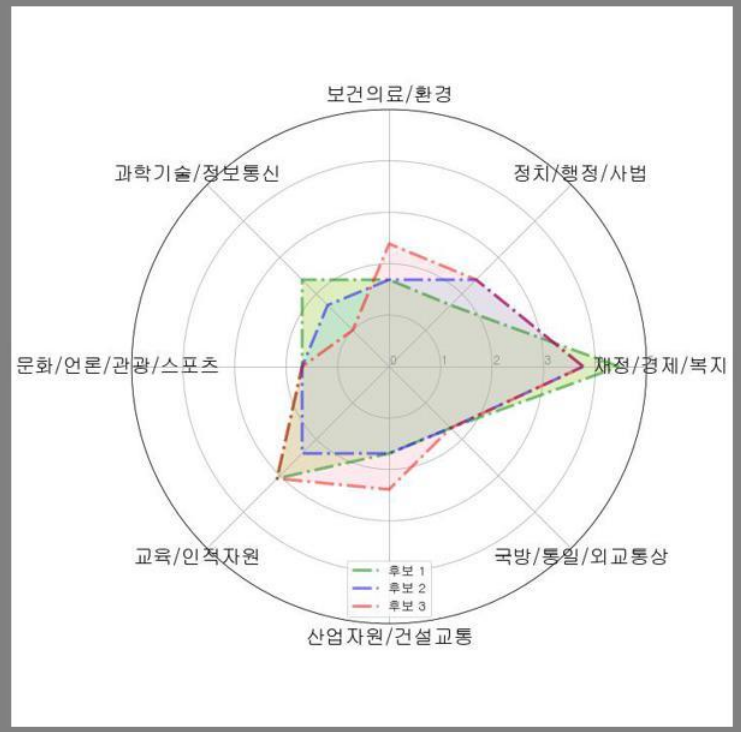

제20대 대선
분야별 워드클라우드
<보건의료/환경>
<과학기술/정보통신>
<문화/언론/관광/스포츠>
<교육/인적자원>
<산업자원/건설교통>
<국방/통일/외교통상>
<재정/경제/복지>
<정치/행정/사법>

4대후보
대선 공약
워드클라우드



#레이더차트

주요후보 레이더 차트


보건의료/환경
정치/행정/사법
재정/경제/복지
국방/통일/외교통상
산업자원/건설교통
교육/인적자원
문화/언론/관광/스포츠
과학기술/정보통신
0
1
2
3
후보 1
후보 2
후보 3

Thank You