정치커뮤니케이션 데이터분석실습(GCO2014) 과제 2

과제를 하시면서 참고할 사항

완성된 과제는 pdf 형태로 출력해서 iCampus에 제출해주세요.이 과제는 전체 성적의 10%를 차지하며, 과제의 마감은 **12월 8일 (수) 오후 11:59분**까지 입니다. 마감 시각 이후에 제출된 과제는 감점이 존재합니다 (강의계획서 참조).

1. 과제를 작성하시기 전에

- **꼭 본인의 구글 드라이브에 복사한 후 과제를 하세요.**
- 서명에 본인의 학번과 이름을 적어주세요.
- 코렙링크에 본인의 노트북 링크를 복사해주세요.

2. 과제를 작성하실 때

- 부교재나 다른 파이썬 교재를 참고하셔도 좋으나, 각 문제를 해결하기 위한 코드는 스스로 작성하시길 바랍니다.
- 코드를 작성하시면서 모든 코드에 주석을 다시길 바랍니다. 코드를 잘 이해하고 있는지를 보기 위해서 입니다 (아래의 예 참조)

```
a = 100 # a 변수에 100을 저장
a += 100 # a 변수에 100을 더한 결과를 a 변수에 저장
print(a) # a 변수의 값을 출력
```

3. 과제를 제출하실 때

저장이 완료되었습니다. ×

행 후 결과 값이 출력되어야 문제를 해결한 것으로 간주 하

- 노트북을 pdf 형태로 출력한 후 iCampus의 해당 과제물을 제출하셔야 과제를 제출한 것으로 인정됩니다.
- 과제 제출 기한을 넘기는 경우, 강의계획서에 공지한 것처럼 매 24시간 마다 1점씩 감점됩니다. `감점 = ((초과시간//24)+1)`

서명

저는 제 스스로 이 과제를 해결했으며, 제가 스스로 코드를 작성했습니다. 서명(예: 2020####23_김민철): 2021312108_송병도

코렙 링크

코랩링크: 본인의 코랩 노트북의 링크를 여기에 복사해 주세요.

[링크]https://colab.research.google.com/drive/1IxanW2c9eac0xJPWroHbjImTq96Jyca6?usp=sharing

데이터 셋

분석에 사용한 데이터를 구글 드라이브에 저장 후, 해당 드라이브를 공유해 주세요. **해당 드라이브의 데이터 내용을 확인할 수 있도록, 링크를 가진 모든 사람이 데이터를 볼 수 있는 권한을 주셔야 합니다.**

데이터셋 링크: 본인이 분석에 사용한 데이터 셋의 링크를 여기에 복사해주세요.

[링크] https://drive.google.com/file/d/1PijG5IT_2uxql6fzr_UrxLHEntvCTeU9/view?usp=sharing

---

과제 설명:

이 과제는 학생 여러분 스스로 관심있는 주제를 정해 수업 시간에 배운 실습 내용(**구글 API를 사용한 이미지 분류**)을 적용해 스스로 정치커뮤니케이션 분석을 실시하는 것을 목적으로 합니다. 실제 해결해야할 문제가 주어진 파이썬 실습 과제와는 다르게 학생 여러분 스스로 주제를 선택해 수업에서 다룬 내용을 활용해 분석을 실시하시면 됩니다.

가능하면 과제 전체의 내용을 파악한 후 과제를 시작해 주세요. 실습 과제는 다음 내용이 포함되어 있습니다. 각 부분에 포함되어야 하는 부분은 아래에 좀 더 자세히 기술되어 있습니다.

참조: 해당 단계마다 제시되는 예들은 단계별 가이드라인이며, 반드시 똑같은 분석을 실시해야하

저장이 완료되었습니다.

1. 문제제기 (해결하고자 하는 문제 기술)
2. 데이터에 대한 기술 (수집 방법/적합성) - 0.5점
3. 데이터 탐색적 분석 (데이터에 대한 기술) - 1점
4. 데이터 전처리 (전처리 과정 기술) - 2점
5. 토픽 모델 학습 (토픽 모델링 수행) - 2점
6. 토픽 레이블링 (토픽에 이름 붙이기) - 2점
7. 분석 결과 시각화 (결과의 시각화) - 1.5점
8. 분석 결과 논의 - 1점

---

1. 문제제기 및 예측

여기에서는 이미지 분석을 수행하고자 하는 문제를 기술해 보세요. 자신이 해당 문제를 풀기 전에 어떠

참조: 아래의 예를 참조해서 이미지 분석을 수행하기 위해 본인 관심이 있는 주제를 선정해 보세요.

예) 정파성에 따라 여당과 야당의 코로나 관련 뉴스 보도에 사용한 이미지를 분석 하고자 합니다. 2020년부터 3월 1일부터 2021년 10월 30일까년까지 조선일보와 한겨레에 신문이 게재한 뉴스 중 '코로나'관련 뉴스에 어떠한 이미지들이 사용되었는지를 분석하려고 합니다. 진보 성향을 가진 매체와 보수 성향을 지닌 매체들이 서로 강조하는 이미지가 다를 것으로 예측됩니다.

[문제제기 및 예측] 정파성에 따른 야당의 대선 후보인 이재명 후보에 대한 언론사별 보도에 사용한 이미지를 분석하고자 합니다. 2021년 9월1일부터 2021년 11월 30일까지의 기사들을 가지고 분석하였습니다. 각 언론사의 진보/보수 성향에 따라 강조하는 이미지가 다르게 나타날 것으로 예측됩니다.

2. 데이터 수집과정 및 데이터 적합성 (0.5점)

여기에는 분석을 위한 데이터 수집과정과 데이터가 왜 본인이 해결하고자 하는 문제에 적절한지에 대해

참고: 기존에 수업에서 수집한 이재명/윤석열 이미지 데이터의 일부를 사용하셔도 괜찮습니다. 이 경우 수업게시판에 공유된 데이터를 다운받아 압축을 푼 후 본인의 구글 드라이브에 저장하여 사용하시기 바랍니다.

- 웹스크랩핑을 통해 직접 데이터를 수집하시는 경우 **보너스 점수(최대 2점)**를 드리도록 하겠
- 데이터를 수집하는 경우 데이터 수집을 위해 사용한 코드
- 또 함께 제시되어야 합니다. **최소 2000개 이상의 이미지를 분석해주시기 바랍니다.**
- 만약 지지율의 변화 등, 텍스트 이외의 외부 데이터를 사용하시는 경우 해당 데이터에 대한 설명 또한 필요합니다. 외부 데이터를 함께 사용하여 분석을 실시하는 경우 **보너스 점수(최대 1점)**를 드리도록 하겠습니다.

저장이 완료되었습니다.

예) 분석에 사용한 데이터는 네이버 뉴스 검색페이지를 활용했습니다. 2021년 3일부터 10월까지를 수집 범위로 지정했습니다. '코로나'라는 키워드를 포함한 기사들을 모두 수집했으며, 정파성에 따른 차이 비교를 용이하게 하기 위해서 진보 성향으로 인식되는 한겨레와 보수 성향으로 인식되는 조선일보만을 사용했습니다.

분석에 사용한 데이터는 네이버 뉴스 검색페이지를 활용하였습니다. 2021년 9월 1일 부터 11월 30일 까지를 수집 범위로 지정했습니다. '이재명'이라는 키워드를 포함한 기사들을 모두 수집했으며, 진보 경향의 경향신문과 보수 성향의 조선 일보의 기사들을 사용하였습니다.

데이터 탐색적 분석 (1점)

데이터의 기본적인 특성에 대해서 통계적 수치를 보여주시면 됩니다.

참고: 평균(mean), 표준편차(std), 데이터수(count) 등 데이터의 기본적인 특성을 보여주시며 됩니다. 파이썬의 복습 시간의 판다스 강좌를 참고하시면 됩니다. 만약, 외부의 데이터 (예: 코로나 확진 환자 수)를 사용한다면 해당 데이터 또한 탐색적 분석을 실시해야 합니다.

탐색적 분석의 예)

1. 데이터의 전체 개수를 데이터의 탐색적 분석을 위해 시간 별로 어떻게 변하는 지를 보여줍니다.
2. 신문사별로 매 월 '코로나 관련 뉴스의 보도 수를 보여줍니다.

```
!pip install --upgrade google-cloud-vision
```

```
Requirement already satisfied: google-cloud-vision in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: google-api-core[grpc]<3.0.0dev,>=1.28.0 in /usr/local/lib/pytl
Requirement already satisfied: proto-plus>=1.15.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: google-auth<3.0dev,>=1.25.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.52.0 in /usr/local/lib/py
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: grpcio-status<2.0dev,>=1.33.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from go
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/pytl
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages
```

저장이 완료되었습니다.

```
# 라이브러리 불러오기
import pandas as pd
import numpy as np
import random

import io
import requests
from io import BytesIO
import os
import matplotlib.pyplot as plt
from PIL import Image
```

```
# 구글 클라우드 비전 관련 라이브러리들
from google.cloud import vision
from googleapiclient.discovery import build
import base64
```

```
# Mounting Google Drive:
from google.colab import drive
drive.mount('/gdrive', force_remount=True)
```

```
Mounted at /gdrive
```

```
from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=len(uplo
```

파일 선택 hidden-cat-33...e609250.json

- **hidden-cat-334509-56370e609250.json**(application/json) - 2322 bytes, last modified: 2021. 12. 8. - 100% done

Saving hidden-cat-334509-56370e609250.jso hidden-cat-334509-56370e609250.json
User uploaded file "hidden-cat-334509-563 09250.json" with length 2322 bytes

```
# Imports Credential File:
os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "hidden-cat-334509-56370e609250.json" #업로드된 J
print("Service Account Key: {}".format(os.environ["GOOGLE_APPLICATION_CREDENTIALS"]))
```

```
Service Account Key: hidden-cat-334509-56370e609250.json
```

```
# 클라이언트 만들기
client=vision.ImageAnnotatorClient()
```

저장이 완료되었습니다.

```
터프레임 생성
다면, 기존에 저장된
# 파일을 저장해야 처음부터 분석을 다시 해야하는 상황을 막을 수 있습니다.

google_api_df = pd.DataFrame()
```

```
# 이미지 리스트 불러오기
image_df = pd.read_csv("https://raw.githubusercontent.com/skku-ai-textbook/gco2014/main/data/GCO
image_df = image_df.reset_index()
image_df['index'] = image_df['index'].astype('str')
```

```
# 저장된 파일이름 만들기
image_df['file'] = "img_"+image_df['query']+"_"+image_df['news_org']+"_"+image_df['index']+".jpg
image_df.head()
```

	index	org	date	title	url	news_org	query	file
0	0	경향신문	2021.09.01.	문재인 정부 마지막 정기국회, 오늘 시작…100일간	https://search.pstatic.net/common/?src=https%3...	경향신문	이재명	img_이재명_경향신문_0.jpg

```
# 만약 특정 두 신문사만 사용할 경우
new_df= image_df[(image_df['news_org']=='경향신문') | (image_df['news_org']=='조선일보')]
```

```
1
```

이미지 전처리 (2점)

이미지 분석을 위해서 구글 API를 사용해 이미지를 분석해주세요.

참고: 14주차 수업을 참고하여, 이미지 분석을 위한 전처리를 실시합니다. 이 때, API를 사용한 이미지 처리 과정의 각 단계를 **주석**을 통해 자세히 설명해주시면 됩니다.

```
raw_data = pd.read_csv("https://raw.githubusercontent.com/skku-ai-textbook/gco2014/main/data/GC(
raw_data.shape
```

```
(9943, 3)
```

저장이 완료되었습니다.

	file	description	score
0	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Forehead	0.984656
1	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Chin	0.966452
2	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Microphone	0.919768
3	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Dress shirt	0.880985
4	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Tie	0.873751

```
# 점수에 제한을 두고 싶은 경우
raw_data[raw_data['score']>.75]
```

```
raw_data.groupby('file')['description'].agg(lambda x: ', '.join(x))
```

```
file
ags.mulvaney 2018-01-12 00.40.56 1690435662175526477_6877103859.jpg       Forehead, Chin, Mi
akgovbillwalker 2017-04-10 14.13.04 1490051352486195607_2137743151.jpg    Sky, Coat, Gesture
akgovbillwalker 2017-04-27 12.33.59 1502322670422109267_2137743151.jpg    Product, Organism,
akgovbillwalker 2017-04-30 13.59.53 1504540224993416801_2137743151.jpg    Clothing, Trousers
akgovbillwalker 2017-06-08 14.30.58 1532822127659867051_2137743151.jpg    Smile, Trousers, C

tomcottonar 2016-11-05 20.55.39 1377188962074361301_495936209.jpg         Smile, Head, Trous
vanhollenformd 2016-02-04 08.29.47 1177530425574582796_1736601433.jpg     Hair, Face, Head, S
vanhollenformd 2016-04-17 20.33.15 1230772993900571162_1736601433.jpg     Smile, Trousers, O
vanhollenformd 2016-07-16 09.51.15 1295679675443509825_1736601433.jpg     Cloud, Sky, Shorts
vp 2017-06-20 18.11.22 1541630368569093424_4837362329.jpg                 Clothing, Health c
Name: description, Length: 1000, dtype: object
```

```
df = pd.DataFrame(raw_data.groupby('file')['description'].agg(lambda x: ', '.join(x)).reset_inde
```

```
df.head()
```

	file	description
0	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Forehead, Chin, Microphone, Dress shirt, Tie, ...
1	akgovbillwalker 2017-04-10 14.13.04 1490051352...	Sky, Coat, Gesture, Naval architecture, Event,...
2	akgovbillwalker 2017-04-27 12.33.59 1502322670...	Product, Organism, Rectangle, Font, Art, Handw...
	akgovbillwalker 2017-04-30 13.59.53	Clothing, Trousers, Shirt, Product, Human,

```
df['tags'] = df['description'].apply(lambda x:",".join([str(word).lstrip(" ").replace(" ", "_")
```

```
df['tags'] = df['tags'].apply(lambda x:x.split(","))
```

저장이 완료되었습니다.

```
0      [Forehead, Chin, Microphone, Dress_shirt, Tie,...
1      [Sky, Coat, Gesture, Naval_architecture, Event...
2      [Product, Organism, Rectangle, Font, Art, Hand...
3      [Clothing, Trousers, Shirt, Product, Human, Cr...
4      [Smile, Trousers, Cloud, Sky, Plant, Water, Tr...
                             ...
995    [Smile, Head, Trousers, Shoulder, Eye, Shorts,...
996    [Hair, Face, Head, Smile, Facial_expression, V...
997    [Smile, Trousers, Outerwear, Photograph, Shirt...
998    [Cloud, Sky, Shorts, Smile, Plant, Bermuda_sho...
999    [Clothing, Health_care_provider, Safety_glove,...
Name: tags, Length: 1000, dtype: object
```

토픽 모델 학습 (2점)

이 파트는 정제 된 데이터에 실제 분석 알고리즘을 적용하는 단계입니다.

참고: 토픽 모델링을 수행시 최적의 K값을 구하는 과정을 보여주시고, 이에 대해 설명하시면 됩니다.

모델 학습 과정의 예:)

1. 여러 K값을 사용하여 모델을 훈련시킨 결과 Perplexity 점수의 변화가 적어지는 N개의 토픽을 최적의 토픽으로 사용하였습니다.
2. 각 토픽 별로 주로 사용되는 단어들을 출력해서 보여줍니다.

```
!pip install tomotopy
```

```
Collecting tomotopy
  Downloading tomotopy-0.12.2-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (16.3
     |████████████████████████████████| 16.3 MB 4.2 MB/s
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from
Installing collected packages: tomotopy
Successfully installed tomotopy-0.12.2
```

```
import tomotopy as tp
```

```
# LDA Model 설정

def lda(k_model, iter, text, a, word_remove=0):

    # 모델 설정
    model = tp.LDAModel(k=k_model, alpha=a, rm_top=word_remove, seed=871017,)

    for n, line in enumerate(text):
        model.add_doc(line.split(','))

    model.burn_in = 100
    model.train(0)

    print('문서 개수:', len(model.docs), ', 단어 개수:', len(model.used_vocabs), ', 단어의 총수
    print('제거된 단어들:', model.removed_top_words)

    print('훈련 중...',flush=True)
    for i in range(0, iter, 10):
        model.train(10)
        print('반복: {}Wt로그-우도: {}'.format(i, model.ll_per_word))

    model.summary()
    return(model)
```

저장이 완료되었습니다.

```
k_list = []
k_list.extend(list(range(2,11, 1)))
```

```
k_list.extend(list(range(12,21, 2)))
k_list.extend(list(range(25,55, 5)))

alpha_list = [.01, .05, .1, .5, 1] # 한 문서(이미지)가 얼마나 많은 토픽에 속할지와 관련 된 변수


perplexity_scores = {} # A dictionary for saving perplexity score.
df_coh = pd.DataFrame()

print(k_list)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40, 45, 50]
```

```
# 주어진 K 모델 별 Perplexity  숫자 구하기

for k in k_list: # k값을 변화시켜 학습
  for al in alpha_list: # alpha값을 변화시켜 학습
    d = {}
    #모델 학습
    model = lda(k_model=k, iter=1000, text=df['tags'], a=al)

    # perplexity score
    perplexity_scores[k]=model.perplexity # 모델의 혼란도(perplexity)값
    perplexity_df = pd.DataFrame.from_dict(perplexity_scores, orient='index', columns=['model_pe
    perplexity_df['k']=perplexity_df.index # 인덱스 저장

    # coherenece score
    coh = tp.coherence.Coherence(model, coherence='u_mass') # 유사도를 구하는 여러 방법 중  `U-m
    average_coherence = coh.get_score()
    coherence_per_topic = [coh.get_score(topic_id=n) for n in range(model.k)]

    d = {'k':[k]*k, 'alpha':[al]*k, 'eta':[1/k]*k,
         'avg_coherence':[average_coherence]*k, 'coherence':pd.Series(coherence_per_topic).T} #

                                    ame(d))
```

저장이 완료되었습니다.

```
| eta (Dirichlet prior on the per-topic word distribution)
|  0.01
|
<Topics>
| #0 (339) : Happy Smile Face Hair Facial_expression
| #1 (297) : Motor_vehicle Vehicle Tire Wheel Car
| #2 (40) : Purple Magenta Pink Metal Violet
| #3 (55) : Building Interior_design Ceiling Lighting City
| #4 (30) : Horn Trophy_hunting Working_animal Terrestrial_animal Bicycle_frame
| #5 (302) : Font Product Adaptation News Happy
| #6 (426) : Microphone Suit Spokesperson Coat Tie
| #7 (43) : Christmas_tree Cemetery Midnight Electricity Event
| #8 (302) : Plant Leisure Sky Tree Shorts
| #9 (38) : Customer Retail Selling Bag Shopping
| #10 (77) : Art Illustration Statue Line Sculpture
| #11 (43) : Shelf Publication Bookcase Book_cover Customer
| #12 (17) : Light Photograph Nature Dusk Botany
| #13 (429) : Building Sky City Cloud Window
| #14 (35) : Asphalt Road_surface Gas Infrastructure Road
| #15 (1454) : Tie Suit Smile Gesture Coat
```

```
| #16 (38) : Watch Real_estate Sleeve Social_group People
| #17 (25) : Plant Orange Peach Amber Annual_plant
| #18 (63) : Sleeve Textile T-shirt Font Sportswear
| #19 (575) : Font Electric_blue Brand Rectangle Logo
| #20 (84) : Font Gas Motor_vehicle Door Fixture
| #21 (80) : Wood Tints_and_shades Font Flooring Metal
| #22 (247) : Sky Cloud Natural_landscape Mountain Water
| #23 (90) : Black Style Black-and-white Gesture White
| #24 (206) : Tableware Food Table Cuisine Ingredient
| #25 (375) : Event Chair Suit Crowd Job
| #26 (314) : Entertainment Display_device Electronic_device Technology Performing_arts
| #27 (384) : Chair Table Suit Interaction Desk
| #28 (39) : Camera Videographer Tripod Video_camera Camera_accessory
| #29 (136) : Water Sky Plant Cloud Lake
| #30 (289) : Plant Tree Sky Grass Cloud
| #31 (25) : Tints_and_shades Darkness Gas Grey Flameless_candle
| #32 (703) : Event T-shirt Crowd Fun Leisure
| #33 (19) : Footwear Sports_gear Sports_equipment Sports_uniform Window
| #34 (79) : Hand Gesture Interaction Joint Shoulder
| #35 (144) : Job Event Engineering Employment Flooring
| #36 (30) : Handwriting Writing Wood Chef Cookware_and_bakeware
| #37 (63) : Jaw Chin Forehead Eyelash Eyebrow
| #38 (390) : Smile Happy Eyewear Vision_care Fun
| #39 (32) : Workwear Motor_vehicle Helmet Personal_protective_equipment Tie
| #40 (100) : Flag Flag_of_the_united_states Event Rectangle Flag_Day_(USA)
| #41 (647) : Clothing Outerwear Coat Photograph Human
| #42 (151) : Military_person Military_uniform Gesture Military_camouflage Government_age
| #43 (139) : Houseplant Comfort Chair Plant Picture_frame
| #44 (124) : World Fan Crowd Gesture Competition_event
| #45 (82) : Carnivore Dog Dog_breed Companion_dog Vertebrate
| #46 (71) : Player Sports_uniform Sports_jersey Jersey Sports_equipment
| #47 (47) : Vertebrate Blue Dress Glasses Azure
| #48 (83) : Cap Hat Baseball_cap Tree Fedora
| #49 (142) : Computer Table Personal_computer Desk Gadget
|
```

저장이 완료되었습니다.

```
coh['avg_coherence'].max()]
```

	k	alpha	eta	avg_coherence	coherence
0	5	0.05	0.2	-1.921228	-1.875106
1	5	0.05	0.2	-1.921228	-1.509168
2	5	0.05	0.2	-1.921228	-2.579780
3	5	0.05	0.2	-1.921228	-1.636173
4	5	0.05	0.2	-1.921228	-2.005913

```
# 최적화 모델
k, al = 5, .05
mdl_k=lda(k_model=k, iter=1000, text=df['tags'], a=al)
```

```
반복: 370       로그-우도: -5.645281110444827
반복: 380       로그-우도: -5.642158356810389
```

```
반복: 390       로그-우도: -5.657499565818498
반복: 400       로그-우도: -5.645323809874342
반복: 410       로그-우도: -5.656242145494977
반복: 420       로그-우도: -5.654922331966866
반복: 430       로그-우도: -5.643694527483941
반복: 440       로그-우도: -5.658657319808551
반복: 450       로그-우도: -5.640700432630321
반복: 460       로그-우도: -5.652401602275413
반복: 470       로그-우도: -5.6534352587056595
반복: 480       로그-우도: -5.665343895502089
반복: 490       로그-우도: -5.653202932904595
반복: 500       로그-우도: -5.656637762419645
반복: 510       로그-우도: -5.652059232512435
반복: 520       로그-우도: -5.6585176918668
반복: 530       로그-우도: -5.6529794719981
반복: 540       로그-우도: -5.659385002445889
반복: 550       로그-우도: -5.6581919645922625
반복: 560       로그-우도: -5.663339959751441
반복: 570       로그-우도: -5.647508603693505
반복: 580       로그-우도: -5.65347074747781
반복: 590       로그-우도: -5.657503088421233
반복: 600       로그-우도: -5.657877474540702
반복: 610       로그-우도: -5.657141841831886
반복: 620       로그-우도: -5.657321257393664
반복: 630       로그-우도: -5.656331604309728
반복: 640       로그-우도: -5.654835355673879
반복: 650       로그-우도: -5.653259162024978
반복: 660       로그-우도: -5.645724866395743
반복: 670       로그-우도: -5.646343243190439
반복: 680       로그-우도: -5.653744372841483
반복: 690       로그-우도: -5.655250679411951
반복: 700       로그-우도: -5.646905808253127
반복: 710       로그-우도: -5.657260740425557
반복: 720       로그-우도: -5.653036976500823
반복: 730       로그-우도: -5.642902406902365
반복: 740       로그-우도: -5.662802281249497
반복: 750       로그-우도: -5.646869334095046
                                              4424607478001
저장이 완료되었습니다.                          039959620605
                                              927963371645
반복: 790       로그-우도: -5.648931464978198
반복: 800       로그-우도: -5.650213424845096
반복: 810       로그-우도: -5.653891721655878
반복: 820       로그-우도: -5.656117750972998
반복: 830       로그-우도: -5.657582662762093
반복: 840       로그-우도: -5.655785048096686
반복: 850       로그-우도: -5.6528737763757615
반복: 860       로그-우도: -5.655562523977625
반복: 870       로그-우도: -5.6552823328113435
반복: 880       로그-우도: -5.665019821997177
반복: 890       로그-우도: -5.655843524563352
반복: 900       로그-우도: -5.662800073156076
반복: 910       로그-우도: -5.654213633489815
반복: 920       로그-우도: -5.66619148510845
반복: 930       로그-우도: -5.659451733757214
```

이미지 토픽 이름 붙이기 (2점)

학습된 모델의 결과를 사용하여 생성된 토픽의 이름을 붙이는 단계입니다.

참조: 문서 당 가장 확률이 높은 토픽을 지정하는 방법을 참조하시면 됩니다. 만약 토픽이 10개보다 많다면, 토픽의 비율이 가장 많은 10개의 토픽을 골라 각 토픽의 이름을 구해주시면 됩니다. 토픽 별 이미지를 출력하는 방식은 14주차 수업을 참조하시면 됩니다.

예) 코로나 보도에 관한 토픽 중 가장 비율이 많은 총 10개의 토픽을 선택했습니다. 매 토픽마다 토픽에 가장 잘 등장하는 단어들과 실제 문서들을 참조하여 본 결과 해당 토픽은 000에 대한 것으로 파악됩니다.

```
topic_df = pd.DataFrame()
for i in range(0, k):
  temp = pd.DataFrame()
  temp = pd.DataFrame(mdl_k.get_topic_words(i, top_n=10))
  temp.columns = ["Topic"+str(i+1),"probs"+str(i+1)]
  temp = temp.reset_index()
  if (i==0):
      topic_df = topic_df.append(temp, ignore_index=True)
  else:
      topic_df = topic_df.merge(temp, left_on="index", right_on="index")

```

```
new_df = pd.DataFrame()

for i, line in enumerate(mdl_k.docs):
    temp=pd.DataFrame(line.get_topic_dist()).T
    new_df=new_df.append(temp)

new_df = new_df.drop(["index"], axis=1)
```

```
new_df
```

저장이 완료되었습니다.

	0	1	2	3	4
0	0.012747	0.959853	0.013232	0.007206	0.006962
1	0.012747	0.677726	0.013232	0.007206	0.289089
2	0.012747	0.019430	0.013232	0.947629	0.006962
3	0.294874	0.583684	0.013232	0.007206	0.101004

```
1 # Assigning Column Names
2 new_df.columns = ['Topic'+ str(x) for x in range(1,k+1)]
```

995 0.482958 0.489642 0.013232 0.007206 0.006962

```
1 df = df.reset_index().drop(['index'], axis=1)
```

```
1 theta_df = df.merge(new_df, left_index=True, right_index=True)
```

998 0.953170 0.019430 0.013232 0.007206 0.006962

```
1 theta_df
```

	file	description	tags	Topic1	Topic2	
0	ags.mulvaney 2018-01-12 00.40.56 1690435662175...	Forehead, Chin, Microphone, Dress shirt, Tie, ...	[Forehead, Chin, Microphone, Dress_shirt, Tie,...	0.012747	0.959853	0.(
1	akgovbillwalker 2017-04-10 14.13.04 1490051352...	Sky, Coat, Gesture, Naval architecture, Event,...	[Sky, Coat, Gesture, Naval_architecture, Event...	0.012747	0.677726	0.(
2		Product, Organism, Rectangle, Font, Art, Handw...	[Product, Organism, Rectangle, Font, Art, Hand...	0.012747	0.019430	0.(
3	akgovbillwalker 2017-04-30 13.59.53 1504540224...	Clothing, Trousers, Shirt, Product, Human, Cro...	[Clothing, Trousers, Shirt, Product, Human, Cr...	0.294874	0.583684	0.(
4	akgovbillwalker 2017-06-08 14.30.58 1532822127...	Smile, Trousers, Cloud, Sky, Plant, Water, Tre...	[Smile, Trousers, Cloud, Sky, Plant, Water, Tr...	0.859128	0.019430	0.(
...	...	...	...	...	...	...

저장이 완료되었습니다.

```
1 theta_df['Highest_Topic']=theta_df[['Topic'+ str(s) for s in range(1, k+1)]].idxmax(axis=1)
```

```
1 theta_df['Highest_Topic'].value_counts()
```

```
Topic2    361
Topic3    230
Topic1    190
Topic4    115
Topic5    104
Name: Highest_Topic, dtype: int64
```

이미지 토픽 시각화 (1.5점)

토픽 모델리의 결과를 시각화를 통해 보여줍니다.

참조: 앞서 선택한 총 10개의 토픽을 사용하여 토픽 모델링을 수행하여 해결하고자 하는 문제에 대해 시각화 해주세요. 해결하고자 하는 문제에 따라 최적의 시각화 방법이 다를 수 있습니다. 예를 들어, 시기의 변화의 경우 토픽의 시기 변화를, 언론사별 변화의 경우 언론사에 따라 토픽 분포가 어떻게 차이나는 지를 보여줘야 합니다.

시각화의 예:) 언론사의 정파성에 따라 코로나 관련 보도의 이미지 표현 방식이 달라지는 살펴보기 위해서 진보(한겨레)와 보수(조선)의 토픽 별 보도량을 시각화하여 보여주었습니다.

```
1 import seaborn as sns
2 # 통계분석에 필요한 모듈
3 import scipy
```

```
1 scipy.stats.pearsonr(x=theta_df['Topic1'], y=theta_df['Topic2'])
```

```
(-0.3503767204682995, 2.9500228236621704e-30)
```

저장이 완료되었습니다. ×

```
1 scipy.stats.pearsonr(x=theta_df['Topic1'], y=theta_df['Topic4'])
```

```
(-0.17642884929229732, 1.9508508303259925e-08)
```

```
1 # 데이터 프레임의 행 이름들을 리스트로 바꾸기
2 cols = theta_df[['Topic1','Topic2','Topic3','Topic4','Topic5']].columns.to_list()
3 print(cols)
```

```
['Topic1', 'Topic2', 'Topic3', 'Topic4', 'Topic5']
```

```
1 for i in range(0,5): # 첫번째 특성
2   for j in range(i+1, 5): # 두번째 특성
3     r, p = scipy.stats.pearsonr(theta_df[cols[j]], theta_df[cols[i]]) # i번째 행과 j 번째 행의 
4     print("{:10}와 {:10}의 상관관계 r = {:10.3f} | 유의도 p = {:10.3f}".format(cols[i], cols[j],
```

```
Topic1    와 Topic2    의 상관관계 r =     -0.350 | 유의도 p =      0.000
Topic1    와 Topic3    의 상관관계 r =     -0.272 | 유의도 p =      0.000
Topic1    와 Topic4    의 상관관계 r =     -0.176 | 유의도 p =      0.000
```

```
Topic1  와 Topic5  의 상관관계 r =  -0.113 | 유의도 p =  0.000
Topic2  와 Topic3  의 상관관계 r =  -0.333 | 유의도 p =  0.000
Topic2  와 Topic4  의 상관관계 r =  -0.305 | 유의도 p =  0.000
Topic2  와 Topic5  의 상관관계 r =  -0.292 | 유의도 p =  0.000
Topic3  와 Topic4  의 상관관계 r =  -0.232 | 유의도 p =  0.000
Topic3  와 Topic5  의 상관관계 r =  -0.244 | 유의도 p =  0.000
Topic4  와 Topic5  의 상관관계 r =  -0.119 | 유의도 p =  0.000
```

```
# 라이브러리 불러오기
import networkx as nx # 네트워크 시각화
```

```
# 시각화를 위한 데이터 선택
df2 = theta_df[['Topic1','Topic2','Topic3','Topic4','Topic5']]
```

```
# 상관관계 구하기
corr = df2.corr()
corr = 1+corr # 음의 값을 갖고 있음으로 해석의 용이를 위해서
```

```
corr
```

	Topic1	Topic2	Topic3	Topic4	Topic5
Topic1	2.000000	0.649623	0.727710	0.823571	0.886564
Topic2	0.649623	2.000000	0.666535	0.695337	0.707752
Topic3	0.727710	0.666535	2.000000	0.768433	0.756442
Topic4	0.823571	0.695337	0.768433	2.000000	0.881167
Topic5	0.886564	0.707752	0.756442	0.881167	2.000000

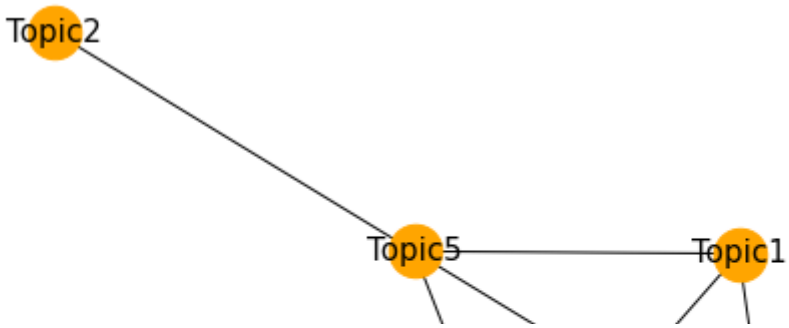
```
links = corr.stack().reset_index()
                                    lue']
```

저장이 완료되었습니다.

```
# 필터링
links_filtered=links.loc[ (links['value'] > 0.7) & (links['var1'] != links['var2']) ]
```

```
# 그래프 그리기
G=nx.from_pandas_edgelist(links_filtered, 'var1', 'var2')
```

```
# 네트워크 그리기
nx.draw(G, with_labels=True, node_color='orange', node_size=700, edge_color='black', linewidths=
```



분석 결과 논의 (1점)

토픽 모델링 분석 결과를 통해 도출된 결론을 논의해보세요

참조: 분석 결과를 간단히 요약하고, 이를 통해 알게 된 사실에 대해서 논의해주세요. 문제 제기 과성에서 예측한 결과와 분석 결과가 일치했는지의 여부를 설명해주세요. 만약 분석 결과가 일치하지 않았다면 일치하지 않은 이유를 생각해보세요.

위의 문제제기 및 예측 단계에서 수행 했던 결과 양상의 예측 결과와 매우 유사한 결과를 얻을 수 있었습니다. 정파성에 따라 이재명 후보에 대한 강조 이미지 사용의 차이가 나타남을 확인할 수 있었습니다. 경향 : 진보 - 우호적 / 조선 : 보수 - 다소 덜 우호적

저장이 완료되었습니다.

✓ 0초 오후 11:51에 완료됨