```
!pip install konlpy
!pip install wordcloud
!pip install collection
!pip install matplotlib
!apt-get install fonts-nanum*
!apt-get install fontconfig
```

```
import matplotlib
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from konlpy.tag import Okt
import re
from collections import Counter
from matplotlib.pyplot import barh

#unnamed 컬럼이름 'num'으로 바꿔주기
#lunch.rename(columns = {'Unnamed: 0': 'num'}, inplace=True)
def rename_Unnamed(dataFrame):
    dataFrame.rename(columns = {'Unnamed: 0': 'num'}, inplace=True)

#필터링 하기
def filltering(dataFrame, column_name, string):
    mask = dataFrame[column_name].str.contains(string)
    result = dataFrame[mask]
    return result
#string 빼고 넣기
def filltering2(dataFrame, column_name, string):
    mask = dataFrame[column_name].str.contains(string)
    mask = ~mask
    result = dataFrame[mask]
    return result

#text 합치기
def make_all_text(dataFrame, columnName):
    dataFrame.reset_index(drop=True, inplace=True)
    result = ""

    if dataFrame[columnName][0][0] == '[':
        for i in dataFrame.index:
            text = " ".join(dataFrame[columnName][i][1:-1].replace(' ', '').replace('₩', '').split
            result = result + " " + text

    else:
        for i in dataFrame.index:
            text = "".join(dataFrame[columnName][i])
            result = result + " " + text
    return result

#wordcloud 만들기
#딕셔너리 or 텍스트 넣기
def make_wordcloud(all_text):
    wordcloud = WordCloud(
```

```
        #font_path = 'C:WWWindowsWWFontsWWH2GTRE.TTF',
        font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf',
        width = 1600,
        height = 900
    )

    if type(all_text) == dict:
      result = wordcloud.generate_from_frequencies(all_text)
    else:
      result = wordcloud.generate(all_text)

    plt.imshow(
    result
    #interpolation='bilinear'
    )
    plt.axis("off")
    plt.show()

#원하는 그림모양으로 워드크라우드 만들기
#딕셔너리 or 텍스트 넣기
def make_wordcloud_background(all_text, background_img_path):
    from PIL import Image
    import numpy as np

    img = Image.open(background_img_path)
    img_matrix = np.array(img)

    wordcloud = WordCloud(
    #font_path = 'C:WWWindowsWWFontsWWH2GTRE.TTF',
    font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf',
    width = 1600,
    height = 900,
    mask = img_matrix,
    background_color='white'
    )

    if type(all_text) == dict:
      result = wordcloud.generate_from_frequencies(all_text)
    else:
      result = wordcloud.generate(all_text)

    plt.imshow(
    result
    #interpolation='bilinear'
    )
    plt.axis("off")
    plt.show()

#좋아요 콤마 지우고 int로 바꿔주기
#스트링으로 들어온 데이터를 int64로 바꿔준다.
def make_like(dataFrame, columnName):
    dataFrame[columnName] = dataFrame[columnName].str.replace(',', '')
    dataFrame[columnName] = dataFrame[columnName].fillna(0).astype('int64')

    return dataFrame
```

```
#여기서만 사용할 함수?
#태그를 한 리스트로 반환
def make_tags_list(tags_series):
    result = []
    for i in tags_series:
        temp_list = i[2: -2].split("', '")
        result = result + temp_list
    return result

#tag_value_counts 상위 n개 태그의 빈도수 df 반환
def value_counts_num(listOrSeries, num):
    if type(listOrSeries) == list:
      result = pd.DataFrame(Counter(listOrSeries).most_common(num))
    else:
      temp = make_tags_list(listOrSeries)
      result = pd.DataFrame(Counter(temp).most_common(num))
    return result

#plot 한글 설정
def han_font_set():
    from matplotlib import font_manager, rc
    font_name = font_manager.FontProperties(fname='/usr/share/fonts/truetype/nanum/NanumBarunGothic
    matplotlib.rc('font', family = font_name)
    matplotlib.font_manager._rebuild()

#content 전처리
def prepro_content(dataFrame):
    dataFrame = dataFrame.fillna("nan")
    dataFrame.isnull().sum() # null 사라짐

    def text_cleaner(text):
        clean = text
        publisher = "₩((.*?)₩)"
        braces = "₩[(.*?)₩]"
        braces2 = "₩{(.*?)₩}"
        braces3 = "₩【(.*?)₩】"
        weird = "[=_₩.,;:~…₩"₩"₩'₩'◇%₩<₩>/ㆍ○★☆—]"
        tab = '₩₩t'
        newline = '₩₩n'

        clean = re.sub(publisher,'', clean)
        clean = re.sub(braces,'', clean)
        clean = re.sub(braces2,'', clean)
        clean = re.sub(braces3,'', clean)
        clean = re.sub('[YTN,OSEN]','', clean)
        clean = re.sub(weird,'', clean)
        clean = re.sub(tab,'', clean)
        clean = re.sub(newline,'',clean)

        return clean

    for idx, text in enumerate(dataFrame['content']):
        dataFrame['content'][idx] = text_cleaner(text)
```

```
#나중에 학습시킬때 사용
def make_vocab(dataframe):
    okt = Okt()

    sentences_pos = []
    for line in dataframe['content']:
        sentences_pos.append(okt.nouns(line))

    max_len = max([len(i) for i in sentences_pos])

    vocab = []
    for line in sentences_pos:
        for word in line:
            vocab.append(word)

    vocab_size = len(vocab) +1
    vocab = sorted(list(vocab))


    vocab = [item for item in vocab if len(item) != 1]

    return sentences_pos, vocab
```

```
#데이터 로드
ato = pd.read_excel('./atozzang.xlsx')
bbo = pd.read_excel('./bbo.xlsx')
haewon = pd.read_excel('./haewon.xlsx')
twomuk = pd.read_excel('./2muk.xlsx')
muksta = pd.read_excel('./muksta.xlsx')
dfs = [ato, bbo, haewon, twomuk, muksta]
```

```
#content 분석

#content 전처리
#1
for i in dfs:
  prepro_content(i)
#2
sentences_pos = []
vocab = []
for i in dfs:
  tempS, tempV = make_vocab(i)
  sentences_pos.append(tempS)
  vocab.append(tempV)
#3
vocab_value_counts = []
for i in vocab:
  vocab_value_counts.append(value_counts_num(i, 15))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:155: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
```

1. 결측값을 'nan'으로 바꿔준다
2. okt모듈을 사용해 'content' 컬럼의 명사를 추출한다 (이 때 한글자는 분석 대상에서 제외시킨다)
3. 추출한 명사의 빈도수를 구해준다.

결과

dfs - 원본 데이터프레임 list

vocab - 'content' 컬럼에서 추출한 명사들이 데이터별로 있는 list

vocab_value_counts - vocab에 있는 명사들의 상위 15개의 빈도수가 있는 list

```
#태그분석

#태그 전처리
#1
for i in dfs:
    rename_Unnamed(i)

#태그수 보기
tag_value_counts = []
for i in dfs:
    tag_value_counts.append(value_counts_num(i['tags'], 15))


#워드크라우드 용 text파일
text_list = []
for i in dfs:
    text_list.append(make_all_text(i, 'tags'))
```

1. Unnamed 컬럼을 num으로 바꿔준다.
2. 태그 빈도수 별로 빈도수를 구해준다
3. word cloud를 만들기 위해 태그들을 text화 시켜준다.

결과

tag_value_counts - 각 데이터프래임의 태그 빈도수를 상위 15개를 추출해 저장한 list

text_list - word cloud를 만들기 위해 모든 태그들을 택스트로 변환 저장한 list

```
#content 시각화
#barplot


fig = plt.figure(figsize = (20,20))
fig.add_subplot(2, 3, 1)
barh(vocab_value_counts[0][0], vocab_value_counts[0][1])
fig.add_subplot(2, 3, 2)
barh(vocab_value_counts[1][0], vocab_value_counts[1][1])
fig.add_subplot(2,3,3)
barh(vocab_value_counts[2][0], vocab_value_counts[2][1])
fig.add_subplot(2,3,4)
barh(vocab_value_counts[3][0], vocab_value_counts[3][1])
fig.add_subplot(2,3,5)
barh(vocab_value_counts[4][0], vocab_value_counts[4][1])
plt.show()
```

각각 사람들의 태그 빈도수를 막대그래프 표현

```
#tag 시각화
#barplot

fig = plt.figure(figsize = (20,20))
fig.add_subplot(2, 3, 1)
barh(tag_value_counts[0][0], tag_value_counts[0][1])
fig.add_subplot(2, 3, 2)
barh(tag_value_counts[1][0], tag_value_counts[1][1])
fig.add_subplot(2,3,3)
barh(tag_value_counts[2][0], tag_value_counts[2][1])
fig.add_subplot(2,3,4)
barh(tag_value_counts[3][0], tag_value_counts[3][1])
fig.add_subplot(2,3,5)
barh(tag_value_counts[4][0], tag_value_counts[4][1])
plt.show()
```

각각 사람들의 태그 빈도수를 막대그래프 표현

```
#시각화
#word cloud

for i in range(len(vocab_value_counts)):
  temp = vocab_value_counts[i].set_index(0).to_dict()[1]
  make_wordcloud(temp)
```

각각 사람들의 단어 빈도수를 워드클라우드로 표현

```
#tag 시각화
#word cloud

for i in range(len(tag_value_counts)):
  temp = tag_value_counts[i].set_index(0).to_dict()[1]
  make_wordcloud(temp)
```

각각 사람들의 태그 빈도수를 워드클라우드로 표현

✓ 1초 오후 3:58에 완료됨