

Dc인사이드 실시간 베스트 갤러리 글 분석

github :https://github.com/bbobbabbi/dc_project

목차

[github :https://github.com/bbobbabbi/dc_project](https://github.com/bbobbabbi/dc_project)

[목차](#)

[데이터 전처리](#)

[url 구조 분석](#)

[게시글 구조 파악과 웹 크롤링](#)

[의도하지 않았던 데이터 처리](#)

[후에 일어나는 문제점](#)

[조회수별, 추천수별 단어 분류](#)

[dcinside_data에 긍정도 붙이기](#)

[데이터 분석](#)

[분석전에](#)

[워드 클라우드](#)

[실시간 베스트 채택글에 대한 시각](#)

[dcinside_data.csv or dcinside_new_data.csv 파일을 사용한 분석](#)

[평균 조회수 이상의 글](#)

[상위 100개 갤러리별 글 수 분포](#)

[상위 50개 갤러리별 채택된글 수](#)

[상위 50개 갤러리별 총조회수](#)

[상위 50개 갤러리 날짜별 조회수 변화](#)

[긍정도 비교](#)

[실시간 베스트의 긍정도](#)

[싱글,아겔의 긍정도](#)

[긍정적인 갤러리가 있을까?](#)

[긍정 비율이 가장 높은 30개 갤러리](#)

[긍정 비율이 가장 높은 30개 갤러리 \(표본글 100개 이상\)](#)

[채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 \(전체 글 수 많은 순\)](#)

[시간에 따른 채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 변화](#)

[실시간 베스트에 쓰인 단어들에 대한 시각](#)

[df_sort_by_view_count.csv or df_sort_by_word_count.csv 파일을 사용한 분석](#)

[단어의 중복횟수,총추천수,총조회수 확인](#)

[총조회수, 총추천수, 중복횟수 비교 그래프](#)

[줄 세우기](#)

[중복횟수가 1번인 명사와 2번 이상인 단어의 비율](#)

[중복 횟수를 다시 봐 보자](#)

[각 단어별 중복횟수,추천수,조회수 확인](#)

[각 단어별 조회수, 추천수, 중복횟수 비교](#)

[줄세우기](#)

[결론](#)

데이터 전처리

url 구조 분석

실시간 베스트 갤러리의 글 목록을 가져오기 위해 먼저 dc의 url 구조를 파악했다

<https://gall.dcinside.com/board/lists/?id=dcbest> 기본 주소에서

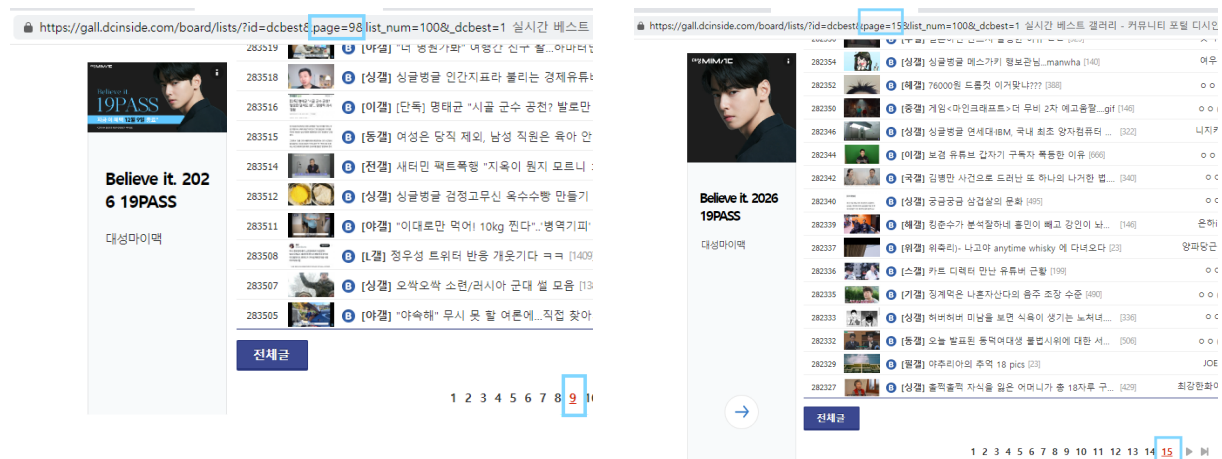


https://gall.dcinside.com/board/lists/?id=dcbest&list_num=100&sort_type=N&search_head=1&page=1

한번에 보는 글 목록을 100개로 늘리면 주소가 이런식으로 변경된다



100개의 글 목록과 게시판의 다음 글을 가져오는 번호를 불러온다면 url이 이런식으로 변경된다



https://gall.dcinside.com/board/lists/?id=dcbest&page=15&list_num=100&dcbest=1

다른 두 페이지를 두고 url을 비교해본다면

누른 번호에 따라 page= 옆의 숫자가 현재 눌러있는 번호와 같다는 것을 알 수 있다

list_num의 값과 page 번호를 다르게 넣어줌으로 다른 번호의 페이지들과 한번의 많은 양의 텍스트를 가져 올 수 있다

추가로 가져올 수 있는 데이터의 한계는 1694번페이지까지였다

(시간에 따라 보이는 글의 수가 달라지는 것 같다, 데이터를 뽑은 날에는 1660페이지까지 보였는데 글을 작성하는 당일에는 1694 페이지까지만 나오는 것을 볼 수 있다)

◀ ◀ 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693

1694

게시글 구조 파악과 웹 크롤링

웹 크롤링을 위해 BeautifulSoup 라이브러리를 사용할 했다

1694번 페이지를 받아 분류해 리스트로 만들고 리스트를 이용해 데이터 프레임으로 변경할 것이다

새로운 게시글이 거의 5분마다 생성되기에

마지막 번호에 있는 마지막 글은 사라지니 마지막 순번부터 1번까지 반복을 하기로 하였다

또한 새글 생성으로 크롤링 하기 바로 전 페이지의 마지막 글은 데이터에 들어가지 않게 됨으로 누락되는 데이터가 되지만 그 수가 많지 않아 고려하지 않기로 했다

가장 마지막에 있던 글의 작성 날짜는

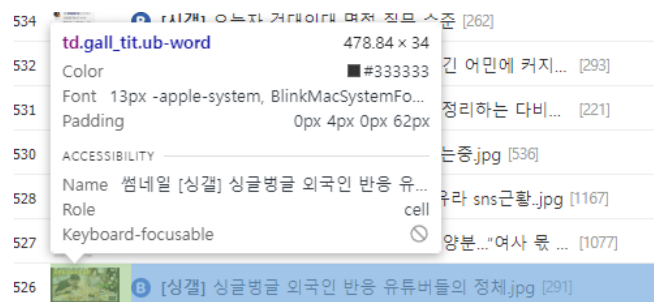
21.04.13로

데이터를 수집한 날인

24.11.12까지의 게시글을 받을 것이다

BeautifulSoup로 각 페이지의 html을 요청하여 받은 후

그 안에서 게시글들의 정보를 뽑아내면 되겠다



개발자 도구로 글의 구조를 보았을 때 뽑아야하는 텍스트는 td 안에 있는 것을 알 수 있다

td는 tr 태그로 묶여 있을 것이고

다른 게시글들의 구조를 확인 해 보았을 때

283531		[상겅] 싱글벙글 기안84 만나서 꼭보 정리하는 다비...	[221]	최강현화이글스면	11.25	24291	33
283530		[주겅] 기안84 이젠 한내 안산다노 하는중.jpg [536]		패력단 총디 (118,180)	11.25	43696	812
283528		[아겅] 정우성에게 핵핀치 보리는 정유라 sns근황.jpg [1167]		강동	11.25	56749	1039
283527		[이겅] [단독] 김건희 약인. 송산 권력 양분. "여사 룩 ...	[1077]	고맙음빠	11.25	9986	47
283526		[상겅] 싱글벙글 외국인 반응 유튜브들의 정체.jpg [291]		말근이	11.25	49996	396
283525		trub-content-us-post.thum 840 x 34 [연장 선임 실화나?] [649]		동결러 (125,183)	11.25	53215	879
283522		ASML [상겅] 싱글벙글 근 10년새 급격하게 커진 회사들 [168]		크달	11.25	28720	115





class가 ub-content us-post thum 인 tr에 td 태그들로 각 게시물에 대한 정보(text)들이 들어있는 것을 볼 수 있다
또한 tr 안에

번호	제목	글쓴이	작성일	조회	추천
<pre><tr class="ub-content us-post thum" data-no="283526" data-type="icon_btimebest"> <td class="gall_num">283526</td> <td class="gall_tit ub-word"> <td class="gall_writer ub-writer" data-nick="달근이" data-uid="roangus11" data-ip data-loc="list"> <td class="gall_date" title="2024-11-25 11:05:02">2024-11-25 11:05:02</td> <td class="gall_count">49996</td> <td class="gall_recommend">396</td> </tr></pre>					

위의 칼럼의 정보들이 다 들어 있기에 tr을 한번에 가져오면 되겠다

이제 위에서 받은 html에서 ub-content us-post thum class의 tr을 통째로 가져와

tr이 감싸고 있는 td 안의 text를 가져와

글번호, 제목, 작성자, 작성날짜, 조회수, 추천수를 컬럼으로 두고 데이터 프레임화 하기 위해

분리 후 리스트에 저장했다

의도하지 않았던 데이터 처리

처음 데이터를 받은 후 예상치 못한 문제가 하나 생겼는데

두번째 컬럼인 제목의 값으로 제목만 들어오는 것이 아닌

실시간 베스트에 올라오기전 원본 글을 가져온 갤러리와

글에 남겨진 댓글 수 가 종괄호에 묶여 포함되어 있는 것이다

[주갤] 마신거 [88]	정인오락실	21.04.13	5870	12
[아갤] 윤탈, 민합, 1조, 적맛, 동남아, 누나, JPG [800]	롤합	21.04.13	77399	1439
[해갤] 진짜 개미친새끼...gif [172]	KB (112.148)	21.04.13	13937	127
[토갤] 플레이스토어 110만원 해킹당함거 후기.jpg [152]	K보스타	21.04.13	12230	69
[새갤] 하태경 패북했다 ㅋㅋㅋㅋㅋㅋㅋㅋ (157)	ㅇㅇ (121.171)	21.04.13	9351	87
[주갤] 행동하는 주동이 정의구현 하고 왔다 [108]	버번위스키	21.04.13	9690	108
[아갤] 공무원갈 논란...jpg [541]	ㅇㅇ (210.178)	21.04.13	23156	83



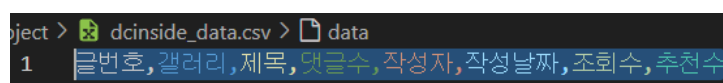
제목의 텍스트는 td 안에 2개의 a 태그중 앞의 것으로 갤러리와 같이 묶여있고 밑의 댓글 수는 따로 a 태그로 묶여 있는 것을 알 수 있다

처음 값을 가져올 때 td 값에 두 값이 포함되는 것을 예측을 하지 못 해 제목 외 두 텍스트가

제목 컬럼에 같이 저장이되어 버렸다

ex) 원했던 값 ⇒ 마신거, 실제 저장된 값 ⇒ [주갤]마신거[88]

앞과 뒤의 정보를 지워버리는 것보단 새로운 컬럼으로 만들어 분석하는게 좋을 것 같아



제목 컬럼 앞에 갤러리를 그 뒤에 댓글수를 가질 수 있도록 데이터 프레임의 컬럼을 바꿨고

데이터를 가져오는 과정에서

정규식을 사용하여 갤러리 제목 댓글수를 분리하여 리스트에 순서대로 추가하였다

이로써 "글번호", "갤러리", "제목", "댓글수", "작성자", "작성날짜", "조회수", "추천수" 순서대로 리스트에 값이 들어가 졌으며

이것을 데이터 프레임으로 만든 후 csv 파일로 추출했다

밑은 위 과정의 전체적인 코드이다

```
import requests
import pandas as pd
import re
from bs4 import BeautifulSoup
import time
import random

headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,

# 읽을 페이지 범위
start_page = 1669
end_page = 0

page_num = start_page

while page_num >= end_page:
    url = f"https://gall.dcinside.com/board/lists/?id=dcbest&list_num=100&sort_type=N&search_he

    try:
        # 요청
        respons = requests.get(url, headers=headers)
        respons.raise_for_status() # HTTP 오류가 발생한 경우 예외를 발생시킴

        html = respons.text
        soup = BeautifulSoup(html, 'html.parser')
        rows = soup.find_all('tr', attrs={"class": "ub-content us-post thum"})
        print(f"응답완료 {page_num} 추출")

    except requests.exceptions.RequestException as e:
        # 요청 실패 시의 페이지 번호와 함께 오류 출력
        print(f"요청 실패: {page_num} 페이지에서 오류 발생")
        print(f"오류 내용: {e}")
        # 요청 실패 후 대기하고 다시 시도
        time.sleep(random.uniform(5, 10)) # 5~10초 대기 후 재시도
        continue # 예외 발생 시 현재 페이지 번호로 다시 시도

data = []

for row in rows:
    columns = row.find_all('td')
    #td 텍스트 추출 후 리스트로 저장
    row_data = [column.get_text(strip=True) for column in columns]
    # 모든 데이터를 하나의 리스트로 추가
    data.extend(row_data)

# 컬럼 2개를 추가하기 위해 새로운 리스트로 변경
chunked_data = [data[i:i+6] for i in range(0, len(data), 6)]

for row in chunked_data:
    # [갤러리] 제목 [댓글수] 형태를 분리하는 정규식
    match = re.match(r"\[(.*?)\]\[(.*?)\]\[(.*?)\]$", row[1])
    if match:
        # '갤러리' 부분, 대괄호 제거
```

```

        row[1] = match.group(1).strip()
        # '제목' 부분
        row.insert(2, match.group(2).strip())
        # '댓글수' 부분, 대괄호 제거
        row.insert(3, match.group(3).strip())

# DataFrame 생성
df = pd.DataFrame(chunked_data, columns=["글번호", "갤러리", "제목", "댓글수", "작성자", "작성날짜"])

# 첫 번째 반복에서만 파일을 새로 생성
if page_num == start_page:
    df.to_csv('dcinside_data.csv', index=False, encoding='utf-8-sig')
else:
    # 이후 반복에서는 기존 CSV 파일에 추가
    df.to_csv('dcinside_data.csv', mode='a', header=False, index=False, encoding='utf-8-sig')

# 1~5초 대기
time.sleep(random.uniform(1, 5))

# 다음 페이지로 이동
page_num -= 1

```

후에 일어나는 문제점

[]로 갤러리, 제목, 댓글수를 구분하도록 했는데

제목에 추가적인 []가 들어가는 글들을 간과하지 못하여 이런 글들에 대한 처리를 하지 못하였다

따라서 [속보] [단독] 등과 같은 뉴스들을 가져온 글들은 후에 분석에서 빠지게 되었다

추가로

1669 페이지의 글을 읽어 166000개의 데이터가 생기는 것이 맞다 생각했는데 어떠한 이유인지 드랍된 데이터가 1303개가 있었고

위의 nan까지 빼고 계산한다면 총 데이터의 크기는

조회수별, 추천수별 단어 분류

가장 많이 쓰인 단어를 육안으로 확인하기 위해 데이터를 분류했다

- 데이터 입출력 및 처리: `pandas`
- 한국어 자연어 처리: `konlpy`
- 효율적인 데이터 집계: `collections.defaultdict`

```

import pandas as pd
from konlpy.tag import Okt
from collections import defaultdict

```

```

# CSV 파일 읽기
df = pd.read_csv('dcinside_data.csv')

# Okt 객체 생성
okt = Okt()

# 중복 처리 및 명사별 조회수, 추천수 합산을 위한 딕셔너리 초기화
noun_info = defaultdict(lambda: {'count': 0, 'view': 0, 'recommend': 0})

# 제목 열에서 명사 추출 중복처리와 동시에 조회수, 추천수 합산
for _, row in df.iterrows():
    title = str(row['제목']) # 제목을 문자열로 변환
    views = row['조회수']
    recommends = row['추천수']

    # 제목이 NaN이거나 빈 문자열이면 처리하지 않도록 skip
    if title.strip() == '':
        continue

    # 제목에서 명사 추출
    nouns = okt.nouns(title)

    # 명사별로 중복횟수와 조회수, 추천수 업데이트
    for noun in nouns:
        noun_info[noun]['count'] += 1
        noun_info[noun]['view'] += views
        noun_info[noun]['recommend'] += recommends

# 중복 처리된 명사들로 새로운 데이터프레임 생성
result_data = []
for noun, data in noun_info.items():
    result_data.append([noun, data['count'], data['view'], data['recommend']])

# 새로운 데이터프레임 생성
result_df = pd.DataFrame(result_data, columns=['명사', '중복횟수', '총조회수', '총추천수'])

#정렬
df_sort_by_view_count = result_df.sort_values(by='총조회수', ascending=False)
df_sort_by_word_count = result_df.sort_values(by='중복횟수', ascending=False)

# CSV 파일로 저장
df_sort_by_view_count.to_csv("df_sort_by_view_count.csv")
df_sort_by_word_count.to_csv("df_sort_by_word_count.csv")

```

```

dc_project > df_sort_by_view_count.csv > data
1 ,명사,중복횟수,총조회수,총추천수
2 13,싱글빙글,30335,1556983791.0,16550106.0
3 162,근황,8135,448334538.0,5372012.0
4 117,이유,6549,326775093.0,4068801.0
5 409,흔적,5171,272509655.0,3257399.0
6 43,한국,4880,232282988.0,2968355.0
7 324,일본,4818,207122511.0,2097926.0
8 274,여자,2515,156624587.0,1859062.0
9 372,요즘,2030,129242622.0,1426399.0
10 145,레전드,2190,127018057.0,1589692.0
11 477,녀,2208,126545444.0,1803220.0
12 246,남자,1932,113451715.0,1300802.0

```

```

c_project > df_sort_by_word_count.csv > data
1 ,명사,중복횟수,총조회수,총추천수
2 13,싱글빙글,30335,1556983791.0,16550106.0
3 162,근황,8135,448334538.0,5372012.0
4 117,이유,6549,326775093.0,4068801.0
5 409,흔적,5171,272509655.0,3257399.0
6 43,한국,4880,232282988.0,2968355.0
7 324,일본,4818,207122511.0,2097926.0

```

한국어 자연어 처리이다 보니 gif와 jpg가 없는 것이 보인다

또한 훌쩍훌쩍과 같이 두 번 반복되는 단어는 훌쩍으로 2번 카운팅 된다는 것이 함정이다

dcinside_data에 긍정도 붙이기

pandas와

beomi/KcELECTRA-base 모델을 사용하기 위해

transformers, torch 라이브러리를 사용하였다

처음 작업은 cpu 사용으로 배치 크기가 1인 상태로 모델을 사용했는데

처리 시간이 너무 오래 걸려 처리할 수 있는 의문이 생겨 gpt에게 예상 시간을 확인해보았다

예상 시간 계산:

1. 모델 특성:

- ELECTRA 는 BERT 계열 모델로, 상대적으로 높은 성능을 요구하는 모델입니다. 텍스트 길이가 평균 50글자 정도라면, 한 문장당 처리 시간이 대체로 50ms에서 100ms 정도일 수 있습니다. (사용하는 모델, 하드웨어 환경, 배치 크기, 입력 길이에 따라 다를 수 있음)

2. 한 배치 처리 시간:

- 배치 크기: 32개의 텍스트를 한 번에 처리합니다. 각 텍스트를 GPU에서 처리하는 데 평균 50ms가 걸린다고 가정하면, 한 배치의 처리 시간은 약 1.6초입니다.

3. 전체 처리 시간:

- 총 데이터가 16만 개이며, 배치 크기가 32이므로 배치 수는:

$$\text{배치 수} = \frac{160,000}{32} = 5000 \text{ 배치}$$

- 각 배치의 처리 시간이 1.6초라면, 전체 시간은 대략:

$$\text{전체 시간} = 5000 \times 1.6 = 8000 \text{ 초}$$

$$8000 \text{ 초} \div 60 \approx 133.33 \text{ 분} \approx 2 \text{ 시간 } 13 \text{ 분}$$

배치 크기를 32개로 한다 해도 2시간 13분이 걸리는데 배치가 1로 돌리는 cpu는 엄청나게 많은 시간이 들어갈 것이라 확신이 들어 가지고 있는 gpu를 사용하기로 했고

내 gpu의 compute capability 값을 확인하여 cuda sdk의 버전을 맞춰 깔아준 뒤

			NVIDIA TITAN RTX, GeForce RTX 2080 Ti, RTX 2080 Super, RTX 2070 Super, RTX 2060 Super, RTX 2060 12GB, RTX 2060, GeForce GTX 1660 Ti, GTX 1660 Super, GTX 1650 Super, GTX 1650, MX550, MX450	Quadro RTX 8000, Quadro RTX 6000, Quadro RTX 5000, Quadro RTX 4000, T1000, T600, T400, T1200(mobile), T600(mobile), T500(mobile), GeForce GTX 1660 Ti, GTX 1660 Super, GTX 1650 Super, GTX 1650, MX550, MX450		
7.5	Turing	TU102, TU104, TU106, TU116, TU117			Tesla T4	

PyTorch Build	Stable (2.5.1)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	CPU
Run this Command:	pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu124			

Compute Capability (CUDA SDK support vs. Microarchitecture)											
CUDA SDK Version(s)	Tesla	Fermi	Kepler (Early)	Kepler (Late)	Maxwell	Pascal	Volta	Turing	Ampere	Ada Lovelace	Hopper
1.0 ^[39]	1.0 – 1.1										
1.1	1.0 – 1.1+x										
2.0	1.0 – 1.1+x										
2.1 – 2.3.1 ^{[40][41][42][43]}	1.0 – 1.3										
3.0 – 3.1 ^{[44][45]}	1.0	2.0									
3.2 ^[46]	1.0	2.1									
4.0 – 4.2	1.0	2.1									
5.0 – 5.5	1.0			3.5							
6.0	1.0		3.2	3.5							
6.5	1.1			3.7	5.x						
7.0 – 7.5		2.0			5.x						
8.0		2.0				6.x					
9.0 – 9.2			3.0				7.0 – 7.2				
10.0 – 10.2			3.0					7.5			
11.0 ^[47]				3.5					8.0		
11.1 – 11.4 ^[48]				3.5					8.6		
11.5 – 11.7.1 ^[49]				3.5					8.7		
11.8 ^[50]				3.5						8.9	9.0
12.0 – 12.5					5.0						9.0

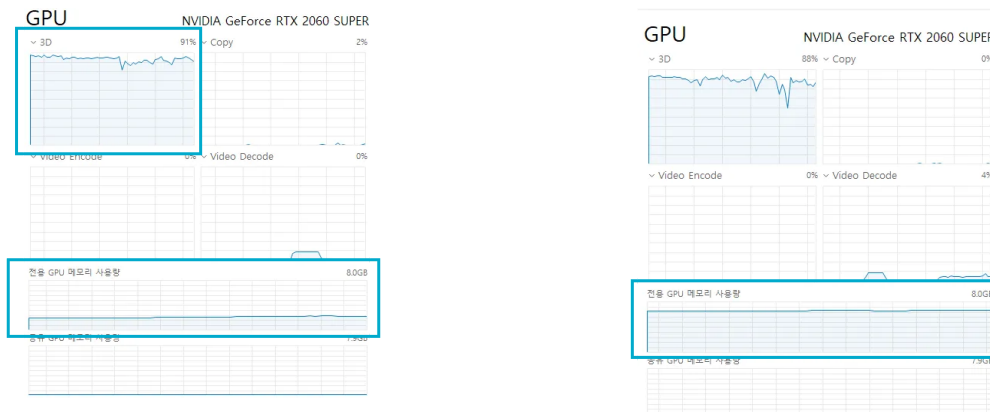
cuda sdk 버전에 따라 사용해야 할 pytorch버전 또한 달라지기에 재 설치 해주었다


```
import torch
print(torch.cuda.is_available()) # True여야 CUDA가 활성화됨
print(torch.cuda.current_device()) # 현재 사용할 GPU의 id
print(torch.cuda.get_device_name(torch.cuda.current_device())) # GPU 이름 출력

✓ 13.6s

True
0
NVIDIA GeForce RTX 2060 SUPER
```

결과적으로 gpu를 사용할 수 있게 되었고
먼저 배치를 32로 확 늘려 코드를 돌려보았는데



확실히 GPU를 사용하고 있음을 알 수 있었다만
GPU의 메모리는 여유가 많이 있어 시행착오를 겪으며 배치 사이즈를 바꿔주었고
메모리를 전부 사용할 때의 배치 사이즈가 1024인 것을 알게 되었다
그 결과 작업은 3분만에 끝나게 되었고

```
# 새로운 csv 파일로 저장
df.to_csv('dcinside_new_data.csv', index=False)

✓ 2m 48.9s
```

```
object > dcinside_new_data.csv > data
1 금번호, 갤러리, 제목, 댓글수, 작성자, 작성날짜, 조회수, 추천수, 긍정
2 86, 메갈, 이득중 해명문 일갈 ㅋㅋㅋㅋㅋㅋㅋㅋ, 132, 0, (182.227), 21.04.13, 13746.0, 211.0, 1
3 85, 군갈, 누리호 글 보고 써보는 클러스터링과 로켓계의 진짜 광기(1), 45, 애드리언뉴이, 21.04.13, 3144.0, 75.0, 0
4 84, 싱갈, 싱갈방글 총격적인 역센징의 정체, 29, 조선족알바, 21.04.13, 6956.0, 113.0, 0
5 82, 아갈, "9,900원... 동네... 트럭 피자... 차 참사... jpg", 639, 피자(14.42), 21.04.13, 56867.0, 980.0, 0
6 81, 새갈, 강남국 실시간 ㅋㅋ 개쫓았다ㅋㅋㅋㅋ, 33, Qdbdv, 21.04.13, 4010.0, 85.0, 0
7 80, 롯데, 허문회 인터뷰ㅋㅋㅋㅋ, gisa, 5, 0, (223.38), 21.04.13, 9128.0, 167.0, 0
8 79, 아갈, 역대급 뽐손 언냐... jpg, 508, 0, (114.204), 21.04.13, 55029.0, 1420.0, 0
9 78, 아갈, 다인생 맛은 햄글루 맛은 햄글루 (100.33), 21.04.13, 3388.0, 138.0, 0
```

글의 제목이 긍정적인지 부정적인지를 판단할 수 있는 데이터를 갖게 되었다

밑의 코드를 실행 시킨다면 기존 dcinside_data.csv 파일에서 긍정 컬럼을 추가할 수 있다

```
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

# GPU 사용 여부 확인
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 한국어 모델 로드
model_name = "beomi/KcELECTRA-base"
```

```

model = AutoModelForSequenceClassification.from_pretrained(model_name).to(device)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# 모델이 GPU에 로드되었는지 확인
print(next(model.parameters()).device) # 'cuda:0'이 출력되면 GPU에 로드된 것

# CSV 파일 읽기
df = pd.read_csv('dcinside_data.csv')
df = df.dropna()

# 제목 열을 가져와서 배치 처리하는 함수
def batch_analyze_sentiment(titles, batch_size=1024):
    # 텍스트를 배치로 나누어 토큰화
    inputs = tokenizer(titles, padding=True, truncation=True, return_tensors="pt")
    # 데이터를 GPU로 이동
    inputs = {key: value.to(device) for key, value in inputs.items()}

    # inputs가 GPU에 있는지 확인 cuda:0이 GPU에서 실행되고 있음
    print(inputs['input_ids'].device)

    # 불필요한 그래디언트 계산을 피하기 위해
    with torch.no_grad():
        outputs = model(**inputs)
        # 예측 결과는 GPU에서 반환됨
        predictions = torch.argmax(outputs.logits, dim=-1)

    return predictions

# 데이터프레임의 제목 열을 배치로 나누어 감정 분석 실행
batch_size = 1024
titles = df['제목'].tolist()

# 배치로 나누어 처리
predictions = []
for i in range(0, len(titles), batch_size):
    batch_titles = titles[i:i+batch_size]
    print(batch_titles)
    batch_predictions = batch_analyze_sentiment(batch_titles)

    # GPU에서 직접 처리된 결과를 그대로 사용
    # 텐서를 리스트로 변환하여 추가
    predictions.extend(batch_predictions.tolist())

# 결과를 긍정 컬럼 값으로 변환 (긍정은 1, 부정은 0)
df['긍정'] = [1 if pred == 1 else 0 for pred in predictions]

# 새로운 CSV 파일로 저장
df.to_csv('dcinside_new_data.csv', index=False)
# 기존 csv 파일로 저장
df.to_csv('dcinside_data.csv', index=False)

```

데이터 분석

분석전에

댓글 수 까지 분석하여 조회수, 추천수 별 점수를 매겨 점수별로 단어를 분류하고 싶었지만

각각이 부여 받는 점수가 달라진다면 내 개인 적인 주관이 많이 들어간 분석이 될 것이라 생각하여

추가적인 점수로 분류하지 않고 각 글이 가진 수로 판별했다

추가적으로 확인하고 싶었던 것은 제목에 들어가는 단어에 따른 조회수였기 때문에

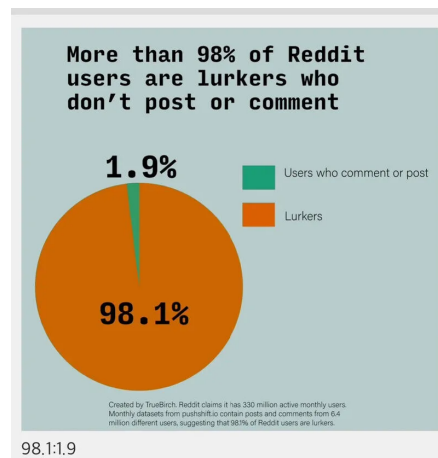
조회수가 중점으로 분석하게 되었다

추천수를 같이 분석하게 된 이유는

추천의 성격이 댓글 작성에 비해 소극적인 참여로 큰 힘이 들지 않고

내 의견을 표출 할 수 있다는 점이다

그로 인해 자기가 의견을 표하고 싶은 글이나 제목을 본다면 더욱 적극적으로 게시글을 누르게 되는 동기가 될 수 있다 판단하여 분석에 넣었다



댓글 수 까지는 그 글을 읽게끔 하는데 중요하지 않다 생각하여 분석하지 않았다

워드 클라우드

- 텍스트 처리 및 데이터 로드: `pandas`
- 워드 클라우드 생성: `wordcloud`
- 이미지 시각화: `matplotlib.pyplot`
- 정규식 제거모듈 : `re`

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("dcinside_data.csv")

text = ""

for index in range(df.shape[0]):
    text += str(df.iloc[index]['제목'])

# 워드 클라우드 생성
wordcloud = WordCloud(font_path="c:/windows/fonts/malgun.ttf", width=800, height=800, background_color="white", min_font_size=10).generate(text)

# 보여주기
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

처음에 이 코드로 했을 때 제대로 작동되지 않았는데

`text +=` 에서 계속해서 새로운 문자열 객체가 생성되어 메모리 할당이 너무 많아지기 때문이었다

이를 해결하기 위해 중간과정에서 새로운 객체가 만들어지지 않는 `str.join()`을 사용했고

```
from wordcloud import WordCloud
import re
import matplotlib.pyplot as plt
import pandas as pd

# 데이터 읽기
df = pd.read_csv("dcinside_data.csv")
text = ' '.join(df['제목'].dropna().astype(str).apply(lambda x: re.sub(r'(?i)jpg|gif|싱글빙글|후기|

# '제목' 열을 바로 워드 클라우드에 전달하여 텍스트 생성
```

```
wordcloud = WordCloud(font_path="C:\\Windows\\Fonts\\malgun.ttf",
                        width=800,
                        height=800,
                        background_color='white',
                        min_font_size=10).generate(text)

# 워드 클라우드 표시
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



이미지를 뽑아냈다

```
.apply(lambda x: re.sub(r'(?i)jpg|gif|싱글병글|후기|홀쩍홀쩍|근황|오썩오썩|스압|이유', '', x)))
```

를 사용해 위 텍스트들은 포함하지 않았다

(너무 많은 중복으로 다른 글들의 비율을 알 수 없어 포함하지 않았다)

조금 더 의미 있는 이미지를 뽑기 위해 dc 로고를 마스크 이미지를 사용해서

워드클라우드를 뽑아 봤다

이번에는 어떠한 수정 없이 만들었다

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from PIL import Image

# 데이터 읽기
df = pd.read_csv("dcinside_data.csv")
text = ' '.join(df['제목'].dropna().astype(str))

icon = Image.open('./dcinside.png').resize((1980,1080))

# 이미지를 RGBA 모드로 변환
icon = icon.convert("RGBA")

# 이미지를 numpy 배열로 변환
icon_data = np.array(icon)

# 흰색 부분을 마스크로 사용 (흰색 부분을 불투명하고, 나머지 부분은 투명하게 설정)
```



```
# 이미지를 numpy 배열로 변환
icon_data = np.array(icon)

# 흰색 부분을 마스크로 사용 (흰색 부분을 불투명하고, 나머지 부분은 투명하게 설정)
mask_data = np.all(icon_data[:, :, :3] == [255, 255, 255], axis=-1)

# 마스크를 생성: 흰색 부분은 255 (불투명), 나머지 부분은 0 (투명)
icon_data[mask_data] = [0, 0, 0, 255]
icon_data[~mask_data] = [255, 255, 255, 0]

# 수정된 이미지를 다시 이미지 객체로 변환
mask = Image.fromarray(icon_data)

# PIL 이미지를 numpy 배열로 변환 (WordCloud는 NumPy 배열을 필요로 함)
mask_array = np.array(mask)

# 제목 열을 바로 워드클라우드에 전달하여 텍스트 생성
wordcloud = WordCloud(font_path="C:\\Windows\\Fonts\\gulim.ttc",
                      collocations=False, # 자주 등장하는 단어들 제외 false
                      max_font_size=1000,
                      min_font_size=10,
                      background_color='white',
                      mask=mask_array
                      ).generate(text)

# 워드 클라우드 표시
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



그 외 더 많은 단어를 표시하고 싶어 이미지의 해상도를 높여주며
최대 나올 수 있는 단어와 최대 단어 크기를 지정해 줬다

```
from wordcloud import WordCloud
import re
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from PIL import Image

# 데이터 읽기
df = pd.read_csv("dcinside_data.csv")
text = ' '.join(df['제목'].dropna().astype(str).apply(lambda x: re.sub(r'(?i)jpg|gif|싱글빙글|후기

# 이미지 불러오기 + 해상도 설정
```

```
icon = Image.open('./dcinside.png').resize((3840,2160))

# 이미지를 RGBA 모드로 변환
icon = icon.convert("RGBA")

# 이미지를 numpy 배열로 변환
icon_data = np.array(icon)

# 흰색 부분을 마스크로 사용 (흰색 부분을 불투명하고, 나머지 부분은 투명하게 설정)
mask_data = np.all(icon_data[:, :, :3] == [255, 255, 255], axis=-1)

# 마스크를 생성: 흰색 부분은 255 (불투명), 나머지 부분은 0 (투명)
icon_data[mask_data] = [0, 0, 0, 255] # 흰색 부분을 불투명하게 설정
icon_data[~mask_data] = [255, 255, 255, 0] # 나머지 부분을 투명하게 설정

# 수정된 이미지를 다시 이미지 객체로 변환
mask = Image.fromarray(icon_data)

# PIL 이미지를 numpy 배열로 변환
mask_array = np.array(mask)

# 제목 열을 바로 워드 클라우드에 전달하여 텍스트 생성
wordcloud = WordCloud(font_path="C:\\Windows\\Fonts\\gulim.ttc",
                      collocations=False, # 자주 등장하는 단어들 제외 false
                      max_words= 2000,
                      max_font_size = 500,
                      background_color='white',
                      mask = mask_array
                      ).generate(text)

# 워드 클라우드 표시
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



jpg|싱글벙글|흠쩍흠쩍|오싹오싹 만 뺀 경우

나머지 단어들이 차지하는 비율도 알았으니 마저 반복률이 큰 단어들을 뺀다면(위 코드) 이와 같은 결과가 나온다

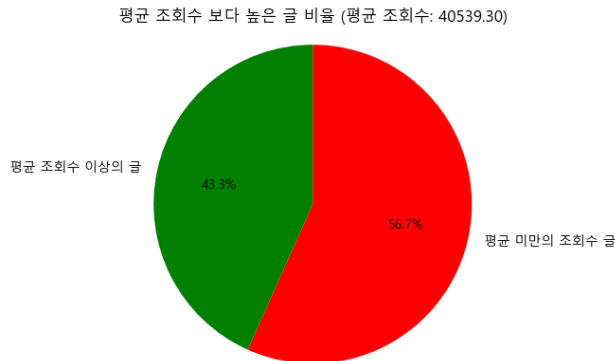

```

ratios = [above_average_ratio, below_average_ratio]
categories = ['평균 조회수 이상의 글', '평균 미만의 조회수 글']
colors = ['green', 'red']

# 그래프 그리기
plt.pie(ratios, labels=categories, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title(f'평균 조회수 보다 높은 글 비율 (평균 조회수: {average_views:.2f})')
plt.axis('equal') # 원이 왜곡되지 않도록 유지

# 그래프 표시
plt.show()

```



평균에 못 미치는 글이 56.7% 로
글간의 조회수 편차가 일정하지 않고 크다는 것을 알 수 있다

상위 100개 갤러리별 글 수 분포

```

import pandas as pd
import matplotlib.pyplot as plt

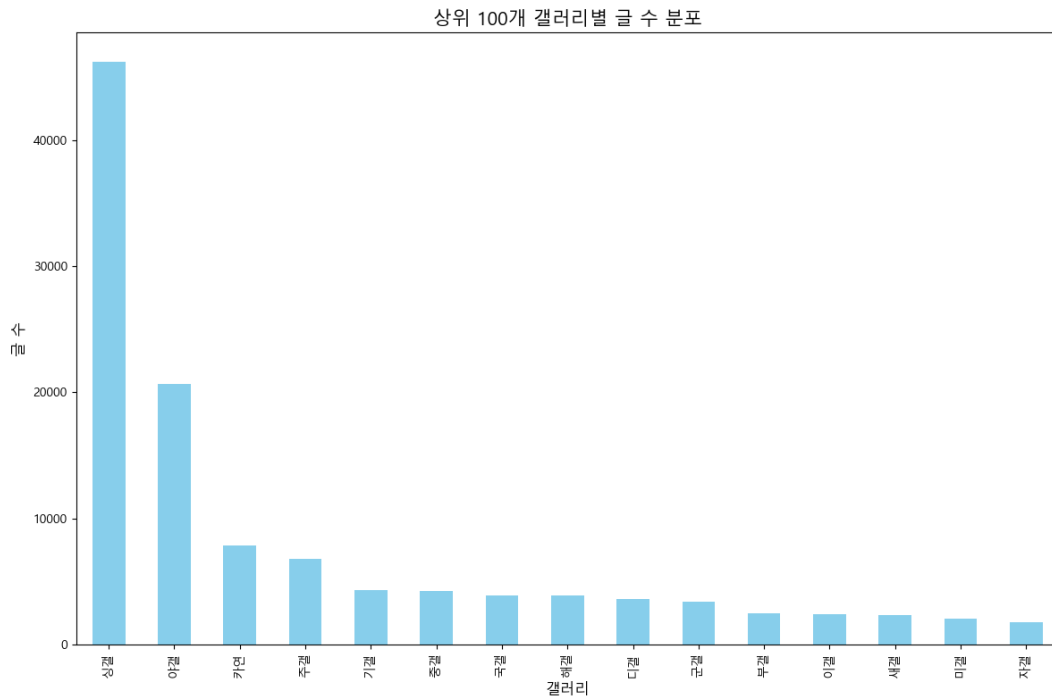
# 1. 데이터 불러오기
df = pd.read_csv('dcinside_data.csv')

# 2. 갤러리별 글 수 계산 (갤러리 열을 기준으로 카운트)
gallery_counts = df['갤러리'].value_counts()

# 3. 상위 100개 갤러리만 선택
top_100_galleries = gallery_counts.head(100)

# 4. 상위 100개 갤러리 글 수 분포도 시각화
plt.figure(figsize=(12, 8)) # 그래프 크기 설정
top_100_galleries.plot(kind='bar', color='skyblue')
plt.title('상위 100개 갤러리별 글 수 분포', fontsize=15)
plt.xlabel('갤러리', fontsize=12)
plt.ylabel('글 수', fontsize=12)
plt.xticks(rotation=90) # x축 레이블이 겹치지 않도록 90도로 회전
plt.tight_layout() # 레이아웃 조정 (x축 레이블이 잘리거나 겹치지 않게)
plt.show()

```



실시간 베스트의 글은

실제 반응이나 추천수와 관계없이 운영자가 개인적으로 선별한 글을 게시하며, 심지어 운영자가 마음대로 글 내용을 수정하거나 2개 이상의 글을 짜깁기하기도 한다

출처:

[https://namu.wiki/w/실시간 베스트 갤러리#s-4](https://namu.wiki/w/실시간_베스트_갤러리#s-4)

이렇기 때문에 운영자가 어느 갤러리에서 많이 글을 가져오는 지 볼 수 있는 결과이다

상위 50개 갤러리별 채택된글 수

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import BoundaryNorm

# 1. 데이터 불러오기
df = pd.read_csv('dcinside_data.csv')

# 2. 갤러리별 글 수 계산 (갤러리 열을 기준으로 카운트)
gallery_counts = df['갤러리'].value_counts()

# 3. 상위 50개 갤러리만 선택
top_30_galleries = gallery_counts.head(50)

# 4. 원의 크기 설정 (글 수에 비례)
sizes = top_30_galleries.values * 3 # 글 수에 비례해 크기 설정 (3을 곱해 크기 조정)

# 5. 랜덤 배치를 위한 x, y 좌표 설정
x_positions = np.random.rand(len(top_30_galleries)) * 100 # x 좌표를 랜덤하게 설정 (0~100 범위)
y_positions = np.random.rand(len(top_30_galleries)) * 100 # y 좌표를 랜덤하게 설정 (0~100 범위)

# 6. 색상 설정 (버블의 크기에 따라 채도가 다르게 설정)
# 색상은 원의 크기(sizes)에 비례하여 설정
```

```

boundaries = [700,1500,2500, 5000, 10000, 30000] # 오름차순으로 정렬된 구간
cmap = plt.get_cmap("viridis") # 색상 맵을 viridis로 변경 (눈에 띄는 색상)
norm = BoundaryNorm(boundaries, cmap.N) # BoundaryNorm을 사용해 구간을 설정

# 7. Bubble Chart 그리기
plt.figure(figsize=(12, 8)) # 그래프 크기 설정

# 8. 각 원 그리기 (크기와 색상 설정)
scatter = plt.scatter(x_positions, y_positions, s=sizes, c=sizes, cmap=cmap, alpha=0.6, edgecolor='black')

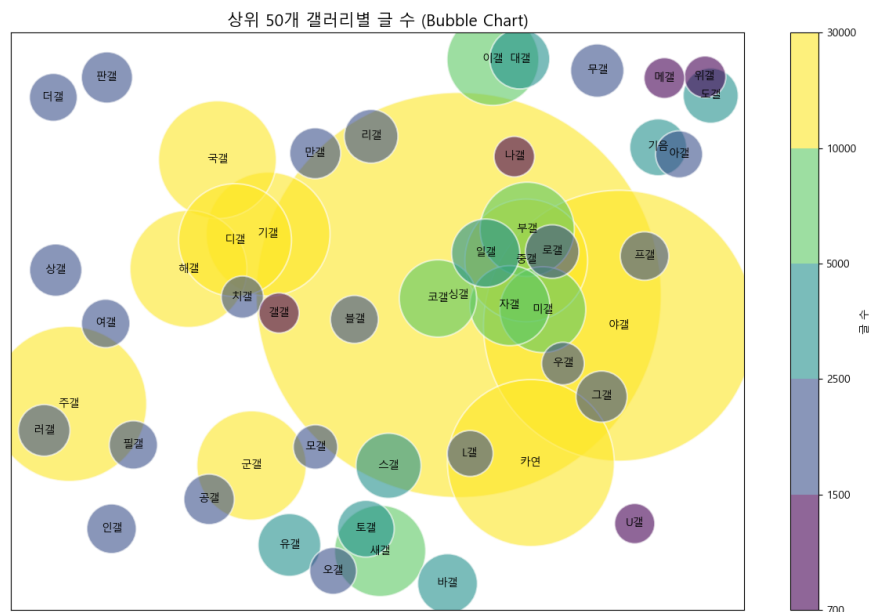
# 9. 색상 바 (컬러맵에 대한 범위 표시)
plt.colorbar(scatter, label='글 수')

# 10. 각 버블 안에 갤러리 이름 넣기
for i, (x, y, size, gallery) in enumerate(zip(x_positions, y_positions, sizes, top_30_galleries)):
    plt.text(x, y, gallery, ha='center', va='center', fontsize=10, color='black')

# 11. 그래프 설정
plt.title('상위 50개 갤러리별 글 수 (Bubble Chart)', fontsize=15)
plt.xticks([]) # x축 레이블 숨기기
plt.yticks([]) # y축 레이블 숨기기
plt.tight_layout() # 레이아웃 조정

# 12. 그래프 표시
plt.show()

```



위 결과 보다 조금 범위를 좁혀

갤러리 별 채택된 글 수를 비교하기 쉽게 만들었다

상위 50개 갤러리별 총조회수

```

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import BoundaryNorm

```

```

from matplotlib.ticker import FuncFormatter

# 1. 데이터 불러오기
df = pd.read_csv('dcinside_data.csv')

# 2. 갤러리별 조회수 합산
gallery_counts = df.groupby('갤러리')['조회수'].sum().sort_values(ascending=False)

# 3. 상위 50개 갤러리만 선택
top_50_galleries = gallery_counts.head(50)

# 4. 원의 크기 설정 (조회수에 비례)
sizes = top_50_galleries.values * 0.00005 # 조회수에 비례해 크기 설정 (크기 조정)

# 5. 랜덤 배치를 위한 x, y 좌표 설정
x_positions = np.random.rand(len(top_50_galleries)) * 100 # x 좌표를 랜덤하게 설정 (0~100 범위)
y_positions = np.random.rand(len(top_50_galleries)) * 100 # y 좌표를 랜덤하게 설정 (0~100 범위)

# 6. 색상 설정 (조회수에 따라 색상이 다르게 설정)
# 조회수 구간을 최대값(20억)과 최소값(천만)에 맞춰 설정
boundaries = [100000000, 500000000, 2000000000, 10000000000, 20000000000] # 조회수 구간 설정
cmap = plt.get_cmap("viridis") # 색상 맵 (viridis 사용)
norm = BoundaryNorm(boundaries, cmap.N) # BoundaryNorm을 사용해 구간을 설정

# 7. Bubble Chart 그리기
plt.figure(figsize=(12, 8)) # 그래프 크기 설정

# 8. 각 원 그리기 (크기와 색상 설정)
scatter = plt.scatter(x_positions, y_positions, s=sizes, c=top_50_galleries.values, cmap=cmap,

# 9. 색상 바 (컬러맵에 대한 범위 표시)
cbar = plt.colorbar(scatter, label='조회수')

# 10. colorbar 레이블을 일반 숫자 형식으로 변경
def format_ticks(x, pos):
    return f'{int(x):,}' # 천 단위 구분 기호를 추가하여 숫자 포맷

cbar.formatter = FuncFormatter(format_ticks)
cbar.update_ticks() # colorbar의 레이블 업데이트

# 11. 각 버블 안에 갤러리 이름 넣기
for i, (x, y, size, gallery) in enumerate(zip(x_positions, y_positions, sizes, top_50_galleries)):
    plt.text(x, y, gallery, ha='center', va='center', fontsize=10, color='black')

# 12. 최대, 최소 조회수 텍스트 추가 (제목 옆에 위치)
max_views = top_50_galleries.max()
min_views = top_50_galleries.min()

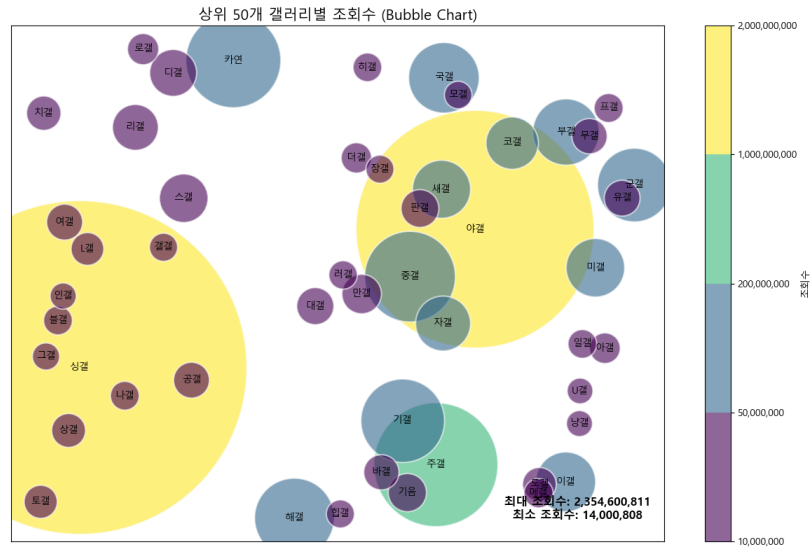
plt.text(90, 3, f'최대 조회수: {int(max_views):,}', ha='center', va='bottom', fontsize=12, color=
plt.text(90, 0.5, f'최소 조회수: {int(min_views):,}', ha='center', va='bottom', fontsize=12, colo

# 13. 그래프 설정
plt.title('상위 50개 갤러리별 조회수 (Bubble Chart)', fontsize=15)
plt.xticks([]) # x축 레이블 숨기기
plt.yticks([]) # y축 레이블 숨기기
plt.tight_layout() # 레이아웃 조정

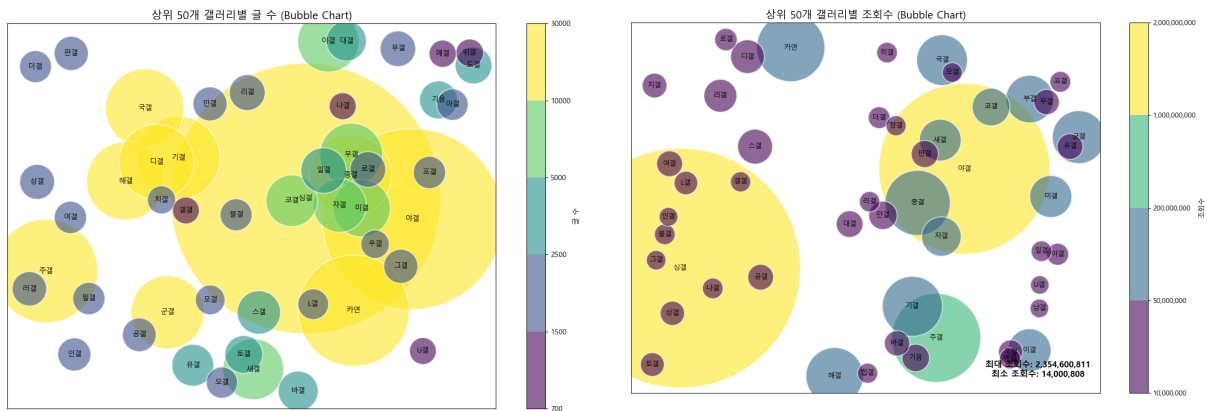
# 14. 그래프 표시
plt.show()

```

```
# 15. 상위 50개 갤러리 리스트를 print로 출력
print("상위 50개 갤러리 목록:")
for idx, gallery in enumerate(top_50_galleries.index, start=1):
    print(f"{gallery}")
```



위의 결과와 한번에 본다면



운영자에게 채택된 빈도수가 높다 해도 그에 비하는 조회수가 나오는 것은 아니라는 걸 볼 수 있다

EX) 주갤과 카연갤을 보면 카연갤은 주갤보다 높은 채택 수에 비해 조회수가 나오지 않는다

또한 싱갤과 야갤 이외 각 갤러리들의 채택 비율이 많이 낮다는 것을 알 수 있다

상위 50개 갤러리 날짜별 조회수 변화

상위 갤러리를 필터링 하기 위해 위 코드 마지막 부분에서

상위 50개 값을 확인 후 각 갤러리들을 직접 필터링 했다

시간에 흐름에 따라 나타낼 그래프이기에

dicinside_data.csv 의 날짜 컬럼을 사용해야 한다

근데 크롤링 했을 때 날짜 값이

이번 년도 외 작성된 글은 년.월.일 로

이번 년도에 작성된 글은 월.일 로

당일 작성된 글은 시:분 으로 저장되어 있다

datetime으로 형변환을 하기 위해

당일 작성 글은 포함시키지 않고(데이터 수가 전체 데이터에 비해 매우 적음)

월.일로 된 글들의 날짜를 24.월.일 포맷으로 바꿔주었다

```
import pandas as pd
import plotly.express as px
import plotly.io as pio
import numpy as np

# Plotly를 브라우저에서 출력하도록 설정
pio.renderers.default = "browser"

# 데이터프레임 읽기
df = pd.read_csv("dcinside_data.csv")
df = df.dropna() # 결측값이 있는 행 삭제

# 작성날짜 포맷 수정 (월.일 -> 24.월.일로 수정)
def fix_date_format(date):
    if ':' in date: # 시간(:)이 포함된 경우
        return "24." + date.split()[0][2:] # "24.11.12" 형식으로 변경
    elif len(date) == 5: # "월.일" 형식일 경우
        return "24." + date
    return date

df['작성날짜'] = df['작성날짜'].apply(fix_date_format)

# 작성날짜를 '년.월.일' 형식으로 변환, errors='coerce'를 추가하여 변환할 수 없는 값은 NaT로 처리
df['작성날짜'] = pd.to_datetime(df['작성날짜'], format='%y.%m.%d', errors='coerce')

# 변환할 수 없는 날짜가 있는 행 삭제
df = df.dropna(subset=['작성날짜'])

# 필터링할 갤러리 리스트
selected_galleries = [
    "싱갤", "야갤", "주갤", "카연", "중갤", "기갤", "해갤", "군갤", "국갤", "부갤",
    "이갤", "미갤", "새갤", "자갤", "코갤", "스갤", "디갤", "리갤", "만갤", "기음",
    "판갤", "대갤", "여갤", "유갤", "공갤", "무갤", "바갤", "치갤", "상갤", "도갤",
    "토갤", "L갤", "로갤", "아갤", "더갤", "메갤", "프갤", "나갤", "블갤", "일갤",
    "히갤", "러갤", "갤갤", "힙갤", "장갤", "모갤", "그갤", "인갤", "냥갤", "U갤"
]

# 조회수를 갤러리별로 합산하여 각 날짜별로 그룹화
df_grouped = df.groupby(['작성날짜', '갤러리'], as_index=False)['조회수'].sum()

# 선택된 갤러리들만 필터링
df_grouped_filtered = df_grouped[df_grouped['갤러리'].isin(selected_galleries)].copy()

print(df_grouped_filtered.head(40))

# 조회수를 로그 변환하여 더 작은 범위로 조정 (로그 변환 후 크기 조정)
# df_grouped_filtered['log_조회수'] = np.log1p(df_grouped_filtered['조회수'])

# Plotly로 Scatter 그래프 생성
fig = px.scatter(
    df_grouped_filtered,
    x='갤러리',
    y='조회수',
```

```

        color='갤러리',
        size='조회수', # 로그 변환된 조회수를 사용하여 점의 크기 결정
        animation_frame='작성날짜',
        animation_group='갤러리',
        title="갤러리별 조회수 변화 (상위 50개)",
        labels={"작성날짜": "작성 날짜", "조회수": "조회수"},
        size_max=50, # 점의 최대 크기 제한
    )

# 그래프를 HTML 파일로 저장 (웹에서 열 수 있도록)
fig.write_html("dcinside_top50_scatter_plot.html")

# 웹 브라우저에서 열기
fig.show()

```

x축에 갤러리를 두어 y축으로 당일의 조회수를 확인할 수 있다

비교를 통해 추가적으로 당일 실시간 베스트의 총 조회수나, 조회수가 높았던 날, 이슈가 된 갤러리를 확인 할 수 있겠다.

긍정도 비교

실시간 베스트의 긍정도

```

# 총 긍정도
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv('dcinside_data.csv')

# '긍정'이 1인 것과 0인 것의 비율 계산
positive_count = df[df['긍정'] == 1].shape[0] # 긍정이 1인 경우
negative_count = df[df['긍정'] == 0].shape[0] # 긍정이 0인 경우

# 긍정과 부정 비율 데이터프레임 생성
data = {
    '긍정/부정': ['긍정', '부정'],
    '비율': [positive_count, negative_count]
}

df_pie = pd.DataFrame(data)

# 원형 차트 생성
fig = px.pie(df_pie,
             names='긍정/부정',
             values='비율',
             title="실시간베스트글의 긍정/부정 비율",
             color='긍정/부정',
             color_discrete_map={'긍정': 'lightgreen', '부정': 'lightcoral'})

# 차트 출력
fig.show()

```



놀랍게도 부정적인 글이 대부분이다

운영자가 사람의 이목을 끄는데 부정적인 부분이 더 크게 끌린다는 것을 이용한 것 같다

싱글,야겔의 긍정도

```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv('dcinside_new_data.csv')

# '갤러리'가 '싱글'인 데이터만 필터링
df_singgal = df[df['갤러리'] == '싱글']

# 전체 글 수
total_count_singgal = df_singgal.shape[0]

# '긍정'이 1인 것과 0인 것의 비율 계산
positive_count_singgal = df_singgal[df_singgal['긍정'] == 1].shape[0] # 긍정이 1인 경우
negative_count_singgal = df_singgal[df_singgal['긍정'] == 0].shape[0] # 긍정이 0인 경우

# 비율 계산
positive_ratio_singgal = positive_count_singgal / total_count_singgal
negative_ratio_singgal = negative_count_singgal / total_count_singgal

# 긍정과 부정 비율 데이터프레임 생성
data_singgal = {
    '긍정/부정': ['긍정', '부정'],
    '비율': [positive_ratio_singgal, negative_ratio_singgal]
}

df_pie_singgal = pd.DataFrame(data_singgal)

# 원형 차트 생성
fig_singgal = px.pie(df_pie_singgal,
    names='긍정/부정',
    values='비율',
    title="싱글의 긍정/부정 비율",
    color='긍정/부정',
    color_discrete_map={'긍정': 'lightgreen', '부정': 'lightcoral'})

# 차트 출력
fig_singgal.show()

# '갤러리'가 '야겔'인 데이터만 필터링
df_singgal2 = df[df['갤러리'] == '야겔']
```



```

# 전체 글 수
total_count_singgal2 = df_singgal2.shape[0]

# '긍정'이 1인 것과 0인 것의 비율 계산
positive_count_singgal2 = df_singgal2[df_singgal2['긍정'] == 1].shape[0] # 긍정이 1인 경우
negative_count_singgal2 = df_singgal2[df_singgal2['긍정'] == 0].shape[0] # 긍정이 0인 경우

# 비율 계산
positive_ratio_singgal2 = positive_count_singgal2 / total_count_singgal2
negative_ratio_singgal2 = negative_count_singgal2 / total_count_singgal2

# 긍정과 부정 비율 데이터프레임 생성
data_singgal2 = {
    '긍정/부정': ['긍정', '부정'],
    '비율': [positive_ratio_singgal2, negative_ratio_singgal2]
}

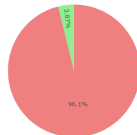
df_pie_singgal2 = pd.DataFrame(data_singgal2)

# 원형 차트 생성
fig_singgal2 = px.pie(df_pie_singgal2,
                      names='긍정/부정',
                      values='비율',
                      title="아갤의 긍정/부정 비율",
                      color='긍정/부정',
                      color_discrete_map={'긍정': 'lightgreen', '부정': 'lightcoral'})

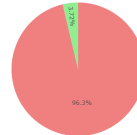
# 차트 출력
fig_singgal2.show()

```

상원의 긍정/부정 비율



아갤의 긍정/부정 비율



당연하게도 실시간 베스트의 긍정도가 낮기에 기여도가 큰 두 갤러리의 긍정도 역시 낮다

긍정적인 갤러리가 있을까?

긍정 비율이 가장 높은 30개 갤러리

```

import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv('dcinside_new_data.csv')

# 각 갤러리별 긍정과 부정 개수 계산
gallery_grouped = df.groupby('갤러리')['긍정'].value_counts().unstack(fill_value=0)

# 긍정 비율 계산 (긍정의 개수 / 총 개수)
gallery_grouped['긍정비율'] = gallery_grouped[1] / (gallery_grouped[0] + gallery_grouped[1])

```

```
# 각 갤러리의 전체 글 수 (긍정 + 부정)
gallery_grouped['전체글수'] = gallery_grouped[0] + gallery_grouped[1]

# 긍정 비율이 높은 갤러리 10개 선택
top_10_positive_galleries = gallery_grouped['긍정비율'].nlargest(30)

# 상위 10개 갤러리의 이름, 긍정 비율, 전체 글 수를 데이터프레임으로 변환
top_10_galleries_df = top_10_positive_galleries.reset_index()

# 전체 글 수 컬럼을 다시 추가
top_10_galleries_df['전체글수'] = top_10_galleries_df['갤러리'].map(gallery_grouped['전체글수'])

# 막대그래프 생성
fig = px.bar(top_10_galleries_df,
             x='갤러리',
             y='긍정비율',
             title='긍정 비율이 가장 높은 30개 갤러리',
             labels={'긍정비율': '긍정 비율', '갤러리': '갤러리'},
             text='전체글수', # 막대 위에 전체 글 수 표시
             hover_data=['전체글수']) # Hover 시 전체 글 수 표시

# 퍼센트 형태로 표시
fig.update_traces(texttemplate='%{text}', textposition='outside')

# y축을 백분율로 표시
fig.update_yaxes(tickformat='.2%', title_text='긍정 비율')

# 그래프 출력
fig.show()
```



막대 위에 나타난 숫자는 각 갤러리의 글이 채택된 숫자이다
채택되는 게시글이 늘어날 수록 긍정도가 바닥으로 떨어지는 것을 볼 수 있다

긍정 비율이 가장 높은 30개 갤러리 (표본글 100개 이상)

```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv('dcinside_new_data.csv')

# 각 갤러리별 글의 중복 횟수 계산
gallery_counts = df['갤러리'].value_counts()
```

```

# 중복이 100번 이상인 갤러리들만 선택
filtered_galleries = gallery_counts[gallery_counts >= 100].index

# 중복 100번 이상인 데이터만 필터링
df_filtered = df[df['갤러리'].isin(filtered_galleries)]

# 각 갤러리별 긍정과 부정 개수 계산
gallery_grouped = df_filtered.groupby('갤러리')['긍정'].value_counts().unstack(fill_value=0)

# 긍정 비율 계산 (긍정의 개수 / 총 개수)
gallery_grouped['긍정비율'] = gallery_grouped[1] / (gallery_grouped[0] + gallery_grouped[1])

# 각 갤러리의 전체 글 수 (긍정 + 부정)
gallery_grouped['전체글수'] = gallery_grouped[0] + gallery_grouped[1]

# 긍정 비율이 높은 갤러리 10개 선택
top_10_positive_galleries = gallery_grouped['긍정비율'].nlargest(30)

# 상위 10개 갤러리의 이름, 긍정 비율, 전체 글 수를 데이터프레임으로 변환
top_10_galleries_df = top_10_positive_galleries.reset_index()

# 전체 글 수 컬럼을 다시 추가
top_10_galleries_df['전체글수'] = top_10_galleries_df['갤러리'].map(gallery_grouped['전체글수'])

# 막대그래프 생성
fig = px.bar(top_10_galleries_df,
             x='갤러리',
             y='긍정비율',
             title='긍정 비율이 가장 높은 30개 갤러리 (표본글 100개 이상)',
             labels={'긍정비율': '긍정 비율', '갤러리': '갤러리'},
             text='전체글수', # 막대 위에 전체 글 수 표시
             hover_data=['전체글수']) # Hover 시 전체 글 수 표시

# 퍼센트 형태로 표시
fig.update_traces(texttemplate='%{text}', textposition='outside')

# y축을 백분율로 표시
fig.update_yaxes(tickformat='.2%', title_text='긍정 비율')

# 그래프 출력
fig.show()

```

긍정 비율이 가장 높은 30개 갤러리 (표본글 100개 이상)



실시간 베스트에 채택된 글 중 긍정 비율이 가장 높은 갤러리도 20%를 넘지 못한다

채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 (전체 글 수 많은 순)

```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv('dcinside_new_data.csv')

# 각 갤러리별 글의 중복 횟수 계산 및 상위 30개 갤러리 선택
top_30_galleries = df['갤러리'].value_counts().nlargest(30).index

# 상위 30개 갤러리만 필터링
df_top_30 = df[df['갤러리'].isin(top_30_galleries)]

# 각 갤러리별 긍정과 부정 개수 계산 및 긍정/부정 비율 추가
gallery_grouped = df_top_30.groupby('갤러리')['긍정'].value_counts().unstack(fill_value=0)
gallery_grouped['전체글수'] = gallery_grouped.sum(axis=1)
gallery_grouped['긍정비율'] = gallery_grouped[1] / gallery_grouped['전체글수']
gallery_grouped['부정비율'] = gallery_grouped[0] / gallery_grouped['전체글수']

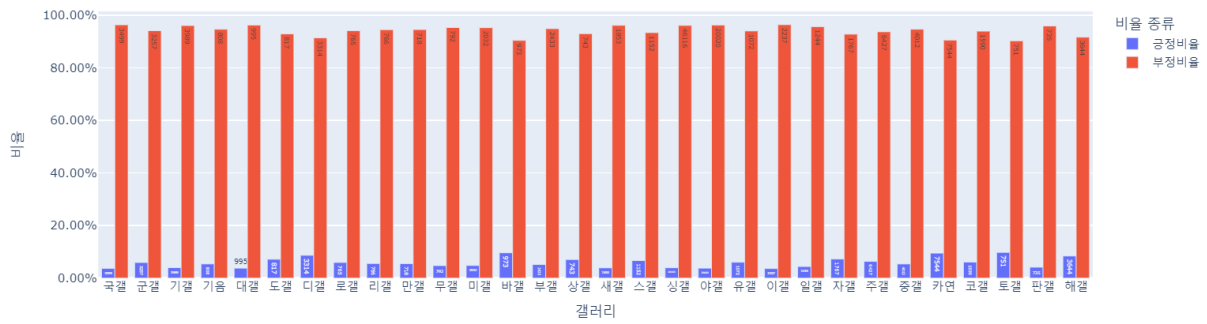
# 긍정 비율과 부정 비율 데이터를 함께 변환
gallery_grouped_melted = gallery_grouped[['긍정비율', '부정비율', '전체글수']].reset_index().melt(
    id_vars=['갤러리', '전체글수'],
    value_vars=['긍정비율', '부정비율'],
    var_name='비율종류',
    value_name='비율'
)

# 그래프 생성 (긍정/부정 비율 함께 표시)
fig = px.bar(gallery_grouped_melted,
             x='갤러리',
             y='비율',
             color='비율종류',
             barmode='group', # 막대를 나란히 배치
             title='채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 (전체 글 수 많은 순)',
             labels={'비율': '비율', '갤러리': '갤러리', '비율종류': '비율 종류'},
             text='전체글수', # 막대 위에 전체 글 수 표시
             hover_data=['전체글수']) # Hover 시 전체 글 수 표시

# y축을 백분율로 표시
fig.update_yaxes(tickformat='.2%', title_text='비율')

# 그래프 출력
fig.show()
```

채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 (전체 글 수 많은 순)



디지털 사진 갤러리와 , 바이크 갤러리, 카툰연재 갤러리, 토토갤, 해병대 갤러리에서 글 수 대비 상당한 긍정도를 가지고 있다

시간에 따른 채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 변화

```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df = pd.read_csv("dcinside_new_data.csv")
df = df.dropna() # 결측값이 있는 행 삭제

# 작성날짜 포맷 수정 (월.일 -> 24.월.일로 수정)
def fix_date_format(date):
    if ':' in date: # 시간(:)이 포함된 경우
        return "24." + date.split()[0][2:] # "24.11.12" 형식으로 변경
    elif len(date) == 5: # "월.일" 형식일 경우
        return "24." + date
    return date

df['작성날짜'] = df['작성날짜'].apply(fix_date_format)

# 작성날짜를 '년.월.일' 형식으로 변환, errors='coerce'를 추가하여 변환할 수 없는 값은 NaT로 처리
df['작성날짜'] = pd.to_datetime(df['작성날짜'], format='%y.%m.%d', errors='coerce')

# 상위 30개 갤러리 선택 (조회수 순으로)
top_30_galleries = df['갤러리'].value_counts().nlargest(30).index

# 상위 30개 갤러리만 필터링
df_top_30 = df[df['갤러리'].isin(top_30_galleries)]

# 긍정/부정 비율 계산
grouped = (
    df_top_30.groupby(['작성날짜', '갤러리', '긍정'])
    .size()
    .unstack(fill_value=0)
    .reset_index()
    .rename(columns={0: '부정', 1: '긍정'})
)
grouped['전체글수'] = grouped['긍정'] + grouped['부정']
grouped['긍정비율'] = grouped['긍정'] / grouped['전체글수']
```

```

grouped['부정비율'] = grouped['부정'] / grouped['전체글수']

# 애니메이션 그래프 생성
fig = px.bar(
    grouped,
    x='갤러리',
    y=['긍정비율', '부정비율'], # 긍정 비율과 부정 비율을 별도의 컬럼으로 표현
    color_discrete_map={'긍정비율': 'lightblue', '부정비율': 'lightcoral'}, # 색상 지정
    animation_frame='작성날짜', # 날짜별 애니메이션
    title='채택 횟수가 높은 30개 갤러리의 긍정/부정 비율 변화',
    labels={'count': '비율', '갤러리': '갤러리 이름'},
    height=600,
    text='전체글수', # 각 막대에 총 글 수 표시
    category_orders={'갤러리': top_30_galleries.tolist()} # X축의 갤러리 순서 고정
)

# 그래프의 텍스트로 총 글 수를 추가
fig.update_traces(texttemplate='%{text}', textposition='outside')

# 레이아웃 및 Y축 설정
fig.update_layout(
    xaxis_title='갤러리',
    yaxis_title='긍정/부정 비율',
    barmode='group', # 막대 그래프를 나란히 배치
    legend_title='비율 유형',
    yaxis=dict(
        tickformat='.2%', # 긍정 비율 표시
    ),
    yaxis2=dict(
        tickformat='.2%',
        overlaying='y',
        side='right', # 오른쪽에 두 번째 Y축
        showgrid=False,
    ),
)

# 그래프를 HTML 파일로 저장 (웹에서 열 수 있도록)
fig.write_html("yes_or_no_ratio.html")

# 그래프 출력
fig.show()

```

글이 많이 선택되는 30개 갤러리의 각 날짜별 올라온 게시물의 긍정도를 볼 수 있다

애니메이션을 본다면 부정적인 글이 너무 많이 올라와서 그렇지

생각보다 긍정적인 글이 올라오는 날이 많다는 것을 알 수 있다

실시간 베스트에 쓰인 단어들에 대한 시각

df_sort_by_view_count.csv or df_sort_by_word_count.csv 파일을 사용한 분석

```

df_sort_by_view_count.csv > data
영사, 중복횟수, 총조회수, 총추천수
3, 싱글형글, 30335, 1556983791.0, 16550106.0
62, 근황, 8135, 448334538.0, 5372012.0
17, 미유, 6549, 326775093.0, 4068801.0

```

```

df_sort_by_word_count.csv > data
영사, 중복횟수, 총조회수, 총추천수
13, 싱글형글, 30335, 1556983791.0, 16550106.0
62, 근황, 8135, 448334538.0, 5372012.0
17, 미유, 6549, 326775093.0, 4068801.0

```

는 총 조회수 기준으로 정렬

는 단어가 언급된 횟수를 중복횟수라 하여 정렬한 데이터이다

단어의 중복횟수, 총추천수, 총조회수 확인

총조회수, 총추천수, 중복횟수 비교 그래프

```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# 데이터프레임 읽기
df = pd.read_csv('df_sort_by_view_count.csv')

# 데이터가 제대로 로드되었는지 확인
print(df.head())

# 1. 총추천수가 가장 높은 20개 명사 (총추천수 기준으로 상위 20개 선택)
top_recommended = df.nlargest(20, '총추천수')

# 2. 총조회수가 가장 높은 20개 명사 (이미 정렬되어 있으므로, 상위 20개 선택)
top_viewed = df.nlargest(20, '총조회수')

#
# 총추천수, 총조회수, 중복횟수 비교 그래프 생성 (총조회수는 막대, 추천수와 중복횟수는 점)
fig = make_subplots(
    rows=1, cols=1,
    specs=[[{'secondary_y': True}]], # secondary y-axis를 사용
    subplot_titles=['조회수, 추천수, 중복횟수 비교 (상위 20개 명사)']
)

# 총조회수 (막대그래프)
fig.add_trace(
    go.Bar(
        x=top_viewed['명사'],
        y=top_viewed['총조회수'],
        name='총조회수',
        text=top_viewed['총조회수'], # 막대 위에 조회수 표시
    ),
    secondary_y=False, # 첫 번째 y축
)

# 총추천수 (점그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
        y=top_viewed['총추천수'],
        mode='markers',
        name='총추천수',
        marker=dict(size=10),
        text=top_viewed['총추천수'], # 점 위에 추천수 표시
    ),
    secondary_y=True, # 두 번째 y축
)

# 중복횟수 (점그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
        y=top_viewed['중복횟수'],
        mode='markers',
```

```

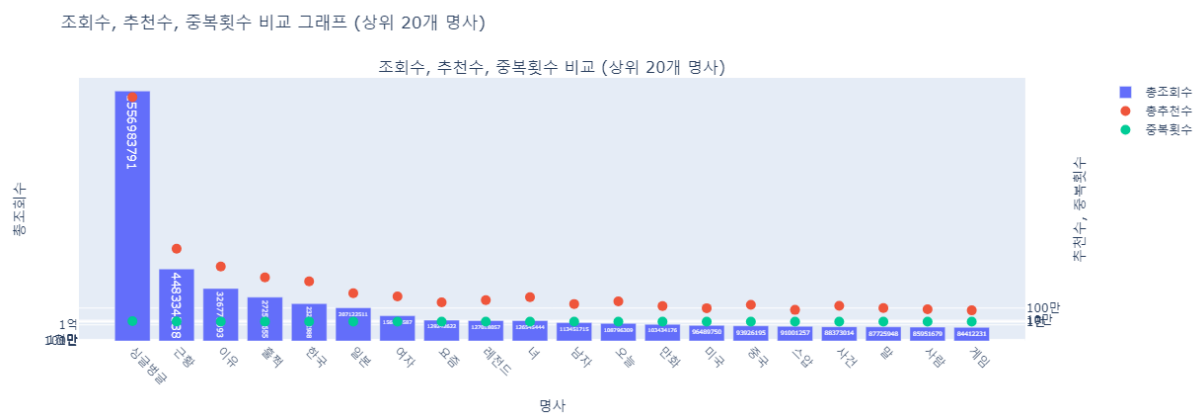
        name='중복횟수',
        marker=dict(size=10),
        text=top_viewed['중복횟수'], # 점 위에 중복횟수 표시
    ),
    secondary_y=True, # 두 번째 y축
)

# Layout 설정
fig.update_layout(
    title='조회수, 추천수, 중복횟수 비교 그래프 (상위 20개 명사)',
    xaxis_title='명사',
    yaxis_title='총조회수',
    showlegend=True,
    xaxis=dict(tickangle=45),
)

# 첫 번째 y축 (총조회수)
fig.update_yaxes(
    title_text='총조회수',
    secondary_y=False,
    tickmode='array',
    tickvals=[10000, 100000, 1000000, 10000000, 100000000], # 주요 값들에 대해 표시할 값 설정
    ticktext=['1만', '10만', '100만', '1천만', '1억'] # 각 값에 해당하는 텍스트 표시
)

# 두 번째 y축 (총추천수, 중복횟수)
fig.update_yaxes(
    title_text='추천수, 중복횟수',
    secondary_y=True,
    tickmode='array',
    tickvals=[1000, 10000, 100000, 1000000], # 주요 값들에 대해 표시할 값 설정
    ticktext=['1천', '1만', '10만', '100만'], # 각 값에 해당하는 텍스트 표시
)

```



싱글병글의 빈도수가 많은 만큼 조회수와 추천수가 다른 글들에 비해 높은 것을 볼 수 있다

위 그래프에서는 중복횟수를 판단하기 힘들기에 새로 만들었다

```

import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# 데이터프레임 읽기
df_w = pd.read_csv('df_sort_by_word_count.csv')

```



```

df_v = pd.read_csv('df_sort_by_view_count.csv')

# 데이터가 제대로 로드되었는지 확인
print(df.head())

top_viewed = df_v.nlargest(21, '총조회수')
top_viewed = top_viewed[1:21]

# 총추천수, 총조회수, 중복횟수 비교 그래프 생성 (서브플롯 추가)
fig = make_subplots(
    rows=2, cols=1, # 2개의 행, 1개의 열
    specs=[[{'secondary_y': True}], [{'secondary_y': False}]], # 첫 번째 플롯: secondary y축, 두
    subplot_titles=['조회수과 추천수 비교 (상위 20개 명사 - 상위 1개 제외)', '중복횟수 비교'],
    vertical_spacing=0.2 # 그래프 간의 간격 설정 (기본값은 0.1)
)

# 총조회수 (막대그래프)
fig.add_trace(
    go.Bar(
        x=top_viewed['명사'],
        y=top_viewed['총조회수'],
        name='총조회수',
        text=top_viewed['총조회수'], # 막대 위에 조회수 표시
        marker=dict()
    ),
    row=1, col=1, # 첫 번째 행, 첫 번째 열
    secondary_y=False, # 첫 번째 y축
)

# 총추천수 (점그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
        y=top_viewed['총추천수'],
        mode='markers',
        name='총추천수',
        marker=dict(size=10),
        text=top_viewed['총추천수'], # 점 위에 추천수 표시
    ),
    row=1, col=1, # 첫 번째 행, 첫 번째 열
    secondary_y=True, # 두 번째 y축
)

# 중복횟수 (굵은선 그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
        y=top_viewed['중복횟수'],
        mode='lines+markers',
        name='중복횟수',
        line=dict(color='green', width=2),
        marker=dict(size=8, color='green'),
        text=top_viewed['중복횟수'], # 점 위에 중복횟수 표시
    ),
    row=2, col=1 # 두 번째 행, 첫 번째 열
)

```

```

)

# 전체 레이아웃 설정
fig.update_layout(
    title='조회수, 추천수, 중복회수 비교',
    xaxis_title='명사',
    showlegend=True,
    height=800, # 전체 그래프 높이 조정
)

# 첫 번째 플롯의 y축 설정
fig.update_yaxes(
    title_text='총조회수',
    secondary_y=False,
    row=1, col=1
)

fig.update_yaxes(
    title_text='추천수',
    secondary_y=True,
    row=1, col=1
)

# 두 번째 플롯의 y축 설정 (중복회수 전용)
fig.update_yaxes(
    title_text='중복회수',
    row=2, col=1
)

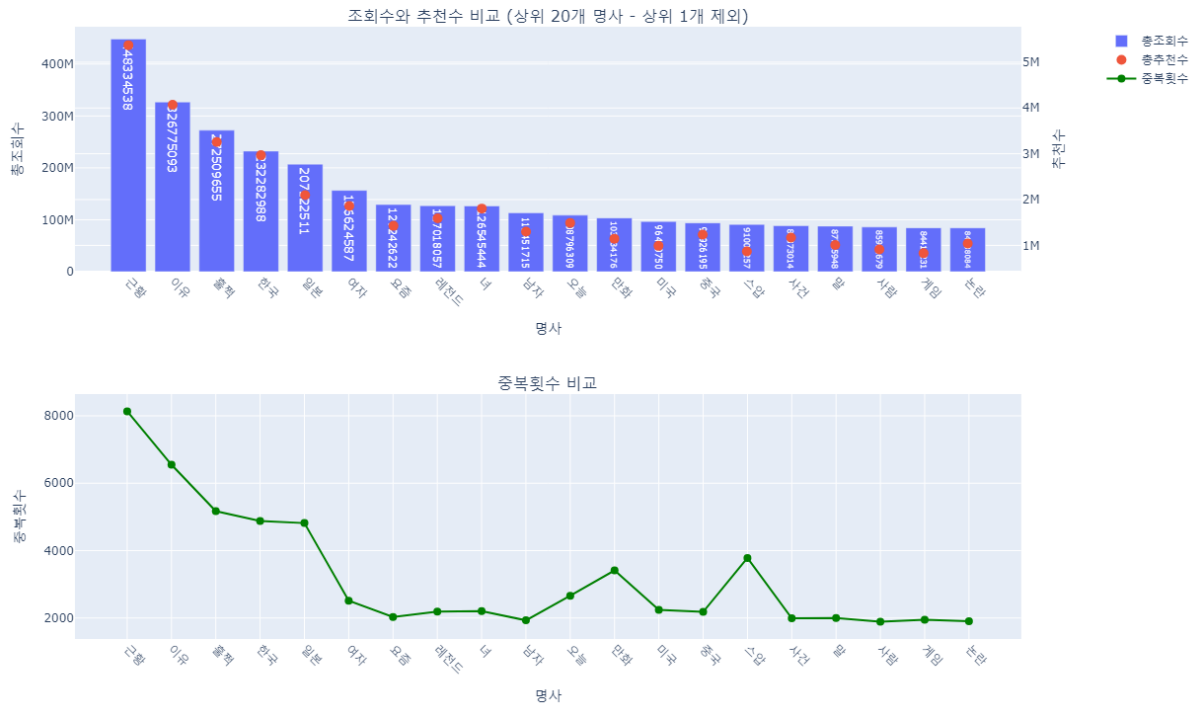
# x축 설정 (모든 플롯에 동일 적용)
fig.update_xaxes(
    title_text='명사',
    tickangle=45,
    row=1, col=1 # 첫 번째 플롯
)

fig.update_xaxes(
    title_text='명사',
    tickangle=45,
    row=2, col=1 # 두 번째 플롯
)

# 그래프 출력
fig.show()

```

조회수, 추천수, 중복횟수 비교



중복횟수에 따라 추천수와 조회수의 총량이 변하기는 하지만 그 비율의 차가 보인다

이 그래프에서 가성비 있는 단어는 여자,너 인 듯 하다

(비슷한 중복횟수 대비 조회수나 추천수가 높은 단어를 가성비 단어라 표현 했음)

줄 세우기

1. 중복횟수가 높은 40개 단어,
2. 중복횟수가 낮은 1000개 단어,
3. 총추천수가 높은 40개 단어,
4. 총추천수 낮은 40개 단어,
5. 총조회수 높은 40개 단어,
6. 총조회수 낮은 40개 단어,
7. 중복횟수 높은 40개 단어 (싱글벙글 제외),
8. 총추천수 높은 40개 단어 (싱글벙글 제외),
9. 총조회수 높은 40개 단어 (싱글벙글 제외)

```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df_w = pd.read_csv('df_sort_by_word_count.csv')
df_v = pd.read_csv('df_sort_by_view_count.csv')

# 3. 중복횟수가 가장 높은 20개 명사
top_duplicate = df_w.nlargest(41, '중복횟수')
row_duplicate = df_w.nsmallest(1000, '중복횟수')
```

```

top_viewed = df_v.nlargest(41, '총조회수')
row_viewed = df_v.nsmallest(40, '총조회수')

# 1. 총추천수가 가장 높은 20개 명사 (총추천수 기준으로 상위 20개 선택)
top_recommended = df_w.nlargest(41, '총추천수')
row_recommended = df_w.nsmallest(40, '총추천수')

# 중복횟수 높은 40개 단어
fig_duplicate = px.bar(
    top_duplicate,
    x='명사',
    y='중복횟수',
    title="중복횟수가 높은 40개 단어",
    labels={'명사': '단어', '중복횟수': '중복횟수'}
)
fig_duplicate.show()

# 중복횟수 낮은 1000개 단어
fig_duplicate = px.bar(
    row_duplicate,
    x='명사',
    y='중복횟수',
    title="중복횟수가 낮은 1000개 단어(1번만 나온 단어들이 너무 많기에 유의미한 결과 x)",
    labels={'명사': '단어', '중복횟수': '중복횟수'}
)
fig_duplicate.show()

# 총추천수 높은 40개 단어
fig_recommended = px.bar(
    top_recommended,
    x='명사',
    y='총추천수',
    title="총추천수가 높은 40개 단어",
    labels={'명사': '단어', '총추천수': '총추천수'}
)
fig_recommended.show()

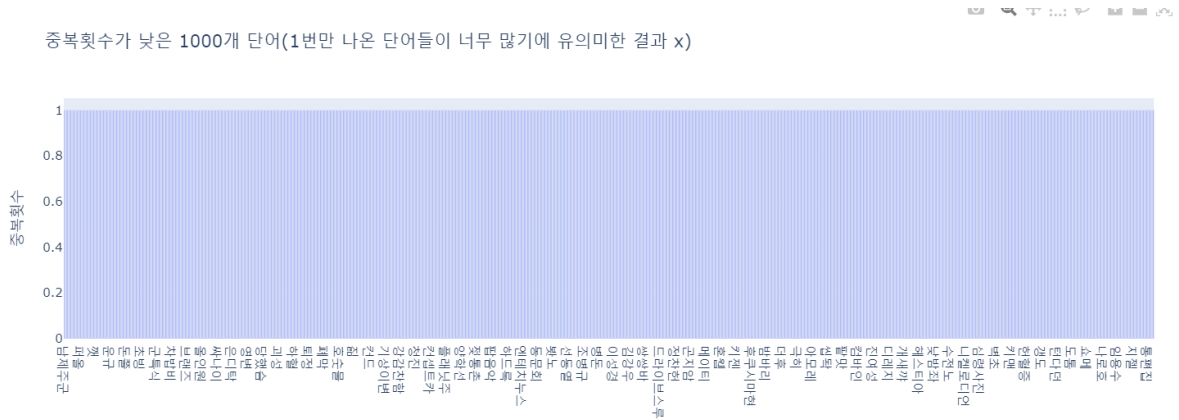
# 총추천수 낮은 40개 단어
fig_recommended = px.bar(
    row_recommended,
    x='명사',
    y='총추천수',
    title="총추천수가 낮은 40개 단어",
    labels={'명사': '단어', '총추천수': '총추천수'}
)
fig_recommended.show()

# 총조회수 높은 40개 단어
fig_viewed = px.bar(
    top_viewed,
    x='명사',
    y='총조회수',
    title="총조회수가 높은 40개 단어",
    labels={'명사': '단어', '총조회수': '총조회수'}
)
fig_viewed.show()

# 총조회수 낮은 40개 단어


```


2. 중복횟수가 낮은 1000개 단어



실시간 베스트 갤러리

설정 | 연관 갤러리(0/39) | 갤주소 복사 | 이용안내

전체글		공지		실시간 베스트		실베라이트		실베나이트	
번호	제목			글쓴이	작성일	조회	추천		
설문	인터넷 트렌드를 가장 빠르게 알고 있을 것 같은 스타는?			운영자	24/11/25	-	-		
AD	파워링크 광고 환경부품업체 조일환경 벨브 환경부품 풀링 데미스타 하이렉스 테라라이트 활성화 sus풀링 월타			등록안내▶ http://www.joillenv.co.kr					
AD	월타 G마켓 월타 전회원 10% 쿠폰 지급! 스마일배송 첫구매 시 15% 할인까지!			http://www.gmarket.co.kr					
273940		B [세경] 스압)코호케미 Rrr 사용 후기 겸 월타 해보기... [31]		고지		10.20	8586	2	
전체글									

전체글

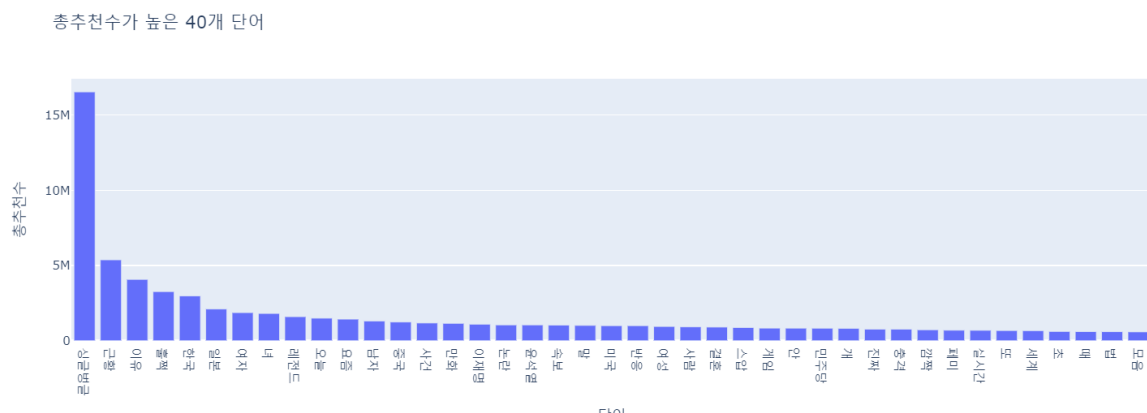
◀ 이전 검색 1 다음 검색 ▶

빠른 이동

게시물은 1만 개 단위로 검색됩니다.

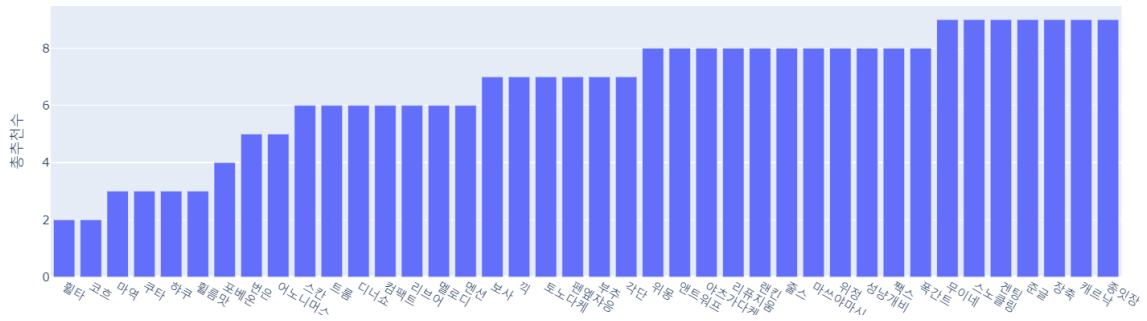
1000개의 단어가 다 X축에 표시되지는 않았지만 대다수의 단어가 일반인들은 알 수 없는 은어로 되어있음을 알 수 있다

3. 총추천수가 높은 40개 단어



4. 총추천수가 낮은 40개 단어

총추천수가 낮은 40개 단어



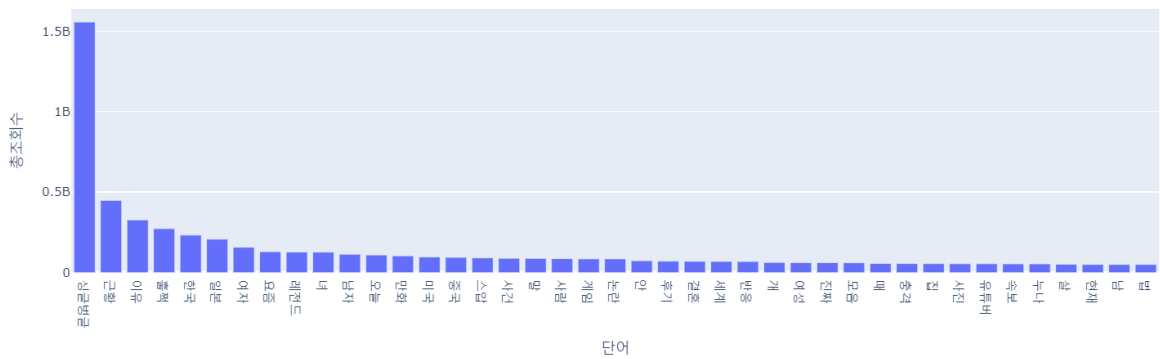
위에서 보았던 험타 단어가 들어간 게시물의 추천수는 2 이며

코흐케미라는 명칭은 코흐만 따로 분리된 것을 볼 수 있다

이 것으로 단어를 분리할 때 은어들이나 상업명 같은 단어들은 한 단어로 확실하게 붙었다 얘기하기 조금은 어렵다는 걸 알 수 있다

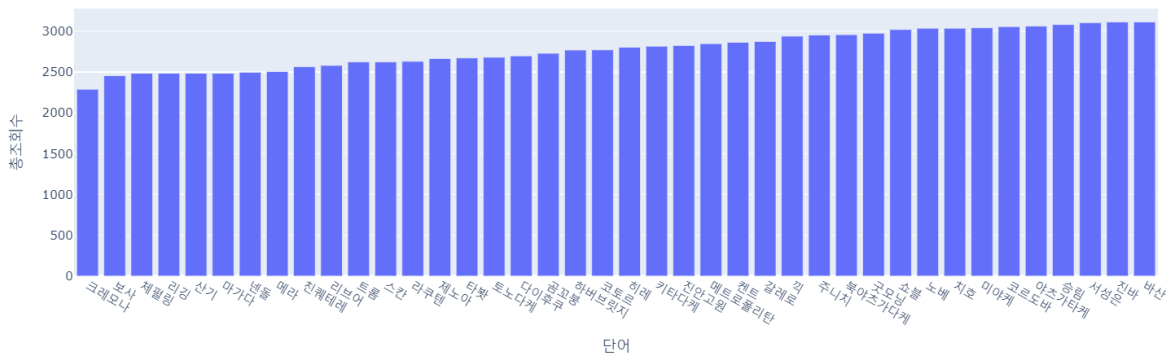
5. 총조회수가 높은 40개 단어

총조회수가 높은 40개 단어



6. 총조회수가 낮은 40개 단어

총조회수가 낮은 40개 단어

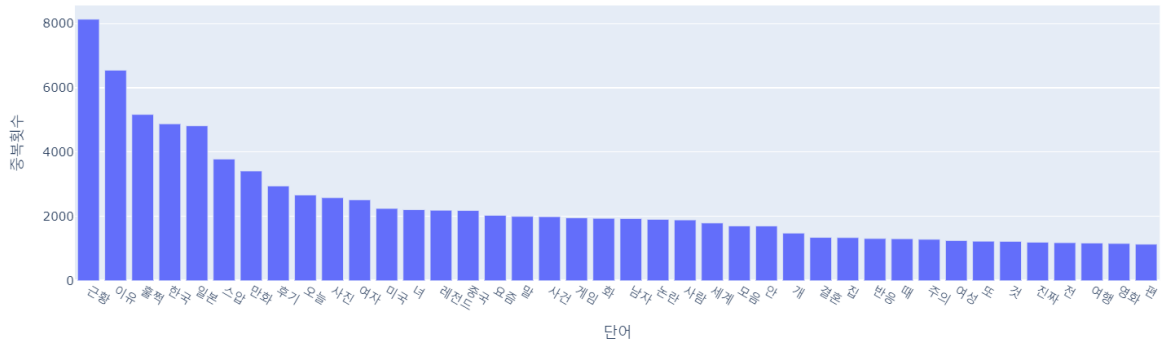


총 조회수가 높은 단어는 일반적으로 많이 사용되는 단어들이 나왔고

조회수가 낮은 단어는 추천수와 마찬가지로 알수 없는 은어들이 많이 있다

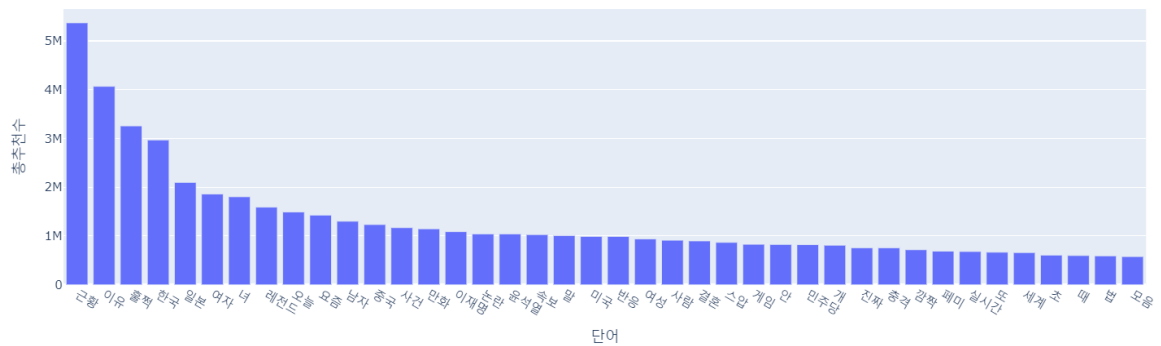
7. 중복횟수가 높은 40개 단어(싱글벙글 제외)

중복횟수가 높은 40개 단어(싱글병글 제외)



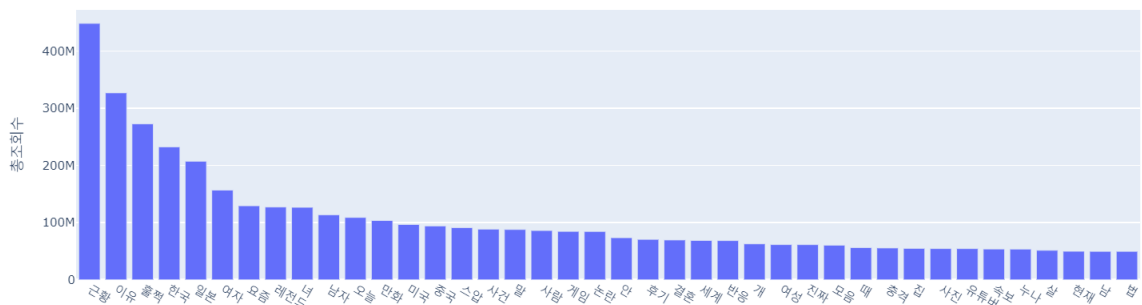
8. 총추천수가 높은 40개 단어(싱글병글 제외)

총추천수가 높은 40개 단어(싱글병글 제외)



9. 총조회수가 높은 40개 단어(싱글병글 제외)

총조회수가 높은 40개 단어(싱글병글 제외)



가장 많이 나오는 값인 싱글병글을 제외하고 본다면

그 전 그래프 보다 단어 간의 차이가 뚜렷하게 보인다

중복횟수가 1번인 명사와 2번 이상인 단어의 비율

```
import pandas as pd
import matplotlib.pyplot as plt

# 데이터프레임 읽기
df_v = pd.read_csv('df_sort_by_view_count.csv')
```



```

# 중복횟수가 1인 단어와 2번 이상인 단어 분리
count_1 = len(df_v[df_v['중복횟수'] == 1])
count_above_1_df = df_v[(df_v['중복횟수'] >= 2)]
count_above_1 = len(count_above_1_df)

# 전체 데이터 개수
total = len(df_v)

# '싱글병글' 단어에 대한 필터링
single_df = df_v[(df_v['명사'] == '싱글병글')]
single_count = single_df['중복횟수'].sum() # 중복횟수의 총합
singl_bungle_ratio = single_count / count_above_1_df['중복횟수'].sum() # 비율 계산

# 비율 계산 (전체 개수로 나누어서 비율 구하기)
count_1_ratio = count_1 / total
count_above_1_ratio = count_above_1 / total # 2번 이상에서 1번인 단어를 제외한 비율

# 원형 그래프에 사용할 데이터
labels = [f'1번인 단어\n({count_1}개)', f'2번 이상인 단어\n({count_above_1}개)']
sizes = [count_1_ratio, count_above_1_ratio]
colors = ['lightblue', 'lightcoral']

# 도넛 그래프 그리기
fig, ax = plt.subplots(figsize=(7, 7))

# 원형 그래프 그리기 (중앙을 비우기 위해 "wedgeprops" 사용)
wedges, texts, autotexts = ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, color

# 그래프 숫자 위치 조정 및 볼드 처리
for autotext in autotexts:
    autotext.set_fontsize(14) # 글씨 크기 조정
    autotext.set_fontweight('bold') # 글씨 두껍게
    autotext.set_color('black') # 글씨 색상 검정으로 설정

# '싱글병글' 비율을 나타내는 부가적인 원형 그래프 그리기 (외각에 추가)
ax_inset = fig.add_axes([0.7, 0.05, 0.25, 0.25]) # 위치를 오른쪽 아래로 조정 (x, y, width, height)
ax_inset.pie(
    [singl_bungle_ratio, 1 - singl_bungle_ratio],
    autopct='%1.1f%%',
    startangle=90,
    colors=['lightgreen', 'lightgray'],
    wedgeprops=dict(width=0.3)
)

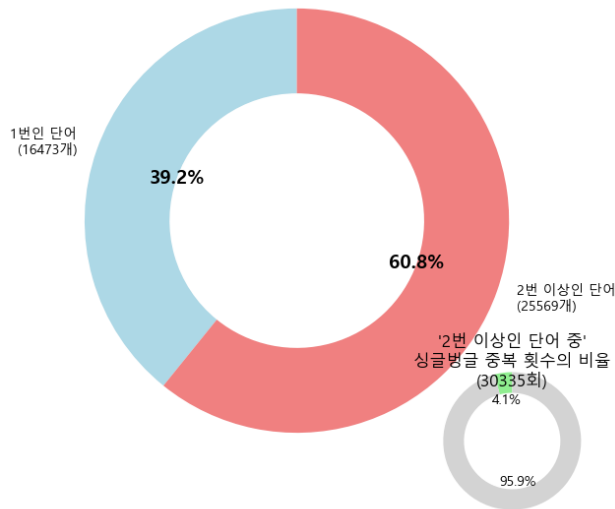
# 부가적인 원형 그래프에 이름 텍스트 추가
ax_inset.text(0, 1.15, f"'2번 이상인 단어 중'\n싱글병글 중복 횟수의 비율\n({single_count}회)", ha='cent

# 그래프 타이틀 설정
ax.set_title("중복횟수가 1번인 단어와 2번 이상인 단어의 비율", fontsize=16)

# 차트 출력
plt.show()

```

중복횟수가 1번인 단어와 2번 이상인 단어의 비율



중복된 적이 없는 단어가 상당히 많은 것을 볼 수 있다

중복 횟수를 다시 봐 보자

이미 중복 횟수대로 정렬 해줬던 데이터지만

인덱스 번호가 뒤죽박죽 이여서 그런지 데이터가 순서대로 나오지 않았기에

코드 상에서 중복횟수를 기준으로 다시 정렬을 했다

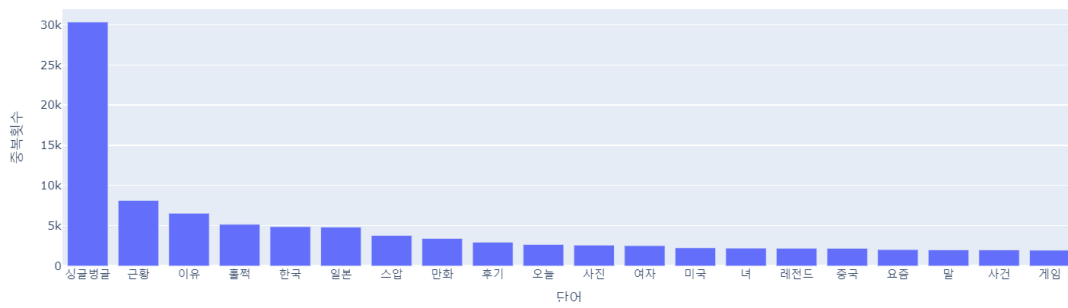
```
import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df_v = pd.read_csv('df_sort_by_view_count.csv')

# head 가 순서대로 정렬된 데이터를 가져오지 않음
# 그래서 결국 재정렬해야함
top_duplicate = df_v.nlargest(20, '중복횟수')

fig_duplicate = px.bar(
    top_duplicate,
    x='명사',
    y='중복횟수',
    title="중복횟수가 높은 20개 명사(1~19)",
    labels={'명사': '명사', '중복횟수': '중복횟수'}
)
fig_duplicate.show()
```

중복횟수가 높은 20개 단어(1~19)



```
import pandas as pd
import plotly.express as px

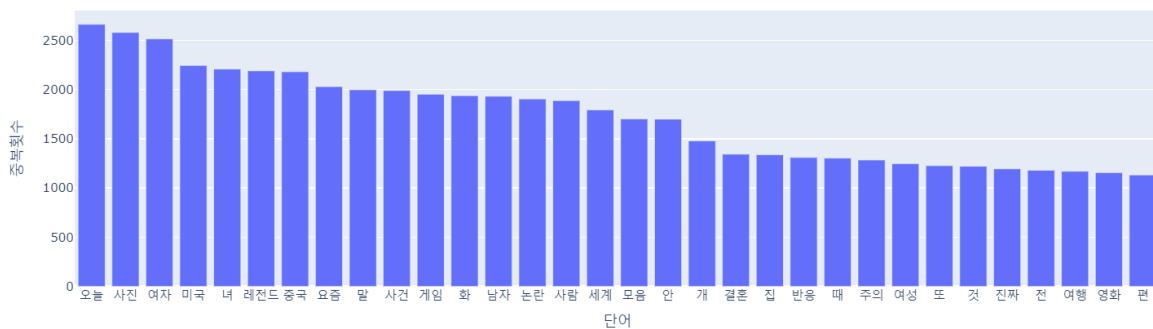
# 데이터프레임 읽기
df_v = pd.read_csv('df_sort_by_view_count.csv')

# head 가 순서대로 정렬된 데이터를 가져오지 않음
# 그래서 결국 재정렬해야함
top_duplicate = df_v.nlargest(41, '중복횟수')

top_10_41_duplicate = top_duplicate[9:41]

fig_duplicate = px.bar(
    top_10_41_duplicate,
    x='명사',
    y='중복횟수',
    title="중복횟수가 높은 40개 단어(10~41)",
    labels={'명사': '단어', '중복횟수': '중복횟수'}
)
fig_duplicate.show()
```

중복횟수가 높은 40개 단어(10~41)



상위 8 위 부터는 비슷 비슷한 중복 횟수를 가지고 있는 것 처럼 보였는데
 그 밑에서 중복 횟수가 더 낮아지는 것들을 볼 수 있다
 (오늘과 게임의 언급 횟수는 5백번이 넘게 차이가 난다)

```
import pandas as pd
import plotly.express as px
```

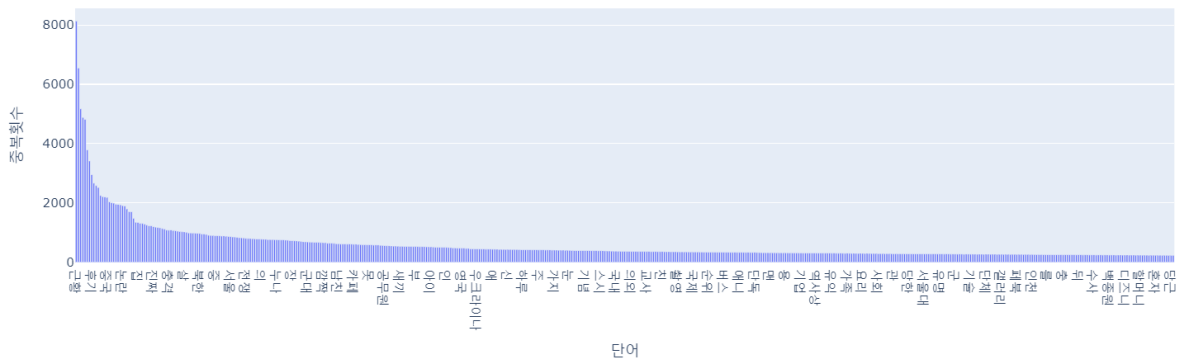
```
# 데이터프레임 읽기
df_v = pd.read_csv('df_sort_by_view_count.csv')

# head 가 순서대로 정렬된 데이터를 가져오지 않음
# 그래서 결국 재정렬해야함
top_duplicate = df_v.nlargest(500, '중복횟수')

top_20_40_duplicate = top_duplicate[1:500]

fig_duplicate = px.bar(
    top_20_40_duplicate,
    x='명사',
    y='중복횟수',
    title="중복횟수가 높은 500개 단어(1등 싱글벙글 제외)",
    labels={'명사': '단어', '중복횟수': '중복횟수'}
)
fig_duplicate.show()
```

중복횟수가 높은 500개 단어(1등 싱글벙글 제외)



그 이하로는 중복되는 횟수가 비슷해 보이며

상위권 단어들은 거의 글을 쓰는데 들어가는 포맷이여 위치하는 경우가 많고

(싱글벙글 ~) 또는 (~근황)

[상겅] 싱글벙글 기혼페미vs비혼페미 내전중언 여시 [303]
 [상겅] 싱글벙글 아프리카세계촌 1201 [21]
 [상겅] 싱글벙글 기간84 "아빠들이 스트레스 받는거 ... [306]
 [아겅] 경제는 자갑다면 윤루카스 근황 드드드드 [541]
 [상겅] 싱글벙글 챗GPT한테 말 거는 미주갤러들 [145]
 [상겅] 싱글벙글 모닝지구촌 (12/1) [20]
 [상겅] 싱글벙글 일본에서 한국인인걸 들켜버린 비류... [391]
 [상겅] 싱글벙글 한 여성이 아기를 안 낳겠다고 하는 ... [1067]
 [상겅] 싱글벙글 지상팔이 후배 군기 잡는 방법 [180]
 [상겅] 싱글벙글 토머의 자고 청소... [94]

284633 [합겅] 마이클 잭슨 사후 수익 현재 드드드드 [14]
 284557 [중겅] 안리얼5연진 차용한 중국계임 드드 [336]
 284535 [미겅] 1997년 대한민국 외환위기 황동 대장 드드그
 284519 [부겅] 인천국제공항 드드드 [373]
 284455 [주겅] 공무원 제설작업에 대한 어루 번들 + 성남 제...
 284447 [아겅] 국적 DMZ에 있다는 소말리 드드.jpg [315]

글을 쓰는 당일을 언급하는 오늘날이나 그 상황을 설명하기 위한 단어들이 많다

그 뒤로는 조금만 순위가 내려가도 언급되는 횟수가 줄어들고

정치,국가, 갈등 소재(여자,남자) 등의 단어나

순위	명칭	분류	주제	글리젠(월)	검색 이용자 남녀성비
-	다시인사이드	갤러리 전체	종합	약 3,000만(개)	남성(51.8%)
1	국내야구	갤러리	자유게시판(남초)	917,770(개)	남성(91.3%)

싱글 다음의 채택되는 글이 많은 야갤이 남자들이 주로 사용하는 사이트기에
(여자,누나,너)같은 타 성별의 키워드들이 언급이 많이 되는 것으로 보인다

야구갤러리를 야갤이라 줄여 말하며 싱글빙글 갤러리가 야갤의 뒤를 잇는 디시인사이드의 수도로 이용되고 있
기에 싱글의 성비도 비슷하다 추측했고 수도는 이용하는 사람이 가장 많은 갤러리를 칭한다

그 외에도 (게임)(사진)등 다양한 주제가 오가는 것을 알 수 있다

각 단어별 중복횟수,추천수,조회수 확인

각 단어별 중복횟수,추천수,조회수 확인을 통해

단어의 힘(게시글을 누르게하는 요인)을 직접적으로 확인 할 수 있는 파트라 생각한다

각 단어별 조회수, 추천수, 중복횟수 비교

```
import pandas as pd
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# 데이터프레임 읽기
df_w = pd.read_csv('df_sort_by_word_count.csv')
df_v = pd.read_csv('df_sort_by_view_count.csv')

# 데이터가 제대로 로드되었는지 확인
print(df_v.head())

# 상위 21개 단어 선택 (1개 제외)
top_viewed = df_v.nlargest(20, '총조회수')

# 총조회수와 총추천수를 각각 총중복횟수로 나누기
top_viewed['조회수_대_중복횟수'] = top_viewed['총조회수'] / top_viewed['중복횟수']
top_viewed['추천수_대_중복횟수'] = top_viewed['총추천수'] / top_viewed['중복횟수']

# 총추천수, 총조회수, 중복횟수 비교 그래프 생성 (서브플롯 추가)
fig = make_subplots(
    rows=2, cols=1, # 2개의 행, 1개의 열
    specs=[[{'secondary_y': True}], [{'secondary_y': False}]], # 첫 번째 플롯: secondary y축, 두
    subplot_titles=['조회수와 추천수 비교 (상위 20개 단어 - 상위 1개 제외)', '중복횟수 비교'],
    vertical_spacing=0.2 # 그래프 간의 간격 설정 (기본값은 0.1)
)

# 조회수 / 중복횟수 (막대그래프)
fig.add_trace(
    go.Bar(
        x=top_viewed['명사'],
        y=top_viewed['조회수_대_중복횟수'],
        name='조회수/중복횟수',
        text=top_viewed['조회수_대_중복횟수'], # 막대 위에 계산된 값 표시
        marker=dict()
    ),
    row=1, col=1, # 첫 번째 행, 첫 번째 열
    secondary_y=False, # 첫 번째 y축
)

# 추천수 / 중복횟수 (점그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
```

```

        y=top_viewed['추천수_대_중복횟수'],
        mode='markers',
        name='추천수/중복횟수',
        marker=dict(size=10),
        text=top_viewed['추천수_대_중복횟수'], # 점 위에 계산된 값 표시
    ),
    row=1, col=1, # 첫 번째 행, 첫 번째 열
    secondary_y=True, # 두 번째 y축
)

# 중복횟수 (굵은선 그래프)
fig.add_trace(
    go.Scatter(
        x=top_viewed['명사'],
        y=top_viewed['중복횟수'],
        mode='lines+markers',
        name='중복횟수',
        line=dict(color='green', width=2),
        marker=dict(size=8, color='green'),
        text=top_viewed['중복횟수'], # 점 위에 중복횟수 표시
    ),
    row=2, col=1 # 두 번째 행, 첫 번째 열
)

# 전체 레이아웃 설정
fig.update_layout(
    title='조회수, 추천수, 중복횟수 비교',
    xaxis_title='단어',
    showlegend=True,
    height=800, # 전체 그래프 높이 조정
)

# 첫 번째 플롯의 y축 설정
fig.update_yaxes(
    title_text='조회수/중복횟수',
    secondary_y=False,
    row=1, col=1
)

fig.update_yaxes(
    title_text='추천수/중복횟수',
    secondary_y=True,
    row=1, col=1
)

# 두 번째 플롯의 y축 설정 (중복횟수 전용)
fig.update_yaxes(
    title_text='중복횟수',
    row=2, col=1
)

# x축 설정 (모든 플롯에 동일 적용)
fig.update_xaxes(
    title_text='단어',
    tickangle=45,
    row=1, col=1 # 첫 번째 플롯
)

fig.update_xaxes(

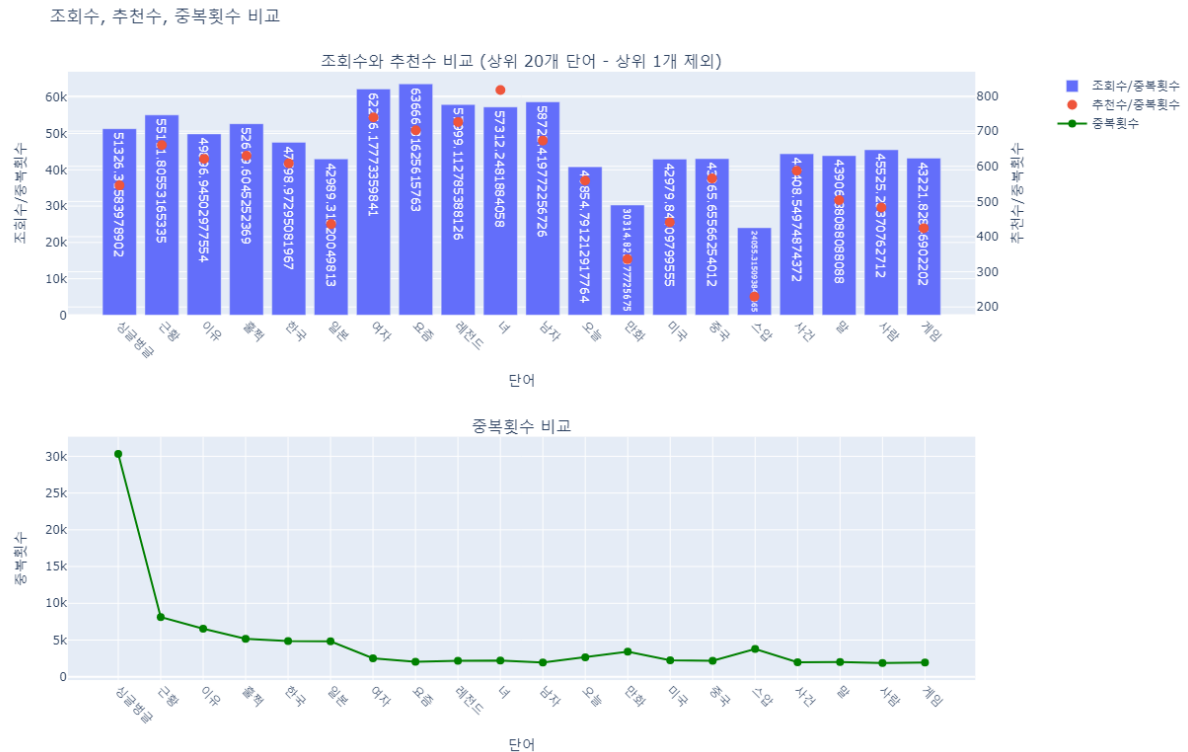
```

```

title_text='단어',
tickangle=45,
row=2, col=1 # 두 번째 플롯
)

# 그래프 출력
fig.show()

```



총합으로 봤을 때 강세를 가지던 싱글벙글이
 높은 언급 대비 조회수가 그렇게 까지 많이 나오지 못하는 단어로
 각각의 단어로 봤을 때 힘을 쓰지 못하는 것을 알 수 있다
 또한 스압이라는 단어도 비슷한 중복횟수 단어들과 다르게 조회수와 추천수에서 모두 낮은 기록을 하고 있는 것이 보인다
 그에 비해
 요즘,레전드,여자,너,남자 라는 단어는 중복되는 횟수에 비하여 엄청나게 많은 조회수를 불러오는 것으로 보인다

줄세우기

1. 평균추천수가 높은 40개 단어
2. 평균추천수가 낮은 40개 단어
3. 평균조회수가 높은 40개 단어
4. 평균조회수가 낮은 40개 단어

```

import pandas as pd
import plotly.express as px

# 데이터프레임 읽기
df_w = pd.read_csv('df_sort_by_word_count.csv')
df_v = pd.read_csv('df_sort_by_view_count.csv')

```

```

df_v['평균조회수'] = df_v['총조회수'] / df_v['중복횟수']
df_v['평균추천수'] = df_v['총추천수'] / df_v['중복횟수']

df_w['평균조회수'] = df_w['총조회수'] / df_w['중복횟수']
df_w['평균추천수'] = df_w['총추천수'] / df_w['중복횟수']

top_viewed = df_v.nlargest(40, '평균조회수')
row_viewed = df_v.nsmallest(40, '평균조회수')

# 1. 평균추천수가 가장 높은 40개 명사 (평균 추천수 상위 40 기준으로 정렬)
top_recommended = df_w.nlargest(40, '평균추천수')
row_recommended = df_w.nsmallest(40, '평균추천수')

fig_recommended = px.bar(
    top_recommended,
    x='명사',
    y='평균추천수',
    title="평균추천수가 높은 40개 단어",
    labels={'명사': '단어', '평균추천수': '평균추천수'}
)
fig_recommended.show()

fig_recommended = px.bar(
    row_recommended,
    x='명사',
    y='평균추천수',
    title="평균추천수가 낮은 40개 단어",
    labels={'명사': '단어', '평균추천수': '평균추천수'}
)
fig_recommended.show()

fig_viewed = px.bar(
    top_viewed,
    x='명사',
    y='평균조회수',
    title="평균조회수가 높은 40개 단어",
    labels={'명사': '단어', '평균조회수': '평균조회수'}
)
fig_viewed.show()

fig_viewed = px.bar(
    row_viewed,
    x='명사',
    y='평균조회수',
    title="평균조회수가 낮은 40개 단어",
    labels={'명사': '단어', '평균조회수': '평균조회수'}
)
fig_viewed.show()

```

1. 평균추천수가 높은 40개 단어

직업	평균 자녀 수
관료	7000
이성명	6900
이계선	6500
한우	5800
김철	5200
다복	4900
이재현	4100
유지	4000
유지	3900
김민	3800
김민	3700
김민	3600
김민	3500
김민	3400
김민	3300
김민	3200
김민	3100
김민	3000
김민	2900
김민	2800
김민	2700
김민	2600
김민	2500
김민	2400
김민	2300
김민	2200
김민	2100
김민	2000
김민	1900
김민	1800
김민	1700
김민	1600
김민	1500
김민	1400
김민	1300
김민	1200
김민	1100
김민	1000
김민	900
김민	800
김민	700
김민	600
김민	500
김민	400
김민	300
김민	200
김민	100
김민	0

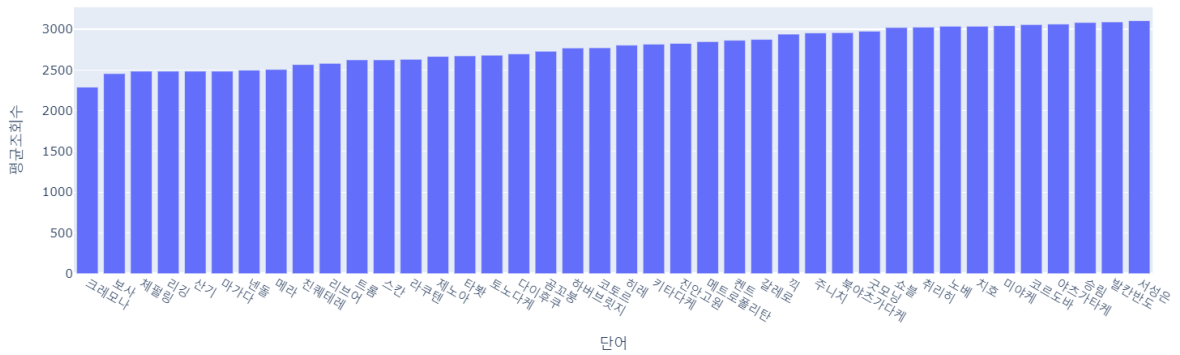
2. 평균추천수가 낮은 40개 단어

[illegible][illegible]

뒤로는 성적인 비속어들이나 인플루언서들이 많이 나왔다

49

평균 조회수가 낮은 40개 단어



평균 조회수가 낮은 단어들은 역시 평균추천수와 같이 은어나 특정 인물 사물의 이름이 많다

결론

자신의 글이 운영자에게 채택되어 실시간 베스트에 올라오게 하고 싶은 사람은

실시간 갤러리의 거의 모든 글에 사진이 첨부되어 있음에도 제목에 .jpg를 붙여 채택 가능성을 높이자

또한 싱글빙글 갤러리에 글을 올리는 것이 유리하며 글을 올릴 때 무조건 앞에는 싱글빙글을 붙이며 ~ 근황.jpg로 끝나는 글을 쓰는 것이 최고라 생각한다

채택된 뒤 조회수까지 생각한다면 선정적이거나 자극적인 단어 또는 유명인 이 들어가게끔 하며

소수의 사람들만이 알고있는 은어나 단어들은 절대 사용하지 않는다

pxxdx tv 유명한 여성 인플루언서 근황.jpg → 앞으로도 높은 조회수를 얻을 수 있을 것으로 예상된다

아니라면 시대 상황에 맞는 적절한 특정 단어들을 제목에 넣는다면 높은 조회수를 기대할 수 있겠다

다양한 인사이트를 주는 실시간 베스트지만 거의 올라오는 글 대부분이 자극적이고 부정적인 글들이 올라오기에

건강한 웹 커뮤니티 사용을 위해 실시간 베스트를 이용하는 것 보단 각각의 성향에 맞는 갤러리 또는 커뮤니티를 이용하는 게 좋을 것 같다