Final Project 보고서 – Clear communication

전공: 아트엔테크놀로지 학년: 4학년 학번: 20191098 이름: 백승주

1. **프로젝트 개요**

세 명의 조원들로 구성하여 하나의 주제를 선정해 Data visualization 한다. Data visualization은 세 가지 형태로 구성된다. Clear communication, UnConvetional charting type, 그리고 Analog material incorportated 다. Clear communication은 명확하게 알아볼 수 있는 형태고 unconventional chatrting type 은 직관적인 정보 전달보다는 data 에 의미를 담아 전달할 수 있게 한다. Analog data 무엇이든 될 수 있다. 소리, 맛, 촉감 등 무엇이든 아날로그적 형태로 data 를 표현한다.// 다. 팀원들과 협의 결과 .최근 늘어나고 있는 혐오 표현에 대한 이야기가 나왔다. 실제로 많은 혐오 표현들과 문제점들에 대해서는 조원들이 모두 공감을 표했다.

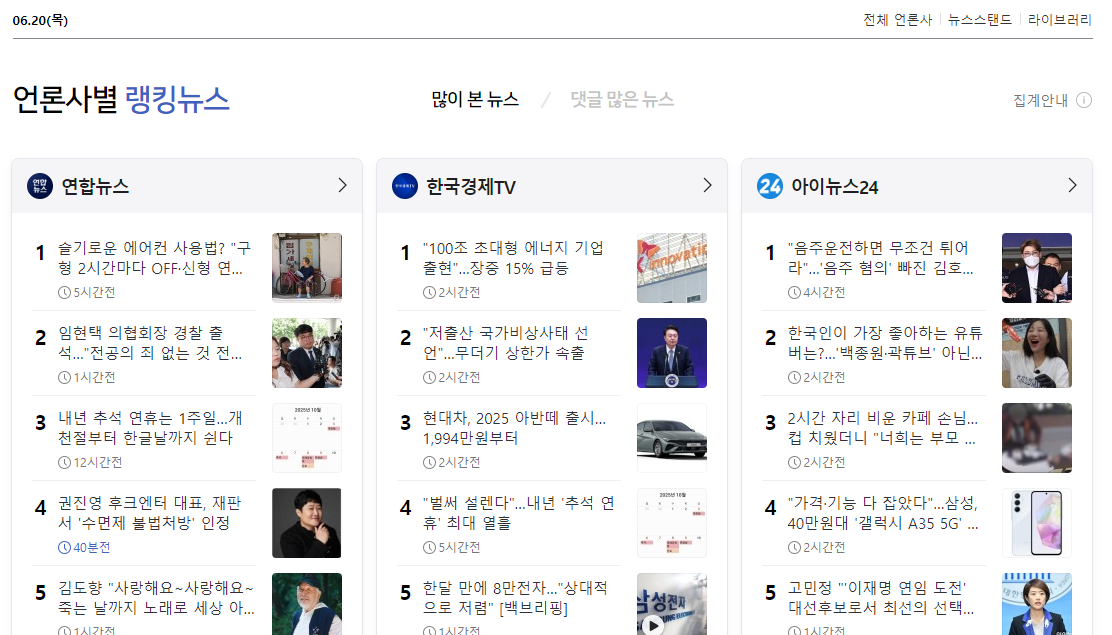
1. **Dataset 추출 -웹 크롤링**
2. **요구되는 Dataset의 조건**
3. 주어진 data를 통해 과거에 비해 얼마나 혐오 표현들이 늘어났는지 확인해야 한다.
4. 혐오 표현 중 어떤 종류의 혐오 대상별로 분류를 해보자.
5. **Dataset – 뉴스 댓글**

그렇다면 늘어나는 혐오 표현들은 어떤 dataset을 통해 조사할 수 있을 것이까. 우리가 선정해야될 data를 통해 관찰하고자 하는 내용들은 다음과 같았다.

1의 조건을 만족하는 ㅎdataset으로 뉴스 댓글이 효과적이었다. 뉴스 댓글, 뉴스 댓글에느는 혐오 표현들이 많다. 특히 사회 뉴스에서는 심심찮게 찾아볼 수 있다. 뉴스 댓글은 효과적으로 혐오 표현이 늘어났는지를 찾아볼 수 있는 data 였다. 그러나 구글링을 해본 결과 원하는 뉴스 댓글들을 모아놓은 data는 찾을 수가 없었다. 하는 수 없이

1. **뉴스 댓글 기준 정하기 – Best comment**
2. **파이썬 코드 – 베스트 댓글 가져오기**

그렇다고; 대략 2000년부터 2024년까지의 모든 뉴스 댓글의 data를 가져오는 것은 현실적으로는 불가능한 작업이었다. 객관적인 기준을 정해 해당 기준에 맞는 데이터들을 선별적으로 가져와야 했다.. 이러한 객관적인 데이터는 어떻게 정할 것인가?



처음에 고안했던 방법은 네이버의 랭킹 뉴스의 상위 5개 기사의 댓글들을 가져오는 것이었다. 그러나 네이버 랭킹 뉴스는 2020년까지만 접근이 가능했다. 우리가 원하는 데이터는 좀 더 먼 최소한 2010년 정도의 과거 데이터는 필요했기에 네이버의 뉴스 댓글 데이터는 ㅇ나쉬움이 있었다.   
  
그러던 중 네이트의 베스트 댓글 시스템을 발견했다. 해당 시스템은 정해진 주별, 또는 일별로 가장 추천을 많이 받은 댓글들을 보여주는 사이트란이었다. 해당 시스템을 통해 특정 분야의 기사들만 쏠림 현상이 발생할 수도 있고 이에 따라 객관성을 확보하기 위해 모든 영역의 댓글들을 다 가져와야 한다는 어려움이 있었는데 해당 시스템은 추천수라는 명확한 기준을 통해 다양한 기사의 댓글들을 가져오기에 댓귿들을 파악하기 위해 대표성을 대표성ㄴ을 가지기에도 분했다.

해당 사이트에서는 단순 ui로는 이번달까지의 DATA를 추출할 수 밖에 없었지만 URL을 조정함으로서 최대 2005년까지의 DATA를 추출할 수 있었다. 이는 과거의 DATA를 파악하기에 충분한 과거였으며 처음 제시했던 1번 조건을 만족하는 데이터였다.

이제 DATA를 파이썬 코드로 출력하기 위한 코드를 짜야한다. 코드를 짜기 전에 먼저   
  
텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명  
위와 같이 해당 웹페이지에서 F12버튼을 눌러 웹페이지의 베스트 댓글이 jreplylist의 usertext로 존재하고 있다는 것을 확인할 수 있었다.

정보들을 토대로 파이썬 파일을 작성하면 다음과 같다.

import requests

from bs4 import BeautifulSoup

import csv

from datetime import datetime, timedelta

def get\_weekly\_best\_comments(date):

    targeted\_url = f'https://news.nate.com/comment?type=best&mid=n1501&sect\_id=&p=week&ymd={date}'

    requested = requests.get(targeted\_url)

    soup = BeautifulSoup(requested.text, 'html.parser')

    iframe = soup.find('iframe', id='ifr\_reple')

    if iframe:

        iframe\_url = 'https:' + iframe['src']

        iframe\_response = requests.get(iframe\_url)

        iframe\_soup = BeautifulSoup(iframe\_response.text, 'html.parser')

        jreplyList = iframe\_soup.find('div', class\_='jreplyList jreplyLast')

        comments = []

        if jreplyList:

            jreply\_wraps = jreplyList.find\_all('div', class\_='jreplyWrap')

            for jreply\_wrap in jreply\_wraps:

                usertxt\_elements = jreply\_wrap.find\_all('dd', class\_='usertxt')

                for usertxt in usertxt\_elements:

                    comment = usertxt.text.strip()

                    comments.append({'date': date[:7], 'comment': comment})  # YYYY-MM 형식

        return comments

    else:

        print("Iframe not found")

        return []

def generate\_weekly\_comments\_csv(start\_year, start\_month, end\_year, end\_month):

    start\_date = datetime(start\_year, start\_month, 1)

    end\_date = datetime(end\_year, end\_month, 1)

    delta = timedelta(days=7)

    filename = f'weekly\_best\_comments\_{start\_year}\_{start\_month}\_to\_{end\_year}\_{end\_month}.csv'

    fieldnames = ['date', 'comment']

    with open(filename, mode='w', newline='', encoding='utf-8') as file:

        writer = csv.DictWriter(file, fieldnames=fieldnames)

        writer.writeheader()

        current\_date = start\_date

        while current\_date <= end\_date:

            date\_str = current\_date.strftime('%Y-%m-%d')

            comments = get\_weekly\_best\_comments(date\_str)

            for comment in comments:

                writer.writerow(comment)

            current\_date += delta

    print(f'Weekly best comments from {start\_year}-{start\_month} to {end\_year}-{end\_month} have been written to {filename}')

# 2023년 5월부터 2024년 5월까지의 주간 베스트 댓글을 CSV 파일로 생성

generate\_weekly\_comments\_csv(2013, 1, 2023, 12)

1. **베스트댓글 가져오기 상세 코드설명**

**get\_weekly\_best\_comments(date) 함수**

주어진 날짜(date)에 해당하는 주간 베스트 댓글을 수집하는 함수이다.

requests 라이브러리를 사용해 Nate 뉴스 댓글 페이지에 GET 요청을 보낸다.

BeautifulSoup을 이용해 HTML을 파싱하고, iframe 태그를 찾는다. iframe 태그가 발견되면 해당 src 속성값을 추출하여 다시 GET 요청을 보낸다.

응답받은 iframe 페이지를 BeautifulSoup으로 파싱하고, 댓글 리스트(jreplyList)를 찾는다.

jreplyList 내에서 댓글 텍스트(usertxt)를 추출하여 리스트로 반환한다. 반환되는 각 댓글에는 해당 날짜의 연월(YYYY-MM)이 포함된다.

**generate\_weekly\_comments\_csv(start\_year, start\_month, end\_year, end\_month) 함수:**

주어진 시작 연월(start\_year, start\_month)부터 종료 연월(end\_year, end\_month)까지의 기간 동안 주간 베스트 댓글을 CSV 파일로 저장하는 함수이다.

시작 날짜와 종료 날짜를 설정하고, 일주일 간격(delta)으로 루프를 돌면서 각 주의 날짜를 생성한다.

각 주의 날짜에 대해 get\_weekly\_best\_comments 함수를 호출하여 댓글을 수집한다.

수집된 댓글을 지정된 CSV 파일에 저장한다.

파일명은 주어진 기간을 포함하는 형태로 설정된다.

마지막으로, generate\_weekly\_comments\_csv(2013, 1, 2023, 12) 함수 호출을 통해 2013년 1월부터 2023년 12월까지의 주간 베스트 댓글을 CSV 파일로 생성하게 된다. 이 함수는 실행 후 "Weekly best comments from 2013-1 to 2023-12 have been written to weekly\_best\_comments\_2013\_1\_to\_2023\_12.csv"라는 메시지를 출력한다.

이 프로그램은 Nate 뉴스 사이트의 주간 베스트 댓글을 체계적으로 수집하고 저장하는 데 유용하다. 웹 스크래핑 기법을 사용하며, requests와 BeautifulSoup 라이브러리를 활용하여 웹 페이지를 파싱하고 필요한 데이터를 추출한다. CSV 파일로 저장함으로써 나중에 데이터를 쉽게 분석할 수 있도록 돕는다.

1. **얻은 데이터 set**

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

코드를 통해서 얻은 데이터는 위와 같다. 사이트를 통해 얻은 Data의 내용은 colum은 날짜를 나타내는 date 와 댓글의 내용 data가 있는 comment 열이다. 그러나 위의 데이터만으로는 연도별로 혐오 표현들이 늘었는지, 그리고 어떤 종류의 혐오 표현들이 있었는지는 전혀 알 수가 없다. 따라서 위 dateset을 가공해 추가적인 dataset 을 만들어야 했다.

1. **댓글 분석 dataset(using Korean unsmile Dataset)**

전의 과정에서 얻었던 데이터들을 토대로 새로운 dataset을 만들기 위해 스마일 게이트의 dataset 분석 모델을 이용했다.

텍스트, 스크린샷, 폰트, 이모티콘이(가) 표시된 사진

자동 생성된 설명

본 데이터셋은 특정 사회적 집단에 대한 혐오 표현에 대해 혐오의 대상이 속한 집단을 분류해주는 프로그램이다. 데이터셋은 혐오 카테고리를 여성/가족, 남성, 성소수자, 인종/국적, 연령, 지역, 종교, 악플/욕설, Clean 으로 구분한다. 여기서 clean 은 해당 표현이 혐오 표현이 아닌 일반 문장임을 의미한다. 이 프로그램은 문장에 대해 해당 카테고리 별로 혐오 수치를 알려준다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

수치가 0에 가까울수록 해당 카테고리와 관련이 없음을 뜻하고 1에 가까울수록 해당 카테고리와 관련이 깊다는 것을 의미한다.

위 프로그램을 이용해 이전에 구해놓은 댓글 dataset을 분류해 분석해주는 파이썬 프로그램을 만들었다. 이 프로그램은 이전 dataset의 comment들을 스마일게이트 dataset 프로그램을 이용해 분석해서 어떤 혐오 표현인지를 카테고리

위 프로그램은 특정 댓글 CSV 파일을 읽어와 댓글 내용에 대해 감정 분석을 수행한 후, 분석 결과를 새로운 CSV 파일로 저장하는 기능을 가진다. 프로그램의 주요 구성 요소와 그 작동 방식은 다음과 같다.

import csv

import torch

from transformers import TextClassificationPipeline, BertForSequenceClassification, AutoTokenizer

# Load the sentiment analysis model and tokenizer

model\_name = 'smilegate-ai/kor\_unsmile'

device = 0 if torch.cuda.is\_available() else -1

model = BertForSequenceClassification.from\_pretrained(model\_name)

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

pipe = TextClassificationPipeline(

    model=model,

    tokenizer=tokenizer,

    device=device,

    top\_k=None,  # Use top\_k=None to get all scores

    function\_to\_apply='sigmoid'

)

def truncate\_text(text, tokenizer, max\_length):

    # Tokenize and truncate the text to the max length

    tokens = tokenizer(text, truncation=True, max\_length=max\_length, padding=False, return\_tensors='pt')

    return tokenizer.decode(tokens['input\_ids'][0], skip\_special\_tokens=True)

def analyze\_comments(input\_csv, output\_csv, max\_length=300):

    # Read the input CSV file

    with open(input\_csv, mode='r', encoding='utf-8') as infile:

        reader = csv.DictReader(infile)

        comments = [row for row in reader]

    # Analyze the comments and prepare data for the output CSV file

    analyzed\_comments = []

    for comment in comments:

        truncated\_comment = truncate\_text(comment['comment'], tokenizer, max\_length)

        analysis = pipe(truncated\_comment)

        result = {'date': comment['date'], 'comment': comment['comment']}

        for score in analysis[0]:

            result[score['label']] = score['score']

        analyzed\_comments.append(result)

    # Define the fieldnames for the output CSV

    fieldnames = ['date', 'comment', '여성/가족', '남성', '성소수자', '인종/국적', '연령', '지역', '종교', '기타 혐오', '악플/욕설', 'clean']

    # Write the analyzed data to the output CSV file

    with open(output\_csv, mode='w', newline='', encoding='utf-8') as outfile:

        writer = csv.DictWriter(outfile, fieldnames=fieldnames)

        writer.writeheader()

        for analyzed\_comment in analyzed\_comments:

            writer.writerow(analyzed\_comment)

    print(f'Analysis results have been written to {output\_csv}')

# Example usage

input\_csv = 'weekly\_best\_comments\_2006\_1\_to\_2024\_1.csv'

output\_csv = '2006-2023\_analyzed\_comments2.csv'

analyze\_comments(input\_csv, output\_csv)

1. **베스트댓글 분석 코드 상세 설명**

**모델 및 토크나이저 로드:** transformers 라이브러리를 사용하여 Smilegate의 한국어 감정 분석 모델인 kor\_unsmile을 로드하고 BertForSequenceClassification 모델과 AutoTokenizer 토크나이저를 로드하여 텍스트 분류 파이프라인을 설정한다. 이때, top\_k=None으로 설정하여 모든 점수를 반환하도록 하고, function\_to\_apply='sigmoid'로 설정하여 시그모이드 함수를 적용한다.

**텍스트 길이 제한 함수:** truncate\_text(text, tokenizer, max\_length) 함수는 주어진 텍스트를 최대 길이로 토크나이즈하고, 토큰을 잘라낸 후 다시 디코딩하여 반환한다. 이는 모델의 입력 길이를 제한하기 위해 사용된다.

**댓글 분석 함수:** analyze\_comments(input\_csv, output\_csv, max\_length=300) 함수는 입력 CSV 파일을 읽고, 각 댓글에 대해 감정 분석을 수행한 후, 결과를 새로운 CSV 파일로 저장한다. 입력 CSV 파일을 읽어 댓글 리스트를 생성한다.

각 댓글에 대해 텍스트를 최대 길이로 자르고, 감정 분석 파이프라인을 통해 분석을 수행한다.

분석 결과를 포함한 딕셔너리를 생성하여 리스트에 추가한다.

출력 CSV 파일에 분석 결과를 저장한다. 이때, CSV 파일의 필드명은 댓글 날짜, 댓글 내용, 각 감정 분류 결과(여성/가족, 남성, 성소수자, 인종/국적, 연령, 지역, 종교, 기타 혐오, 악플/욕설, clean)로 구성된다.

위 프로그램을 통해 2006년부터 2023년까지의 모든 댓글들을 분석했다. 그러나 위 자료만으로 년도별로 변화와 어떤 혐오 표현들이 많이 쓰였는지를 알기는 쉽지 않았다. 따라서 연도별로 통계를 내주는 프로그램을 작성했다..

1. **연도별 혐오 표현 통계 구하기 프로그램**

import csv

from collections import defaultdict

def calculate\_statistics(input\_csv, output\_csv):

    # Read the input CSV file

    with open(input\_csv, mode='r', encoding='utf-8') as infile:

        reader = csv.DictReader(infile)

        comments = [row for row in reader]

    # Initialize a nested dictionary to count maxname occurrences per year

    stats = defaultdict(lambda: defaultdict(int))

    total\_counts = defaultdict(int)

    for comment in comments:

        year = comment['date'][:4]

        maxname = comment['maxname']

        stats[year][maxname] += 1

        total\_counts[year] += 1

    # Prepare data for output CSV

    output\_data = []

    categories = ['여성/가족', '남성', '성소수자', '인종/국적', '연령', '지역', '종교', '기타 혐오', '악플/욕설', 'clean']

    for year, counts in stats.items():

        year\_data = {'year': year}

        for category in categories:

            count = counts[category]

            total = total\_counts[year]

            percentage = (count / total) \* 100 if total > 0 else 0

            year\_data[f'{category}\_count'] = count

            year\_data[f'{category}\_percentage'] = round(percentage, 2)

        output\_data.append(year\_data)

    # Define the fieldnames for the output CSV

    fieldnames = ['year'] + [f'{category}\_count' for category in categories] + [f'{category}\_percentage' for category in categories]

    # Write the processed data to the output CSV file

    with open(output\_csv, mode='w', newline='', encoding='utf-8') as outfile:

        writer = csv.DictWriter(outfile, fieldnames=fieldnames)

        writer.writeheader()

        for data in output\_data:

            writer.writerow(data)

    print(f'Statistics have been written to {output\_csv}')

# Example usage

input\_csv = '2006-2023\_processed\_comments.csv'

output\_csv = '2006-2023\_yearly\_statistics.csv'

calculate\_statistics(input\_csv, output\_csv)

위 프로그램은 댓글 데이터 CSV 파일을 읽어 연도별로 각 카테고리의 댓글 수와 비율을 계산하여 새로운 CSV 파일로 저장하는 기능을 가진다. 프로그램의 주요 구성 요소와 그 작동 방식은 다음과 같다.

**입력 CSV 파일 읽기:** calculate\_statistics(input\_csv, output\_csv) 함수는 주어진 입력 CSV 파일을 읽어 댓글 데이터를 리스트로 저장한다.csv.DictReader를 사용하여 CSV 파일의 각 행을 딕셔너리 형태로 읽어들인다.

**통계 계산을 위한 초기화:** defaultdict를 사용하여 연도별 각 카테고리의 댓글 수를 저장할 중첩 딕셔너리를 초기화한다.total\_counts는 각 연도의 총 댓글 수를 저장하기 위한 딕셔너리이다.

**댓글 데이터 처리 및 통계 계산:** 각 댓글에 대해 연도와 카테고리(maxname)를 추출하여 해당 연도의 해당 카테고리 댓글 수를 증가시킨다.또한, 해당 연도의 총 댓글 수도 증가시킨다.

**출력 데이터를 준비:** 각 연도에 대해 카테고리별 댓글 수와 비율을 계산하여 output\_data 리스트에 저장한다.카테고리 목록은 categories 리스트로 정의된다.

각 연도의 데이터는 딕셔너리 형태로 저장되며, 카테고리별 댓글 수와 비율을 계산하여 저장한다.

**출력 CSV 파일 작성:** 출력 CSV 파일의 필드명(fieldnames)을 정의한다. 필드명은 연도(year), 카테고리별 댓글 수와 비율을 포함한다.

csv.DictWriter를 사용하여 output\_data 리스트의 데이터를 출력 CSV 파일에 작성한다.

**사용 예시:** calculate\_statistics 함수를 호출하여, 2006-2023\_processed\_comments.csv 파일의 댓글 데이터를 분석하고, 결과를 2006-2023\_yearly\_statistics.csv 파일에 저장한다.

위 프로그램은 댓글 데이터의 감정 분석 결과를 기반으로 연도별로 각 감정 카테고리의 댓글 수와 비율을 계산하여 저장하는 데 유용하다. 이를 통해 특정 연도에 특정 감정 카테고리의 댓글이 얼마나 많이 발생했는지, 그 비율은 어떻게 되는지를 쉽게 파악할 수 있다. 이 dataset은 최종 dataset이다.

1. **베스트댓글 가져오기 상세 코드설명**

이제 이제까지 얻은 dataset을 토대로 clear 하게 data visualization을 할 방법을 고안해야한다. 내가 clear communication에 요구하는 요구사항은 두가지였다. 첫째는 연도별로 혐오 표현들이 과거에 비해 얼마나 늘어났는지를 알 수 있어야 한다는 것이고 두 번째는 각 혐오 표현 카테고리 별로 어느 정도 비율을 차지하는지 알 수 있어야 된다는 점이었다.

1. **원형 차트**

맨 처음 고안했던 방법은 원형 차트였다.

텍스트, 스크린샷, 원, 도표이(가) 표시된 사진

자동 생성된 설명

원형 차트의 경우 각 종류별 혐오 수치를 단순에 알 수 있다는 장점이 있었다. 그러나 clear communication을 위해 설계했던 두 번째 조건은 만족했으나 첫 번째 조건은 만족하지 못했다. 다시 말해 원형 차트로는 혐오 표현의 카테고리별 비율은 알 수 있지만 연도별 추이를 알 수 없었기에 원형 차트는 clear communication에 적절치 않았다.

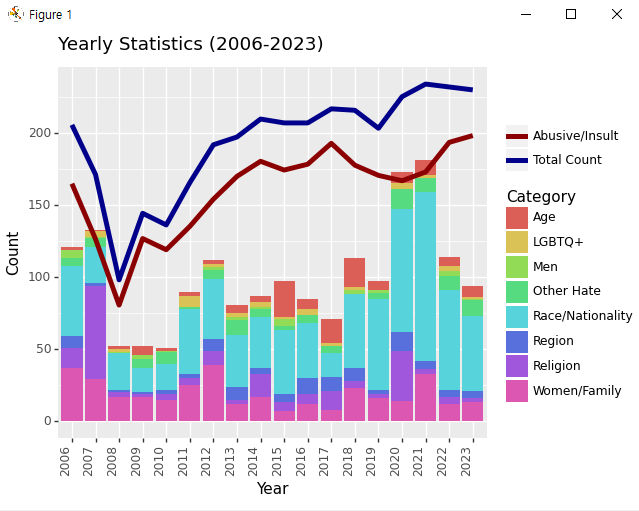
1. **선 그래프**

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

두번째로 고려한 차트의 형태는 선 그래프 형태다. 선 그래프를 통해 연도별 추이는 알 수 있었고 각 카테고리별 얼마나 혐오 표현들이 있었는지도 알 수 있었다. 그러나 악플/혐오 카테고리와 clean 카테고리들에 비해 나머지 카테고리들의 비율이 너무 적기 때문에 그 수치들이 한 눈에 들어오지 않았다. 또한 위 그래프에 추가적으로 연도별 추이도 나타낼 수 있지만 문제는 한 눈에 봤을 때 연도별 추이가 특별하게 보이지 않고 여타 다른 카테고리들의 선 그래프와 동일한 종류의 선 그래프로 보인다는 것이었다. 따라서 line graph는 clear communication에 적절하지 않았다.

1. **혼합 그래프**



세 번째로 고려한 차트의 형태는 혼합 그래프(막대 그래프 + 선 그래프)의 형태다. 이 때 그래프가 너무 복잡해질 수 있기 때문에 막대 그래프는 누적 막대 그래프를 사용했고 전체 혐오 표현 및 악플의 수치는 선 그래프로 표현했다. 처음에는 위 그래프와 같은 형태를 최종 데이터로 만들려 했으나 곰곰이 생각해보니 문제점이 한 가지 있었다. 누적 막대 그래프를 만든 것은 그 누적 수치도 식별 가능한 데이터가 되기 때문이었는데 현재의 누적 데이터는 악플/ 욕설 수치를 제외한 데이터였기 때문에 누적 데이터는 의미가 없는 값이었다.

또한 만일 악플 수치도 누적 막대 그래프에 포함시킨다면 누적 그래프만으로도 연도별 추이가 식별 가능해져 굳이 선그래프로 전체 수치를 나타낼 필요가 없어진다.

1. **막대 그래프(누적 막대 그래프)**

결국 최종적으로 clear 커뮤니케이션을 위해 내가 고려한 형태는 위와 같은 막대 그래프

의 형태다. 그런데 막대 그래프도 몇 가지 종류가 있다. 위 그림처럼 각 카테고리별로 개별적인 막대 그래프로 나타낸 형태도 있었고 아예 누적 막대 그래프로 나타낸 형태도 있었다. 각 카테고리별 막대 그래프로 나타낸다면 2005년부터 2023까지를 모두 나타내다보면 가로 길이가 무척 길어져 가독성이 떨어질 것이다. 반대로 누적 막대 그래프로 나타내면 연도별 추이를 쉽게 인식할 수 있고 카테고리별 차지하는 비율도 한 눈에 들어온다.   
 다음 그림은 최종 막대 그래프의 그림이다.

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

1. **막대 그래프 생성 코드 및 설명**
2. *import* pandas *as* pd
3. *import* matplotlib.pyplot *as* plt
4. *from* plotnine *import* ggplot, aes, geom\_bar, labs, theme, element\_text
5. *# Load the data*
6. data = pd.read\_csv('2006-2023\_yearly\_statistics.csv')
7. *# Combine categories into '기타 혐오\_count' (Other Hate)*
8. data['기타 혐오\_count'] += data['남성\_count'] + data['성소수자\_count'] + data['연령\_count'] + data['지역\_count'] + data['종교\_count'] + data['기타 혐오\_count']
9. *# Keep only the necessary columns*
10. data = data[['year', '여성/가족\_count', '인종/국적\_count', '악플/욕설\_count', '기타 혐오\_count']]
11. *# Melt the data to long format*
12. data\_melted = data.melt(*id\_vars*=['year'],
13. *value\_vars*=['여성/가족\_count', '인종/국적\_count', '악플/욕설\_count', '기타 혐오\_count'],
14. *var\_name*='Category',
15. *value\_name*='Count')
16. *# Define category labels in English for clarity in the graph*
17. category\_labels = {
18. '여성/가족\_count': 'Women/Family',
19. '인종/국적\_count': 'Race/Nationality',
20. '악플/욕설\_count': 'Abusive/Insult',
21. '기타 혐오\_count': 'Other Hate'
22. }
23. data\_melted['Category'] = data\_melted['Category'].map(category\_labels)
24. *# Plot the stacked bar chart using ggplot*
25. plot = (ggplot(data\_melted, aes(*x*='factor(year)', *y*='Count', *fill*='Category')) +
26. geom\_bar(*stat*='identity', *position*='stack') +
27. labs(*title*='Yearly Statistics (2006-2023)', *x*='Year', *y*='Count') +
28. theme(*axis\_text\_x*=element\_text(*rotation*=90, *hjust*=1)))
29. *# Display the plot*
30. print(plot);

**데이터 로드 및 전처리**

**데이터 로드:** pandas를 사용하여 CSV 파일을 읽어와 데이터프레임으로 저장한다.

카테고리 결합: 기타 혐오 카테고리에 다른 관련 카테고리의 댓글 수를 더한다.

필요한 열 추출: 분석에 필요한 열만 남긴다.

**데이터 변환:** pandas.melt 함수를 사용하여 데이터를 긴 형식으로 변환한다.

카테고리 이름 매핑: 카테고리 이름을 영어로 변경하여 그래프에서 명확히 표시되도록 한다.

**각 구성 요소의 설명**

**ggplot(data\_melted, aes(x='factor(year)', y='Count', fill='Category')):**

ggplot 객체를 생성하며, data\_melted 데이터프레임을 사용한다.

aes 함수는 그래프의 미적 매핑을 설정한다. x축은 연도(year), y축은 댓글 수(Count), 색상(fill)은 카테고리(Category)로 매핑된다.

**geom\_bar(stat='identity', position='stack'):**

geom\_bar는 막대 그래프를 그리는 함수이다.

stat='identity'는 데이터의 실제 값을 사용하여 막대를 그리도록 설정한다.

position='stack'은 각 카테고리의 값을 스택 형태로 쌓아 올린다.

labs(title='Yearly Statistics (2006-2023)', x='Year', y='Count'):

labs 함수는 그래프의 제목과 축 레이블을 설정한다.

title은 그래프 제목, x는 x축 레이블, y는 y축 레이블을 지정한다.

theme(axis\_text\_x=element\_text(rotation=90, hjust=1)):

theme 함수는 그래프의 테마를 설정한다.

axis\_text\_x=element\_text(rotation=90, hjust=1)는 x축의 텍스트를 90도 회전하여 세로로 표시하고, 텍스트의 가로 정렬을 조정한다.

**그래프 출력**

마지막으로 print(plot)을 통해 생성된 ggplot 객체를 출력하여 그래프를 화면에 표시한다.