인터넷 기사의 댓글 속 혐오 양상에 대하여

20205178 박 현

인공지능 알고리즘을 이용한 사회과학연구 – 기말 프로젝트

주제 선정 이유

최근 온 · 오프라인 상에서의 혐오가 사회적 문제로 떠오르고 있다. 이에 수 년간의 인터넷 기사 댓글을 수집하여 Smilegate AI에서 공개한 Korean UnSmile Dataset에서 제공하는 인공지능 모델을 이용하여 각 댓글에 해당하는 혐오 분류를 찾아 이들에 대한 분포 및 양상을 분석하고자 한다.

사용한 패키지 – GNU S

future: Unified Parallel and Distributed Processing in R for Everyone

future.apply: Apply Function to Elements in Parallel using Futures

rvest: Easily Harvest (Scrape) Web Pages

tidyverse: Easily Install and Load the 'Tidyverse'

RmecabKo: Rcpp Wrapper for Eunjeon Project

plotly: Create Interactive Web Graphics via 'plotly.js'

wordcloud2: Create Word Cloud by 'htmlwidget'

sqldf: Manipulate R Data Frames Using SQL

httr: Tools for Working with URLs and HTTP

사용한 패키지 – Python

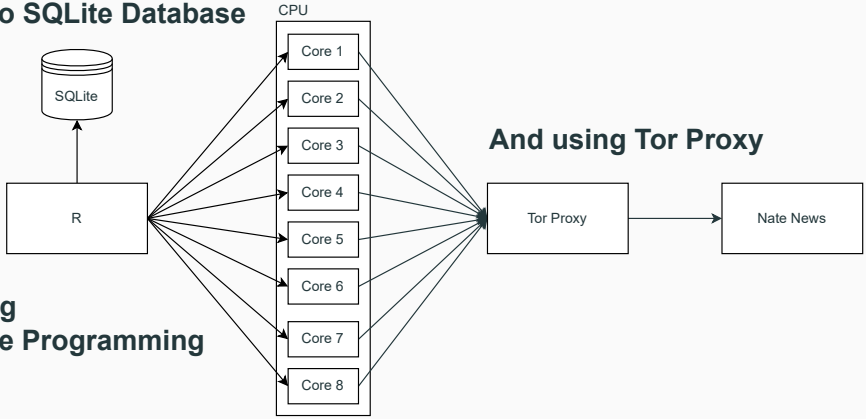
Hugging Face: Datasets, Transformers.

Pandas: Python Data Analysis Library

flask: A simple framework for building complex web applications.

자료 수집

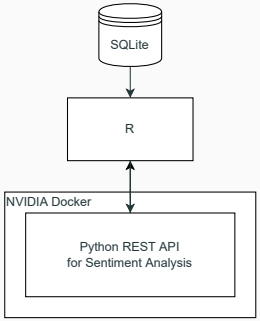
Finally, store to SQLite Database



And using Tor Proxy

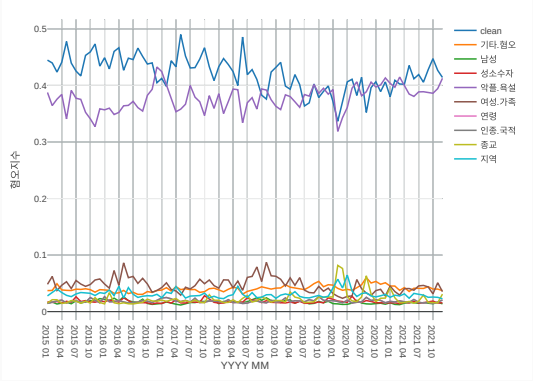
Web Scrapping
with Multi Core Programming

감정 분석



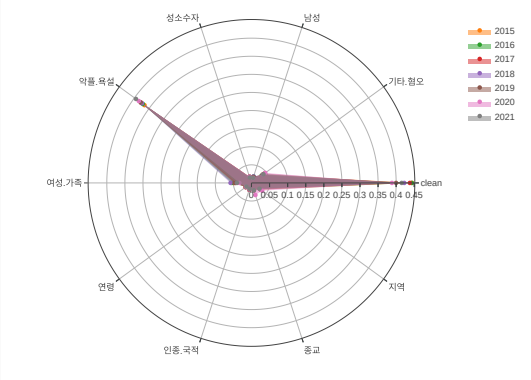
1. SQLite로부터 수집한 댓글을 가져오기.
2. 댓글마다 R의 `httr::POST(url)`을 이용하여 도커 내 REST API 서버에 POST 요청을 보냄.
3. Docker 안에서 BERT 모델을 이용해 감정 분석을 GPU로 가속하며 진행.
4. R에서 감정 분석 결과들을 `do.call(rbind, dfs)`를 이용해 병합.

시각화 – 꺾은선 그래프



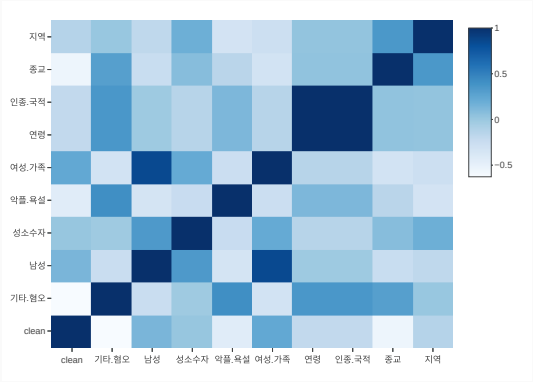
- clean과 악플의 비율은 비슷하다.
- 남성과 여성의 양상이 유사하다.
  → 상관분석을 해보자.

시각화 – 레이더 차트



- 여러 분류에 대한 분포를 쉽게 볼 수 있는 레이더 차트를 이용하여 시각화해보았다.
- clean과 악플이 다른 분류에 비해 매우 높아 보기 어려워 시연 파트에서 인터랙티브한 그래프를 보여주며 살펴보고자 한다.

시각화 – 히트 맵



- `cor(df)`를 이용하여 상관계수를 구한 결과를 히트 맵을 이용하여 시각화해보았다.
- 위에서 꺽은선 그래프를 통해 확인한 바와 같이 남성과 여성의 상관관계가 매우 높다는 사실을 알 수 있다.
- 그 외에도 지역과 종교 간의 상관성과 남성과 성소수자 간의 상관성 또한 높음을 히트 맵을 통해 확인할 수 있었다.

결과

감정 분석

악플과 그렇지 않은 댓글의 비율은 비슷했다.

상관 분석

여성에 대한 혐오와 남성에 대한 혐오가 매우 큰 상관성을 갖는 것으로 나왔으며, 종교와 지역 간의 상관성과 남성과 성소수자 간의 상관성 또한 다른 요소에 비해 높은 것으로 나타났다.

코드 – 웹 스크래핑

```
get_news_list ← function(date, week) {
    parameter ← sprintf("sc=&p=%s&date=%s", ifelse(week, "week", "day"), date)
    url ← paste("https://news.nate.com/rank/cmt", parameter, sep = "?")
    doc ← rvest::read_html(url)
    tail ← doc ▷ rvest::html_elements(".mduSubject > *")
    head ← doc ▷ rvest::html_elements(".mduSubjectList")
    extract_tail_element ← function(e) {
        a ← rvest::html_element(e, "a")
        ems ← rvest::html_elements(e, "em")
        list(
            rank = as.integer(rvest::html_text(ems[1])),
            comment =
                rvest::html_text(ems[2]) ▷
                    stringr::str_remove_all(",") ▷
                    as.integer(),
            title = rvest::html_text(a),
            url = paste0("https:", rvest::html_attr(a, "href"))
        )
    }
    extract_head_element ← function(e) {
        ems ← rvest::html_elements(e, "em")
        a ← rvest::html_element(e, "a")
        list(
            rank = as.integer(rvest::html_text(ems[1])),
            comment =
                rvest::html_text(ems[2]) ▷
                    stringr::str_remove_all(",") ▷
                    as.integer(),
            title =
                rvest::html_element(a, "strong") ▷
                    rvest::html_text(),
            url = paste0("https:", rvest::html_attr(a, "href"))
        )
    }
    tail ← Map(extract_tail_element, tail)
    head ← Map(extract_head_element, head)
    do.call(rbind.data.frame, append(head, tail))
}
```

```
get_news_comments ← function(url, page) {
    artc_sq ← urltools::url_parse(url)$path ▷ substring(6)
    mid ← urltools::param_get(url, "mid")[[1]]
    comment_url ← paste0(
        "https://comm.news.nate.com/Comment/ArticleComment/List?",
        "artc_sq=%s&",
        "order=&cmtr_fl=0&prebest=0&clean_idx=&user_nm=&fold=&",
        "mid=%s&domain=&argList=0&best=0&return_sq=&connectAuth=N&page=%d"
    ) ▷
        sprintf(artc_sq, mid, page)
    doc ← rvest::read_html(comment_url)
    cmt_items ← doc ▷ rvest::html_elements(".cmt_item")
    do.call(rbind.data.frame, Map(function(item) {
        list(
            name = item ▷
                rvest::html_element(".nameui") ▷
                rvest::html_text() ▷
                stringr::str_trim(),
            date = item ▷
                rvest::html_element(".date") ▷
                rvest::html_text() ▷
                substring(3) ▷
                paste(substring(artc_sq, 1, 4)),
            text = item ▷
                rvest::html_element(".usertxt") ▷
                rvest::html_text() ▷
                stringr::str_trim()
        )
    }, cmt_items))
}
```

코드 – 워드 클라우드

```
dbGetQuery(conn, "
SELECT
    DISTINCT (name || date || comments.text) as uniquecol,
    date,
    ct.*
FROM comments
INNER JOIN classified_texts ct
ON comments.text = ct.text") ▷
    mutate(ym = paste(
        substring(date, 13, 16),
        substring(date, 1, 2))) ▷
    mutate(year = as.integer(substring(date, 13, 16))) ▷
    select(!uniquecol) → joined_df
data.frame(word = future_Map(
    nouns,
    sqldf("
SELECT text FROM joined_df
WHERE ym LIKE'2015 __'")$text) ▷
    unlist() ▷
    as.vector()) → words_df
sqldf("
SELECT
    word,
    COUNT(1) as count
FROM words_df
GROUP By word
HAVING length(word) > 2
  AND count > 50
") ▷ wordcloud2(rotateRatio = 0)
```

SQL를 이용하여 데이터를 가공해 데이터 프레임으로 가져와 tidyverse에 있는 여러 함수를 이용해 필요에 맞게 다시 한 번 가공한 후 `RmecabKo::nouns` 함수를 이용하여 명사를 추출해 워드 클라우드를 그리는 코드.

코드 – GitHub

프로젝트을 진행하며 작성한 모든 코드는 아래 저장소에 전부 업로드 되어 있습니다.

https://github.com/HyunP-dev/Social-Science-Study-using-AI-Final-Project/

– Demonstration –