텍스트
데이터 분석

사례연구 5

A3조 남원식, 문태웅, 이순규

목 차

서론

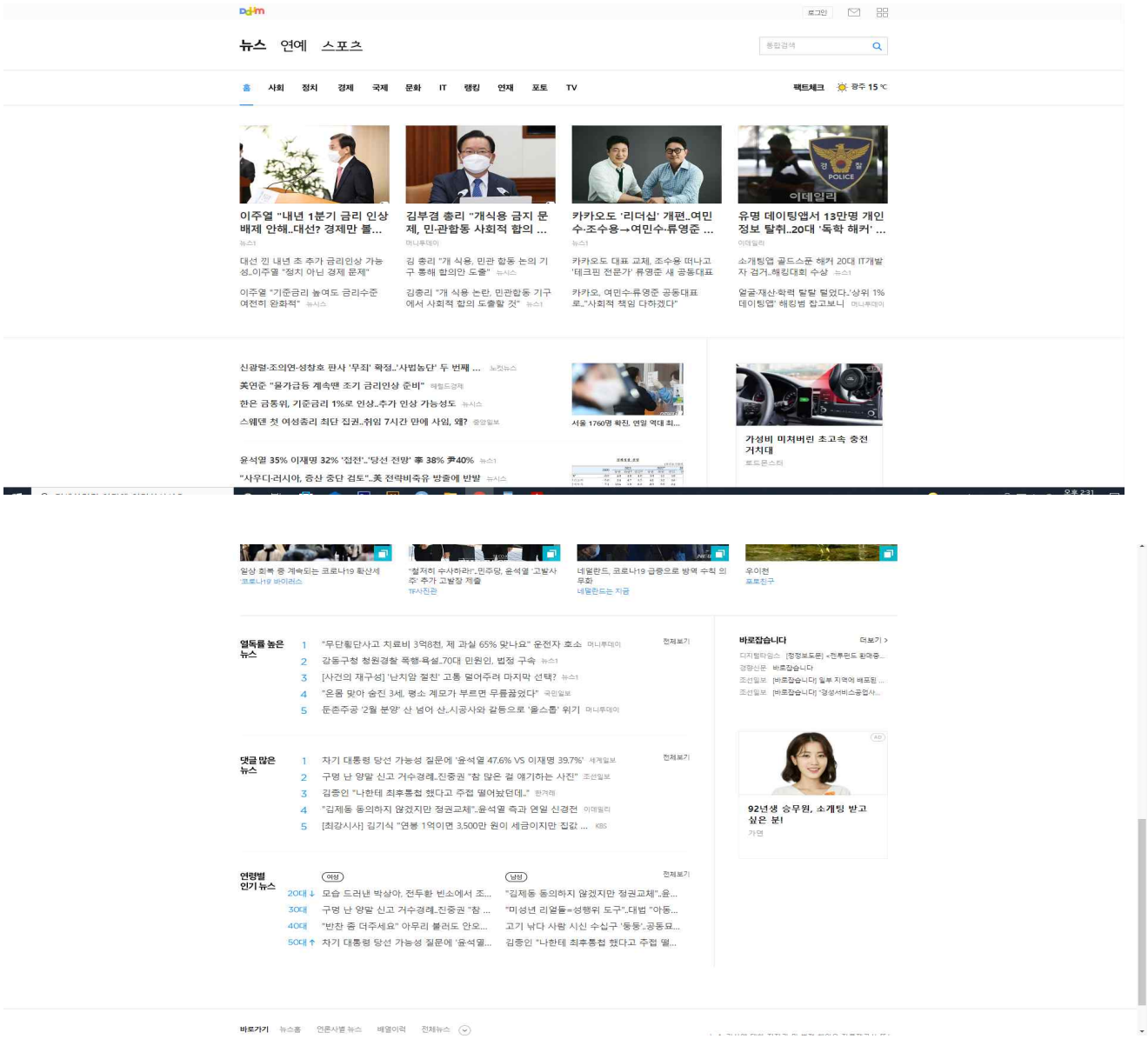
1. 데이터

1-1. 링컨 게티스버그 연설문

링컨 게티스버그 연설문 (1863.11.19)

87년 전 우리의 선조들은 이 대륙에 자유의 정신으로 잉태되고 만인이 평등하게 창조되었다는 신념이 바쳐진 새로운 나라를 세웠습니다.지금 우리는 바로 그 나라가, 아니 이러한 정신과 신념으로 잉태되고 헌신하는 어느 나라이든지,과연 오래도록 굳건할 수 있는가 하는 시험대인 거대한 내전에 휩싸여 있습니다. (중략)-- 국민의,[15]국민에 의한, 국민을 위한 통치가 지상에서 소멸하지 않아야 한다는 그 위대한 사명에 우리 스스로를 바쳐야 합니다.

1-2.. 다음 실시간 뉴스 토픽 데이터



본론

1. 링컨의 연설문 분석

1-1. 토픽분석 데이터 전처리

데이터셋을 불러와 readline()를 이용하여 줄별로 분리

```
lincoln <- file("lincoln.txt", encoding = "UTF-8")
lincoln_l <- readLines(lincoln)
```

corpus()를 이용하여 말뭉치를 생성

```
myCorpus <- Corpus(VectorSource(lincoln_word))
```

문장부호 , 불용어들을 제거한다. (tm패키지 이용)

```
lincoln_pre <-  tm_map(myCorpus,removePunctuation) #문장부호 제거
lincoln_pre <-  tm_map(lincoln_pre,removeNumbers) # 수치제거
```

전처리된 단어들을 TermDoucumentMatrix()를 이용하여 2~8음절 단어로 추려내기

```
mylincoln_pre <- TermDocumentMatrix(lincoln_pre,control = list(wordLengths
= c(4,16)))
```

단어 구름을 생성하기 위한 빈도수를 구한다.

```
lincoln_word <- sort(rowSums(mylincoln_pre_df),decreasing = TRUE)
#내림차순정렬
```

1-2. 토픽분석

데이터 전처리를 토대로 빈도수를 이용해 토픽분석을 해보았다. 연설의 내용이기 때문에 빈도수를 이용하는 것은 한계점이 있었다. 첫째로 가장 많이 나온 단어는 우리이다. 왜 우리라는 말이 많이 언급이 되었는지는 그 당시의 배경과 연결을 지어보니 전쟁이 막 끝난 상황이었고 하나로 뭉쳐야 하는 힘든 상황이 었던 것으로 판단이 된다. 두번째 빈도수가 높았던 단어는 나라인데 위와 같은 이유서 국민들을 하나로 뭉치기 위해 '나라' 라는 단어를 사용한 것으로 판단이 된다. 세번째로 많은 빈도수를 차지한 단어는 '자리','헌신','국민' 등이 있다. '헌신'이라는 단어는 전쟁중 사망한 용사들을 위해 존경을 표하는 단어로 표현되었다. 하지만 "국민을 위한","국민에 의한","국민의"의 뜻이 담겨있는 주요문장은 빈도수가 3에 불과하여 단순 단어만으로는 토픽을 분석하고 중요한 문장을 찾는 데에는 한계점이 있다는 것으로 사려가된다.

1-3. 토픽분석 시각화


우리
나라
자리
헌신
마지막
국민
정신
스스로
축성
거대
신념
임태
헌정
자유
봉헌
싸움터
위대
통치가
고귀
대의
신명
탄생
수행
안식처
자신
전진
결단
신성
였기에
생존
신성화
티스버그
굳건
용맹
미완
오늘
고결
시험대
선조
대륙
마땅
과업
연설문
영령
주목
아래
러한
용사
링컨
여기
따위
창조
일부
세상
죽음
평등
전쟁
지상
전사자
소멸
보잘
내전
의미
무위
여기모여
기억
순국선열

2. 링컨의 연설문 데이터에 관한 연관어 분석

2-1. 연관어 분석 데이터 전처리

데이터셋을 불러와 readline()를 이용하여 줄별로 분리 및 줄 단위 단어 추출

```
Lincoln <- file('Lincoln.txt', encoding = "UTF-8")
Lincoln_data <- readLines(Lincoln)
lword <- Map(extractNoun, Lincoln_data)
length(lword)
```

중복 단어를 제거하기 unique()이용

```
lword <- unique(lword)
length(lword)
```

2~4글자의 한글만 추출하기

```
filter1 <- function(x) {
  nchar(x) <= 4 && nchar(x) >= 2 && is.hangul(x)
}
filter2 <- function(x) { Filter(filter1, x) }
```

트랜잭션 생성하기

```
wordtran <- as(lword, "transactions")
wordtran

tranrules <- apriori(wordtran,
                     parameter = list(supp = 0.15, conf = 0.99))
tranrules
```

연관어 분석 실행

```
detach(package:tm, unload = TRUE)
inspect(tranrules)
```

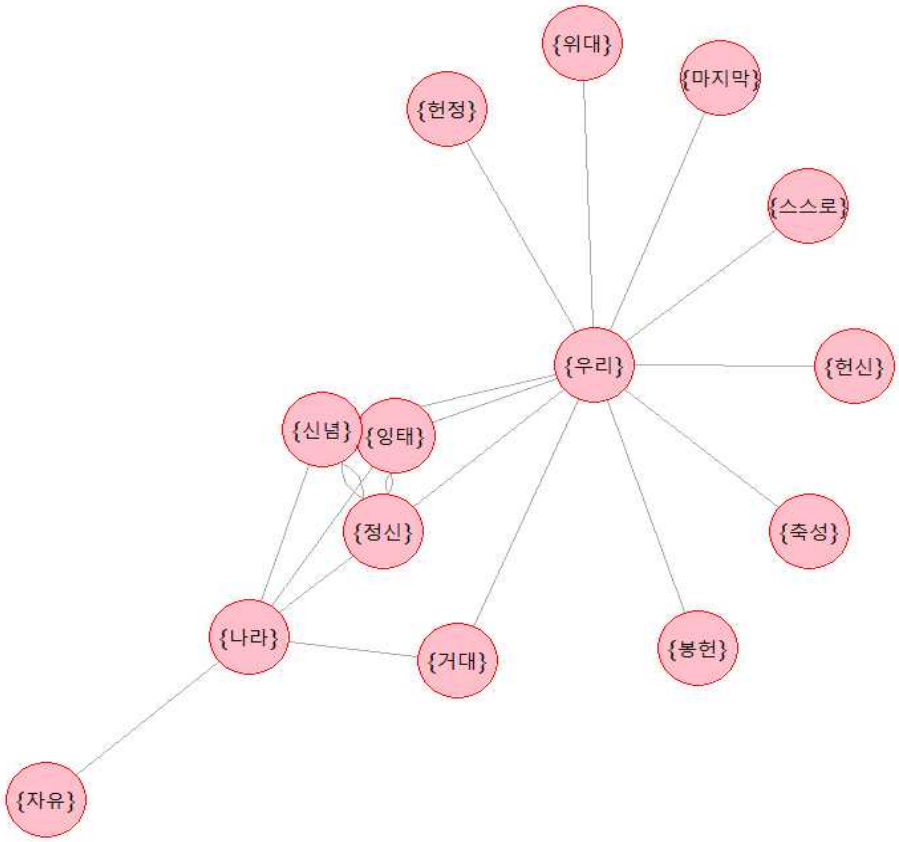
2-2. 연관어 분석

트랜잭션을 생성 할 때 지지도 비율을 0.16으로 설정, 신뢰도를 0.99로 설정을 하였다.

지지도 비율을 0.16으로 한 이유는 트랜잭션을 생성 후 0.17이상으로 연관 규칙을 생성 한 결과 연관된 단어 비율이 없다라고 도출이 되며, 0.16이하로 설정을 해야 연관된 단어 비율이 존재 한다고 도출이 되었으며, 신뢰도 비율을 0.99로 한 이유는 지지도 비율을 0.16으로 설정한 결과 모든 도출된 단어들의 신뢰도가 1로 도출 되었기 때문이다.

위의 연관어 분석을 토대로 한 시각화 결과는 아래와 같다.

2-3. 연관어 분석 시각화



3. 실시간 뉴스를 토픽 분석 후 시각화 및 주요 이슈에 대한 설명

3-1. 토픽 분석 데이터 전처리

사용할 URL 주소 가져오기

```
url <- "http://news.daum.net/"
web <- GET(url)
html <- htmlTreeParse(web,useInternalNodes = T,trim =T,encoding = "UTF-8")
rootNode <- xmlRoot(html)
news <- xpathApply(rootNode,"//a[@class = 'link_txt']",xmlValue)
```

Gsub()를 이용하여 불문들 제거하기

```
news_pre <- gsub("[₩r₩n₩t]", ' ', news) #줄바꿈
news_pre <- gsub('[[:punct:]]', ' ', news_pre) #특수문자
news_pre <- gsub('[[:cntrl:]]', ' ', news_pre)
news_pre <- gsub('₩₩d+', ' ', news_pre) #숫자
news_pre <- gsub('[a-z]+', ' ', news_pre)#소문자
news_pre <- gsub('[A-Z]+', ' ', news_pre)#대문자
news_pre <- gsub('₩₩s+', ' ', news_pre) #빈칸
```

Corpus()를 이용하여 말뭉치 생성하기

```
newsCorpus <- Corpus(VectorSource(news_word))
```

TermDocumentMatirx()를 이용하여 2~8음절의 단어 분리하기

```
news_tdm <- TermDocumentMatrix(newsCorpus, control = list(wordLengths = c(4, 16)))
```

Matrix형태의 데이터를 DataFrame형으로 변환하기

```
tdm.df <- as.data.frame(as.matrix(news_tdm))
```

토픽분석 및 시각화를 위한 빈도수 구하기

```
wordResult <- sort(rowSums(tdm.df), decreasing = TRUE)
```

3-2. 토픽 분석

토픽별 빈도수	
금리	6
김종인	5
윤석열	4
주접	3
최후통첩	3
기준	3

3-3. 토픽 분석 시각화

3-4. 주요 이슈에 대한 설명

토픽을 빈도수 데이터로 만든 결과 "금리","김종인","윤석열","주접","최후통첩","기준" 순으로 많은 기사가 있는 것을 알 수 있다. 금리의 경우 0%의 금리 시대가 종료된 뉴스 기사들로 인한 1순위 빈도로 나왔으며, 김종인 ~ 기준의 경우 윤석열 대선후보와 김종인 비전대위원장의 주고 받은 서로에 대한 의견 이슈 충돌로 인한 빈도수 결과를 볼 수 있었다.

뉴스 기사에 대한 토픽분석을 빈도수로 진행 한 결과 1번의 링컨 연설문의 경우에는 한사람이 한 대화의 형식이기 때문에 빈도수를 이용하여 중요도를 분석 하는데에 무리가 있었지만, 뉴스 기사의 경우에는 많이 나온 단어일수록 그 시점에 사람들이 관심 있어 하는 이슈가 무엇 인지 알 수가 있었다.

결론

텍스트 데이터 분석 기법

링컨의 연설문 데이터와 다음 실시간 뉴스의 데이터를 가지고 텍스트 데이터 분석을 한 결과 연설문의 쓰인 내용을 토대로 링컨 대통령이 무엇에 대한 자신의 의견을 강조 하고 싶은지, 단어를 분석하여 볼 수 있었으며, 강조하는 단어가 나왔을 때 따라 나오는 단어도 무엇인지 파악 할 수 있었다. 또한 실시간으로 변화하는 뉴스 데이터 경우 특정한 시간을 맞추어 분석을 시도하면, 그 시간 때의 국민들이 사회의 어떠한 이슈에 관해 보거나, 들을 수 있는 데이터를 볼 수 있게하는 결과를 보았다.

사용한 R코드

```
# 패키지 설치

install.packages("KoNLP") # 형태소 분석 패키지

install.packages("tm") # 텍스트 분석 패키지

install.packages("wordcloud") # 워드클라우드 시각화 패키지

install.packages("Sejong") # 한글 사전 패키지

install.packages("arules") # 연관성 분석 패키지

install.packages("wordcloud2") # 워드클라우드2 시각화 패키지

install.packages("igraph") # 변수들 사이의 네트워크 연결망을 그려주는 패키지

install.packages("httr") # http에 요청하고 응답에 필요한 패키지

install.packages("XML") # XML 문서처리 패키지


# 패키지 실행

library(KoNLP)

library(tm)

library(wordcloud)

library(Sejong)

library(arules)

library(wordcloud2)

library(igraph)

library(httr)

library(XML)
```

```
# 텍스트 자료 가져오기

rm(list = ls())

setwd("c:/Rwork/data") # 경로 설정

lincoln <- file("lincoln.txt", encoding = "UTF-8") # 텍스트 파일 UTF-8인코딩 및 변수에 담기

lincoln_l <- readLines(lincoln) # 불러온 데이터를 라인화 하여 변수에 저장

lincoln_l


# 단어 추출 사용자 정의함수로 만들기

exNouns <- function(x){paste(extractNoun(as.character(x)),

                             collapse = " ")}


#### (1)제공된 데이터를 이용하여 토픽 분석을 실시하여 단어구름으로 시각화 하고 단어 출현 빈도수를 기반하여 어떤 단어 들이 주요 단어인지 설명하시오.####

#install.packages("KoNLP") # 형태소 분석 패키지

#install.packages("tm") # 텍스트 분석 패키지

#install.packages("wordcloud") # 워드클라우드 시각화 패키지

#install.packages("Sejong") # 한글 사전 패키지

#install.packages("wordcloud2") # 워드클라우드2 시각화 패키지

# exNouns 함수를 이용하여 단어 추출

lincoln_word <- sapply(lincoln_l, exNouns)

lincoln_word


# 전처리

# 말뭉치 생성하기

myCorpus <- Corpus(VectorSource(lincoln_word))
```

```
# 문장부호 제거

lincoln_pre <-  tm_map(myCorpus,removePunctuation) # tm_map() 공백 제거 및 불필요한 불용어 제거에 사용하는 함수


# 수치제거

lincoln_pre <-  tm_map(lincoln_pre,removeNumbers)


# 전처리된 단어들 2~8음절인 단어만 추려내기

mylincoln_pre <- TermDocumentMatrix(lincoln_pre,control = list(wordLengths = c(4,16)))

# TermDocumentMatrix() : 여러 문서들의 집단에서 단어를 추출한 뒤,

# 행에 출현했던 단어 리스트를 나열하고, 열에 각 문서들을 나열할때 사용된다.


# matrix형을 dataframe형으로 변환

mylincoln_pre_df <- as.data.frame(as.matrix(mylincoln_pre))

dim(mylincoln_pre_df)


# 단어구름으로 시각화 하기위해 빈도수 구하기

# 워드클라우드1

lincoln_word <- sort(rowSums(mylincoln_pre_df),decreasing = TRUE) # 내림차순으로 순서 정렬

wordcount <- table(lincoln_word) # 빈도수 wordcount 에저장

myName <- names(lincoln_word) # 변수명 myName에 저장

word.df <- data.frame(word = myName, freq = lincoln_word) # 빈도수와 변수명을 데이터 프레임 word.df로 생성

pal <- brewer.pal(10, "Paired") # 단어의 색깔 지정

wordcloud(word.df$word, word.df$freq, scale = c(3,0.5),
```

```
          min.freq = 2, random.order = F,
          rot.per = .1, colors = pal)
# min.freq : 최소 빈도
# random.order = F : 최빈단어가 중간에 나오도록 설정(F)
# rot.per : 단어 회전도
# scale : 글자크기
# colors : 색상지정

# 워드 클라우드2
#install.packages("wordcloud2")
#library(wordcloud2)
wc <- wordcloud2(data = word.df,size = 1.5,color = "random-light",backgroundColor = "black",rotateRatio = 0.75)
# size : 글자 크기
# color : 글자색
# backgroundColor : 배경색
# rotateRatio : 글자 회전도
```

(2)제공된 데이터를 이용하여 연관어 분석을 실시하여 연관어를 시각화 하고 시각화 결과에 대해 설명하시오

```
# 패키지 설치

#install.packages("backports") # 오류발생시 설치하는 패키지

#library(backports)

#install.packages("arules") # 연관성 분석 패키지

#install.packages("tm") # 텍스트 분석 패키지

#install.packages("wordcloud2") # 워드클라우드2 시각화 패키지


# 줄별로 단어 추출

lword <- Map(extractNoun, Lincoln_data) # Map() : 리스트 형태로 반환해주는 함수

length(lword)

# 중복단어 제거

lword <- unique(lword)


# 단어 필터링 함수 정의(2 ~ 4 글자의 한글만 추출하는 필터)

filter1 <- function(x) {

  nchar(x) <= 4 && nchar(x) >= 2 && is.hangul(x)

}

filter2 <- function(x) { Filter(filter1, x) }


# 줄 단위로 추출된 단어 전처리

lword <- sapply(lword, filter2)

lword


# 트랜잭션 생성하기
```

```
wordtran <- as(lword, "transactions")

wordtran


# 연관규칙 발견

tranrules <- apriori(wordtran,

                     parameter = list(supp = 0.15, conf = 0.99))

tranrules

# supp = : 지지도 , conf = : 신뢰도


# 연관규칙 생성 결과보기

detach(package:tm, unload = TRUE)

inspect(tranrules) # inspect() 연관도 분석 함수


# 연관단어 시각화를 위한 자료구조 변경

l_rules <- labels(l_wordtran_rule, ruleSep = " ")

# levels에 대한 문자형 벡터를 지정, 벡터에 있는 각각 원소의 값을 다른 문자형 유형으로 변경


# 문자열로 묶인 연관 단어를 행렬구조로 변경

rules <- labels(tranrules, ruleSep = " ")

rules

# 행 단위로 묶어서 matrix로 변환

rulemat <- do.call("rbind", rules)

class(rulemat)

# 연관어 시각화를 위한 igraph 패키지 설치와 실행

#install.packages("igraph")
```

```
#library(igraph)


# edgelist 보기
ruleg <- graph.edgelist(rulemat[c(1:22), ], directed = F)
ruleg
# edgelist 시각화
par(family="NanumGothicBold")
plot.igraph(ruleg, vertex.label = V(ruleg)$name,
                   vertex.label.cex = 1.2,
                   vertex.label.color = 'black',
                   vertex.size = 20,
                   vertex.color = 'pink',
                   vertex.frame.color = 'red')
```

```
####뉴스(https://news.daum.net/) 전처리 ####
#install.packages("httr") # http에 요청하고 응답에 필요한 패키지
#library(httr)
#install.packages("XML") # XML 문서처리 패키지
#library(XML)
#install.packages(tm) # TermdocumentMatrix()사용
#library(tm)

# 웹 문서 요청
url <- "http://news.daum.net/"
web <- GET(url)
# HTTP 요청에 관한 함수들: GET(),POST()
# 1) GET 방식으로 HTTP 통신이 사용된 경우
# - GET 방식은 가장 일반적인 HTTP Request 형태로
# - 웹 브라우저에 다음과 같은 요청 데이터에 대한
# - 인수를 URL(Uniform Resource Locator)을 통해 전송한다.

# 2) POST 방식으로 HTTP 통신이 사용된 경우
# - POST 방식은 URL에 요청 데이터를 기록하지 않고 HTTP 헤더에
# - 데이터를 전송하기 때문에 GET 방식에서의 '?page=1&search=test'와
# - 같은 부분이 존재하지 않는다.
web

# HTML 파싱
html <- htmlTreeParse(web,useInternalNodes = T,trim =T,encoding = "UTF-8")
```

```
rootNode <- xmlRoot(html)

# 태그자료 수집
news <- xpathApply(rootNode,"//a[@class = 'link_txt']",xmlValue)

# 전처리
# gsub() : 자료를 찾아 바꾸는 함수
news_pre <- gsub("[₩r₩n₩t]", ' ', news) #줄바꿈
news_pre <- gsub('[[:punct:]]', ' ', news_pre) #특수문자
news_pre <- gsub('[[:cntrl:]]', ' ', news_pre)
news_pre <- gsub('₩₩d+', ' ', news_pre) #숫자
news_pre <- gsub('[a-z]+', ' ', news_pre) #소문자
news_pre <- gsub('[A-Z]+', ' ', news_pre) #대문자
news_pre <- gsub('₩₩s+', ' ', news_pre) #빈칸
news_pre # 1~ 46번까지만 기사내용임

# 1~46까지의 기사 내용만 변수에 저장
news_data <- c(news_pre[1:46])

# 세종 사전에 단어 추가
user_dic <- data.frame(term = c("이재명", "무단횡단"), tag = 'ncn')
buildDictionary(ext_dic = 'sejong', user_dic = user_dic)

# 수집한 데이터 저장하고 읽기
setwd('C:/Rwork/Data')
```

```
write.csv(news_data, "news_data.csv", quote = F)

news_data <- read.csv("news_data.csv", header = T, stringsAsFactors = F)

str(news_data)

names(news_data) <- c("no", "news_text")

head(news_data)

news_text <- news_data$news_text

news_text


##### (3) 다음 포털사이트의 실시간 뉴스(https://news.daum.net/)를 수집하고 실시간 토픽 분석을 실행하여 단어구름으로 시각화 하고,

#####분석 시점에서 주요 이슈가 무엇인지 설명하시오.####

#install.packages("tm") # 텍스트 분석 패키지

#install.packages("wordcloud2") # 워드클라우드2 시각화 패키지


# 전처리한 파일 토픽분석하기

exNouns <- function(x) { paste(extractNoun(x), collapse = " ")}

news_word <- sapply(news_text, exNouns)

news_word


# 말뭉치 생성후 집계행렬 만들기

newsCorpus <- Corpus(VectorSource(news_word))


# 집계행렬 생성

news_tdm <- TermDocumentMatrix(newsCorpus, control = list(wordLengths = c(4, 16)))


# 데이터프레임 변경
```

```
tdm.df <- as.data.frame(as.matrix(news_tdm))

# 단어구름 시각화 위한 빈도수 구하기
wordResult <- sort(rowSums(tdm.df), decreasing = TRUE)

# 변수와 빈도수 데이터 프레임 만들기
news_name <- names(wordResult)
news.df <- data.frame(word = news_name, freq=wordResult)

# 클라우드 생성
news_cloud2 <- wordcloud2(data = news.df,size = 0.75,color =
"random-light",backgroundColor = "black",rotateRatio = 0.75)
news_cloud2
# size : 글자 크기
# color : 글자색
# backgroundColor : 배경색
# rotateRatio : 글자 회전도
```

첨부 자료

사례연구5(통합본).R

사례연구5(보조1).R

사례연구5(보조2).R

사례연구5(보조3).R

Lincoln.txt