텍스트 분석

# 한국어 형태소 분석 패키지 설치

# Rtools 설치

# https://cran.r-project.org/bin/windows/Rtools/index.html

install.packages("Sejong")

install.packages("hash")

install.packages("tau")

install.packages("RSQLite")

install.packages("devtools")

# github 버전 설치

install.packages("remotes")

# 64bit 에서만 동작합니다.

remotes::install\_github('haven-jeon/KoNLP', upgrade = "never", INSTALL\_opts=c("--no-multiarch"))

library(KoNLP)

useSejongDic() # 사전을 구성하는 것

word\_data <- readLines("data/애국가(가사).txt")

word\_data

word\_data2 <- sapply(word\_data, extractNoun, USE.NAMES = F) # 명사만 뽑는 함수

word\_data2

word\_data3 <- extractNoun(word\_data) # 명사만 뽑는 함수

word\_data3

add\_words <- c("백두산", "남산", "철갑", "가을", "달") # 딕셔너리에 단어 추가

buildDictionary(user\_dic=data.frame(add\_words, rep("ncn", length(add\_words))), replace\_usr\_dic=T) # 데이터프레임으로 만들어서 딕셔너리에 단어 추가 // ncn == 비서술성 보통명사

word\_data3 <- extractNoun(word\_data) # 7행의 "철갑을" --> "철갑"

word\_data3

undata <- unlist(word\_data3) # 벡터화

undata

word\_table <- table(undata) # 각 단어가 몇 개인지 파악

word\_table

undata2 <- Filter(function(x) {nchar(x) >= 2}, undata) # 필터라는 함수 / 첫번째 아규먼트는 함수, 두번째는 데이터셋 / 함수 결과가 TRUE인 것만 뽑아냄 / 단어의 길이가 2 이상인 것만 뽑아냄

word\_table2 <- table(undata2)

word\_table2

final <- sort(word\_table2, decreasing = T) # 많이 등장한 순으로 정렬

head(final, 10) # 상위 10개 출력

extractNoun("대한민국의 영토는 한반도와 그 부속도서로 한다") # 명사 추출 // 문장이 하나이기에 벡터로 리턴

SimplePos22("대한민국의 영토는 한반도와 그 부속도서로 한다") # 22개 세부 품사로 분석

SimplePos09("대한민국의 영토는 한반도와 그 부속도서로 한다") # 9개 품사로 분석

# 워드 클라우드 // 보기좋게 시각화

install.packages("wordcloud") # 텍스트 분석 결과를 시각화 // 워드클라우드인이라는 방식으로 시각화 / 정적

install.packages("wordcloud2") # 텍스트 분석 결과를 시각화 // 동적

library(wordcloud)

library(wordcloud2)

(words <- read.csv("data/wc.csv")) # 명사와 출현빈도수가 출력

head(words)

?windowsFonts # 윈도우즈 운영체제가 내장하는 폰트만 쓸 것이면 윈도우즈폰트 함수로 가능

windowsFonts(lett=windowsFont("휴먼옛체"))

wordcloud(words$keyword, words$freq) # 기본폰트로 출력 // 실행할 때마다 위치가 달라짐 // 단순한 이미지

wordcloud(words$keyword, words$freq,family="lett")

wordcloud(words$keyword, words$freq,

min.freq = 2, # 두번 이상 등인 경우에만 워드 클라우드에 표현

random.order = F, # 순서를 랜덤하게 할것인가? F이면 가장 값이 큰 것이 가운데에 온다. 값이 큰 것을 가운데로

rot.per = 0.5, # 로테이션 비율이 0이면 회전을 안시킴, 0.5면 50프로는 회전 /

scale = c(4, 1), # 제일 많은 값 스케일은 4, 가장 작은 값 스케일은 1

family="lett", # 글씨체 변화 // 값이 같으면 가은 색

colors = rainbow(15)) # 글자색은 무지개색으로 7개

wordcloud2(words, fontFamily = "휴먼옛체") # 동적 /

wordcloud2(words,rotateRatio = 1) # 전부 회전을 줌

wordcloud2(words,rotateRatio = 0.5)

wordcloud2(words,rotateRatio = 0)# 회전을 주지 않음

wordcloud2(words,size=0.5,col="random-dark") # 크기 0.5 / 색은 어두운 계열

wordcloud2(words,size=0.5,col="random-dark", figPath="data/peace.png") # 이미지로 저장

wordcloud2(words,size=0.7,col="random-light",backgroundColor = "black") # 배경 검은색, 글자색을 밝게 해 강조

wordcloud2(data = demoFreq) # str(demoFreq) // 워드클라우드용 데이터셋 // 단어가 많으면 타원형으로 간다.

wordcloud2(data = demoFreq, figPath="data/peace.png")

wordcloud2(data = demoFreq, shape = 'diamond') # 마름모꼴로 출력

wordcloud2(data = demoFreq, shape = 'star') # 별모양으로 출력

wordcloud2(data = demoFreq, shape = 'cardioid') # 하트?? 너무 커

wordcloud2(data = demoFreq, shape = 'triangle-forward') # 재생버튼 모양

wordcloud2(data = demoFreq, shape = 'triangle') # 삼각형 모양

result<-wordcloud2(data = demoFreq, shape = 'pentagon') # 오각형 모양 // 변수에 저장 가능

library(htmlwidgets) # 파일에 저장하기 위한 함수

saveWidget(result,"tmpwc.html",selfcontained = T) # T값으로 주면 제대로 표현 X / 부가파일이 없어서

saveWidget(result,"tmpwc2.html",selfcontained = F) # tmpwc.html 라는 파일로 저장

head(demoFreq)

str(demoFreq)

wordcloud(names(final), final) # 빈도수가 최대 3개 이상인 게 디폴트

wordcloud(names(final), final, min.freq = 1) # 디폴트를 1개로 설정

wordcloud2(final)

# 트위터 글 워드클라우드

library(rtweet)

appname <- "edu\_data\_collection"

api\_key <- "RvnZeIl8ra88reu8fm23m0bST"

api\_secret <- "wTRylK94GK2KmhZUnqXonDaIszwAsS6VPvpSsIo6EX5GQLtzQo"

access\_token <- "959614462004117506-dkWyZaO8Bz3ZXh73rspWfc1sQz0EnDU"

access\_token\_secret <- "rxDWfg7uz1yXMTDwijz0x90yWhDAnmOM15R6IgC8kmtTe"

twitter\_token <- create\_token(

app = appname,

consumer\_key = api\_key,

consumer\_secret = api\_secret,

access\_token = access\_token,

access\_secret = access\_token\_secret)

key <- "취업"

key <- enc2utf8(key)

result <- search\_tweets(key, n=100, token = twitter\_token)

str(result)

content <- result$retweet\_text

content <- gsub("[[:lower:][:upper:][:digit:][:punct:][:cntrl:]]", "", content)

content <- gsub("취업", "", content) # 취업으로 검색 --> 취업 단어 삭제

word <- extractNoun(content)

cdata <- unlist(word)

cdata

cdata <- Filter(function(x) {nchar(x) < 6 & nchar(x) >= 2} ,cdata)

wordcount <- table(cdata)

wordcount <- head(sort(wordcount, decreasing=T),30)

par(mar=c(1,1,1,1))

wordcloud(names(wordcount),freq=wordcount,scale=c(3,0.5),rot.per=0.35,min.freq=1,

random.order=F,random.color=T,colors=rainbow(20))

word\_table <- table(wordcount)

wordcloud2(wordcount, fontFamily = "맑은고딕", size=0.5,

color="random-light", backgroundColor="black")

wordcloud2(scale(wordcount, center=F, scale=T), fontFamily = "맑은고딕", size=0.5,

color="random-light", backgroundColor="black")

# scale함수 --> 글자 빈도수 차가 너무 크면 문제, 이를 조정 // 그닥 중요X

# ========================================================

# ========================================================

install.packages("tm")

library(tm)

getSources()

lunch <- c("커피 파스타 치킨 샐러드 아이스크림",

"커피 우동 소고기김밥 귤",

"참치김밥 커피 오뎅",

"샐러드 피자 파스타 콜라",

"티라무슈 햄버거 콜라",

"파스타 샐러드 커피"

)

cps <- VCorpus(VectorSource(lunch))

tdm <- TermDocumentMatrix(cps)

tdm

as.matrix(tdm)

cps <- VCorpus(VectorSource(lunch))

tdm <- TermDocumentMatrix(cps,

control=list(wordLengths = c(1, Inf)))

tdm

(m <- as.matrix(tdm))

rowSums(m)

colSums(m)

com <- m %\*% t(m)

com

install.packages("qgraph")

library(qgraph)

qgraph(com, labels=rownames(com), diag=F,

layout='spring', edge.color='blue',

vsize=log(diag(com)\*800))

library(XML)

html.parsed <- htmlParse("data/TextofSteveJobs.html")

text <- xpathSApply(html.parsed, path="//p", xmlValue)

text

text <- text[4:30]

text

docs <- VCorpus(VectorSource(text))

docs

toSpace <- content\_transformer(function(x, pattern){return(gsub(pattern, " ", x))})

docs <- tm\_map(docs, toSpace, ":")

docs <- tm\_map(docs, toSpace, ";")

docs <- tm\_map(docs, toSpace, "'")

docs[[17]]

docs[[19]]

docs[[17]]$content

docs[[19]]$content

docs <- tm\_map(docs, removePunctuation)

text[17]

docs[[17]]$content

docs <- tm\_map(docs, content\_transformer(tolower))

docs[[17]]$content

docs <- tm\_map(docs, removeNumbers)

docs[[17]]$content

docs <- tm\_map(docs, removeWords, stopwords("english"))

docs[[17]]$content

docs <- tm\_map(docs, stripWhitespace)

docs[[17]]$content

docs <- tm\_map(docs, stemDocument)

docs[[17]]$content

tdm <- TermDocumentMatrix(docs)

tdm

inspect(tdm[50:60, 1:5])

termFreq <- rowSums(as.matrix(tdm))

head(termFreq)

termFreq[head(order(termFreq, decreasing=T))]

barplot(termFreq[termFreq >= 7],

horiz=T, las=1, cex.names=0.8,

col=rainbow(16), xlab="word Frequency", ylab="Words")

# 문서(문장)의 유사도 분석

install.packages("proxy")

library(proxy)

dd <- NULL

d1 <- c("aaa bbb ccc")

d2 <- c("aaa bbb ddd")

d3 <- c("aaa bbb ccc")

d4 <- c("xxx yyy zzz")

dd <- c(d1, d2, d3, d4)

cps <- Corpus(VectorSource(dd))

dtm <- DocumentTermMatrix(cps)

as.matrix(dtm)

inspect(dtm)

m <- as.matrix(dtm)

com <- m %\*% t(m)

com

dist(com, method = "cosine")

dist(com, method = "Euclidean")

# tm 패키지를 활용한 숫자, 특수문자, 불용어 삭제하기

mystopwords <- readLines("data/stopwords\_ko.txt", encoding="UTF-8")

text <- readLines("data/stopwords\_testdata.txt", encoding="UTF-8")

docs <- Corpus(VectorSource(text))

inspect(docs)

docs <- tm\_map(docs, removeNumbers)

inspect(docs)

docs <- tm\_map(docs, removePunctuation)

inspect(docs)

docs <- tm\_map(docs, removeWords, mystopwords)

inspect(docs)

docs2 <- Corpus(VectorSource(text))

tdm1 <- TermDocumentMatrix(docs2, control=list(wordLengths = c(1, Inf)))

as.matrix(tdm1)

tdm2 <- TermDocumentMatrix(docs2, control=list(

removePunctuation = T,

removeNumbers = T,

wordLengths = c(1, Inf),

stopwords=mystopwords))

as.matrix(tdm2)