**GEK 6185 Final Exam**

제출 마감시간: 2021년 6월 8일 22:00

**Q1. [60점] 한국어 자연어 처리에서 형태소 분석을 위해 KoNLP, RMecab 등 외에도 많은 package가 공개되어 있으나, Q1에서는 새로운 NLP4kec 패키지를 활용해 보는 기회를 갖고자 합니다. 대부분 KoNLP를 사용하지만, 명사를 추출해서 사용하기 때문에, 중요한 용언들이 누락되는 단점이 있습니다. 물론 KoNLP에서 제공하는 SimplePos09()나 SimplePos22()를 사용해 용언 분석이 가능하지만,**

**NLP4kec 패키지는 입력된 텍스트를 형태소 분석한 후, 용언을 기본형의 형태로 바꾸는 작업이 자동적으로 수행되는 장점이 있습니다.**

1. **Java 설치를 확인해야 합니다.**

Sys.getenv() # JAVA\_HOME 버전을 확인할 수 있습니다.

Sys.setenv(JAVA\_HOME="C:/Users/gbock/corretto/jdk11.0.10\_9") # 저는 좌측 같이 설정,

Sys.setenv(JAVA\_HOME="C:/Program Files/Java/jre1.8.0\_191") 이런 형태일 수도 있습니다.

아래 명령어로 확인해 볼 수 있습니다.

.jinit()

.jcall("java/lang/System", "S", "getProperty", "java.runtime.version")

# rjava 설치, KoNLP때 설치하였음므로, library(rJava) 실행으로 충분할 수 있습니다.

install.packages("rJava")

library(rJava)

1. **한글 텍스트 분석을 위한 환경을 만듭니다.**

https://github.com/NamyounKim/NLP4kec 접속해 windows 용 파일을 다운받아 ./Data directory에 옮겨놓으시기 바랍니다

install.packages("./Data/NLP4kec\_1.4.0.zip" , repos=NULL, type="win.binary")

library(NLP4kec)

library(tm) # text mining package

1. **주어진 [dictionary]와 [raw\_data] directory를 ./Data 하부에 생성하고, 불용어와 유사어 사전을 만드십시오.**

stopWordDic <- read.csv("./Data/dictionary/stopword\_ko.csv")

synonymDic <- read.csv("./Data/dictionary/synonym.csv")

1. **[5점] Speech.xlsx 파일은 4명의 대통령 노무현, 이명박, 박근혜, 문재인 대통령 취임사를 순서대로 포함하고 있습니다. 연설문의 형태소 분석은 file\_parser\_r() 함수를 사용합니다. 상세한 옵션은 각자 학습해야 합니다. 사용자가 만든 사전은 user\_dictionary.txt로 통일하고 개인별 사용자 사전은 만들지 않습니다.**
2. **[20점] 전처리를 수행합니다.**

* **동의어를 처리합니다.**
* **Corpus를 생성합니다.**

단어 간 space를 추가합니다. (NLP4kec 패키지의 알고리즘 상 필요한 과정)

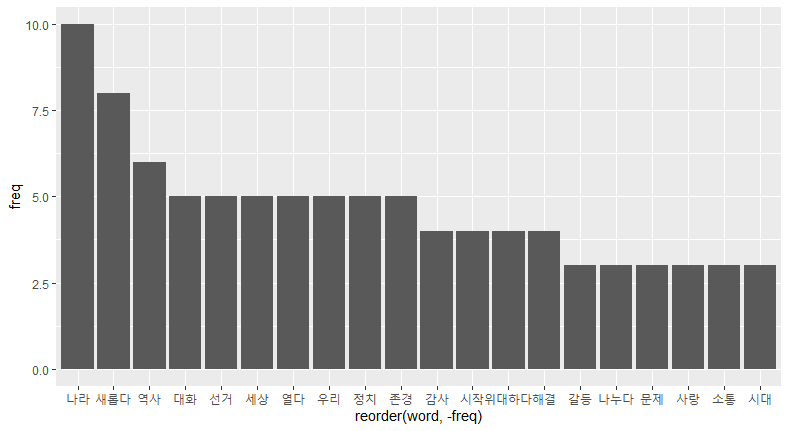
Corpus를 생성합니다.

* **특수문자를 제거합니다.**
* **숫자를 삭제합니다.**
* **소문자로 변경합니다. 영어가 있을 경우를 대비합니다. [실제로 영어는 없음].**
* **불용어 제거: 주어진 불용어 사전을 활용하여 데이터에서 제거합니다.**
* **텍스트 문서 형식으로 변환합니다.**

1. **[5점] DTM(Document Term Matrix)과 TDM(Term Document Matrix)을 생성하시오. (TDM은 DTM에서 행과 열만 바뀐 matrix) (단어 Length는 2로 설정합니다.)**
2. **추후 프로세싱을 위해 단어 양 옆 space 제거 및 한 글자 단어 제외합니다.**
3. **[5점] Sparse Terms 삭제하십시오. (값이 작아질 수록 term수가 줄어드는데, 0.7로 set.)**
4. **DTM, TDM을 데이터프레임 형식으로 저장하십시오.**
5. **단어 발생 빈도를 구하고, 문재인 대통령이 많이 쓴 단어 20개 추출하십시오.**
6. **[5점] 단어 빈도 정보 Data Set 만드십시오.**
7. **[5점] 문재인 대통령이 사용한 상위 20개 단어로 bar chart로 제시하십시오.**

library(ggplot2)

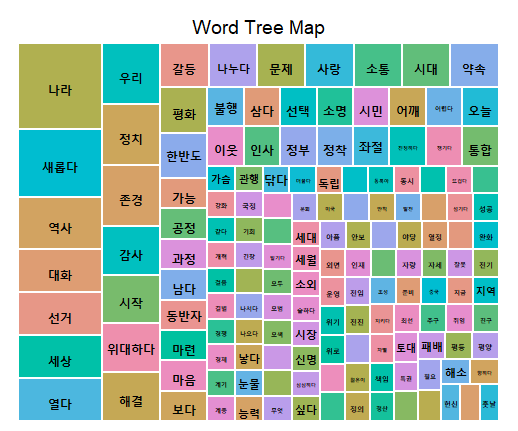
library(dplyr)



1. **[5점] 다음과 같은 문재인 대통령 연설문 treeMap 생성하시오.**

install.packages("treemap")

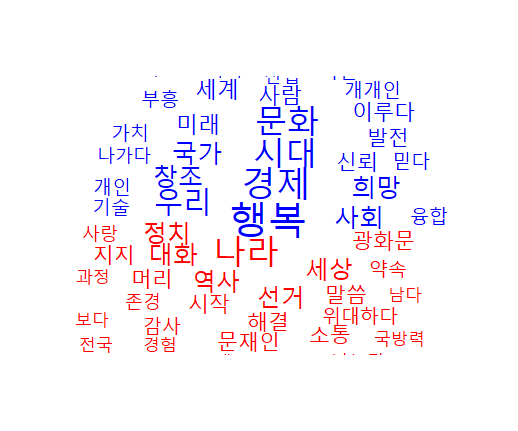
library(treemap)



1. **[5점] 다음과 같이 문재인, 박근혜 대통령 연설문을 word cloud로 비교하십시오.**

install.packages("wordcloud")

library(wordcloud)

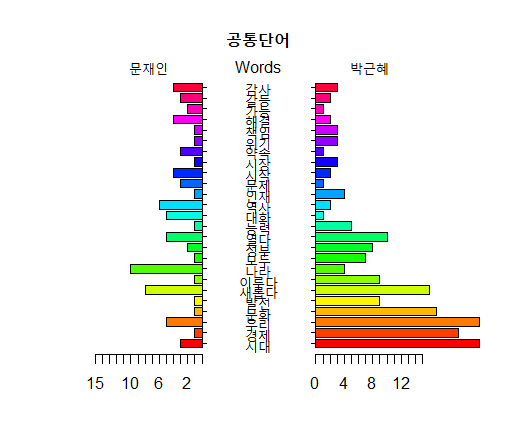


1. **[5점] 문재인과 박근혜 대통령이 사용한 공통단어 피라미드 그래프를 생성하시오.**

install.packages("plotrix")

library(dplyr)

library(plotrix)



install.packages("repr")

install.packages("gridExtra")

install.packages("ggthemes")

install.packages("data.table")

install.packages("readr")

install.packages("textclean")

install.packages("fuzzyjoin")

install.packages("lubridate")

install.packages("ggrepel")

install.packages("ggthemes")

install.packages("chron")

install.packages("reshape2")

install.packages("readxl")

install.packages("wordcloud2")

install.packages("rJava")

install.packages("remotes")

install.packages("nord")

if("stringr" %in% installed.packages("stringr") == FALSE)install.packages("stringr")

if("KoNLP" %in% installed.packages("KoNLP") == FALSE)install.packages("KoNLP")

if("wordcloud" %in% installed.packages("wordcloud") == FALSE)install.packages("wordcloud")

if("RColorBrewer" %in% installed.packages("RColorBrewer") == FALSE)install.packages("RColorBrewer")

devtools::install\_github("lchiffon/wordcloud2")

install\_github("lchiffon/wordcloud2")

install.packages("tm")

install.packages("xlsx")

install.packages("excel.link")

install.packages("caret")

install.packages("mlbench")

install.packages("party")

install.packages("e1071")

install.packages("partykit")

install.packages("randomForest")

install.packages("tree")

install.packages("treemap")

install.packages("plotrix")

install.packages("XLConnect")

require(devtools)

library(devtools)

library(gapminder)

library(tidyverse)

library(datasets)

library(dplyr)

library(ggplot2)

library(knitr)

library(repr)

library(gridExtra)

library(ggthemes)

library(data.table)

library(readr)

library(textclean)

library(fuzzyjoin)

library(lubridate)

library(ggrepel)

library(ggthemes)

library(chron)

library(reshape2)

library(readr)

library(data.table)

library(purrr)

library(readxl)

library(wordcloud)

library(wordcloud2)

library(stringr)

library(RColorBrewer)

library(tm)

library(devtools)

library(xlsx)

library(excel.link)

library(caret)

library(rpart)

library(mlbench)

library(party)

library(e1071)

library(partykit)

library(randomForest)

library(tree)

library(treemap)

library(plotrix)

library(XLConnect)

Sys.getenv() # JAVA\_HOME 버전을 확인할 수 있습니다.

Sys.setenv(JAVA\_HOME="C:/Program Files/Java/jre1.8.0\_291")

library(KoNLP)

install.packages("rJava")

library(rJava)

install.packages("/Data/NLP4kec\_1.4.0.zip" , repos=NULL, type="win.binary")

library(NLP4kec)

.jinit()

.jcall("java/lang/System", "S", "getProperty", "java.runtime.version")

# rjava 설치, KoNLP때 설치하였음므로, library(rJava) 실행으로 충분할 수 있습니다.

stopWordDic <- read.csv("/Data/dictionary/stopword\_ko.csv")

synonymDic <- read.csv("/Data/dictionary/synonym.csv")

stopWordDic

# d. [5점] Speech.xlsx 파일은 4명의 대통령 노무현, 이명박, 박근혜, 문재인 대통령 취임사를 순서대로 포함하고 있습니다.

#연설문의 형태소 분석은 file\_parser\_r() 함수를 사용합니다. 상세한 옵션은 각자 학습해야 합니다.

#사용자가 만든 사전은 user\_dictionary.txt로 통일하고 개인별 사용자 사전은 만들지 않습니다.

parsedData = file\_parser\_r(path = "/Data/speech.xlsx",language = "ko", useEn = T, korDicPath = "./Data/user\_dictionary.txt")

#e. [20점] 전처리를 수행합니다.

#- 동의어를 처리합니다.

parsedData = synonym\_processing(parsedVector = parsedData, synonymDic = synonymDic)

#- Corpus를 생성합니다. 단어 간 space를 추가합니다.

parsedData\_df = data.frame(doc\_id = textData$doc\_id,text = parsedData)

#(NLP4kec 패키지의 알고리즘 상 필요한 과정) Corpus를 생성합니다.

corp = VCorpus(DataframeSource(parsedData\_df))

#- 특수문자를 제거합니다.

corp <- tm\_map(corp, removePunctuation)

#- 숫자를 삭제합니다.

corp = tm\_map(corp, removeNumbers)

#- 소문자로 변경합니다. 영어가 있을 경우를 대비합니다. [실제로 영어는 없음].

corp = tm\_map(corp, tolower)

#- 불용어 제거: 주어진 불용어 사전을 활용하여 데이터에서 제거합니다.

corp = tm\_map(corp, removeWords, stopWordDic$stopword)

#- 텍스트 문서 형식으로 변환합니다.

corp = tm\_map(corp, as.Plain텍스트Document)

#f. [5점] DTM(Document Term Matrix)과 TDM(Term Document Matrix)을 생성하시오.

#(TDM은 DTM에서 행과 열만 바뀐 matrix) (단어 Length는 2로 설정합니다.)

dtm = DocumentTermMatrix(corp, control=list(removeNumbers=FALSE, wordLengths=c(2,Inf)))

tdm = TermDocumentMatrix(corp, control=list(removeNumbers=TRUE, wordLengths=c(2,Inf)))

#g. 추후 프로세싱을 위해 단어 양 옆 space 제거 및 한 글자 단어 제외합니다.

colnames(dtmW) = trimws(colnames(dtmW))

dtmW = dtmW[,nchar(colnames(dtmW)) > 1]

#h. [5점] Sparse Terms 삭제하십시오. (값이 작아질 수록 term수가 줄어드는데, 0.7로 set.)

dtm = removeSparseTerms(dtm, as.numeric(0.98))

#i. DTM, TDM을 데이터프레임 형식으로 저장하십시오.

dtm\_df = as.data.frame(as.matrix(dtm))

tdm\_df = as.data.frame(as.matrix(tdm))

#j. 단어 발생 빈도를 구하고, 문재인 대통령이 많이 쓴 단어 20개 추출하십시오.

freq = colSums(as.matrix(dtm))

length(freq)

freq[head(order(-freq), 20)]

sort(freq, decreasing = T)[1:20]

#k. [5점] 단어 빈도 정보 Data Set 만드십시오.

wordDf = data.frame(word=names(freq), freq=freq)

#l. [5점] 문재인 대통령이 사용한 상위 20개 단어로 bar chart로 제시하십시오.

library(forcats)

top20 = wordDf %>% top\_n(20)

ggplot(top20, aes(x=word, y=freq)) + geom\_bar(stat = "identity") + theme(axis.text.x = element\_text(family = "AppleGothic"))

#m. [5점] 다음과 같은 문재인 대통령 연설문 treeMap 생성하시오.

library(treemap)

top100 = wordDf %>% top\_n(100)

treemap(top100 # 대상 데이터 설정

,title = "Word Tree Map"

,index = c("word") # 박스 안에 들어갈 변수 설정

,vSize = "freq" # 박스 크기 기준

,fontfamily.labels = "AppleGothic" # 맥 폰트 설정

,fontsize.labels = 12 # 폰트 크기 설정

,border.col = "white") # 경계선 색깔 설정

#n. [5점] 다음과 같이 문재인, 박근혜 대통령 연설문을 word cloud로 비교하십시오.

library(wordcloud)

library(RColorBrewer)

wordcloud2(data = top100

, color = "random-light"

, shape = "star"

, size = 0.5

, fontFamily = "나눔고딕")

#o. [5점] 문재인과 박근혜 대통령이 사용한 공통단어 피라미드 그래프를 생성하시오.

par(mfrow=c(1,1))

common\_words\_25 <- wordDf %>%

mutate(label = rownames(wordDf)) %>%

dplyr::filter(D3 > 0 & D4 >0) %>%

mutate(diff = abs(D3 - D4)) %>%

arrange(desc(diff)) %>% slice(1:25)

plotrix::pyramid.plot(common\_words\_25$D3, common\_words\_25$D4,

labels = common\_words\_25$label, gap = 8,

top.labels = c("문재인", "Words", "박근혜"),

main = "Words in Common", laxlab = NULL,

raxlab = NULL, unit = NULL)

**Q2. [20점] 종속변수가 범주형인 경우의 의사결정나무를 살펴보도록 하겠습니다. 자료는 Titanic호 승선자들에 관한 자료입니다. 1912년 4월 10일 영국 사우샘프런 항을 출발한 Titanic호는 뉴욕을 향해 첫 항해를 나섰다가 빙산과 충돌해 타고 있던 2,227멍 중 1,517명이 사망하였습니다. 이 자료의 변수는 Survived, Pclass, Sex, Age, (Siblings, Spouses, Abroad), (Parents, Children, Abroad), Fare 가 있습니다. 그 밖에 자료에 포함되지 않은 ticket, cabin, embarked 등이 있으나 생존여부와 논리적 연결이 없어 제외했습니다. 아울러 이름과(Name) 요금(Fare)는 생존여부와 논리상 연결이 없어 분석에서 제외합니다. Train 데이터는 전체 데이터의 70%로 하며, 추출 시 set.seed(1234)로 합니다.**

R에는 의사결정나무를 분석할 수 있는 패키지가 다수 존재합니다. 그 중 대표적인 3개의 패키지를 꼽자면 **tree, rpart, party** 가 있습니다. 각각의 패키지는 의사결정나무를 만들 때 가지치기하는 방법에 차이가 있습니다. **tree** 패키지는 binary recursive partitioning을, **rpart** 패키지는 CART(classification and regression trees) 방법론을 사용합니다. 이 패키지들은 엔트로피, 지니계수를 기준으로 가지치기를 할 변수를 결정하기 때문에 상대적으로 연산 속도는 빠르지만 과적합화의 위험성이 존재합니다. 그래서 두 패키지를 사용할 경우에는 Pruning 과정을 거쳐서 의사결정나무를 최적화 하는 과정이 필요합니다. **party** 패키지는 Unbiased recursive partitioning based on permutation tests 방법론을 사용합니다. p-test를 거친 Significance를 기준으로 가지치기를 할 변수를 결정하기 때문에 biased 될 위험이 없어 별도로 Pruning할 필요가 없다는 장점이 있습니다. 대신 입력 변수의 레벨이 31개로 제한되어 있다는 단점이 있습니다. 이 문제에서는 **party** 패키지를 사용하도록 합니다.

# 분석에 필요한 패키지 설치

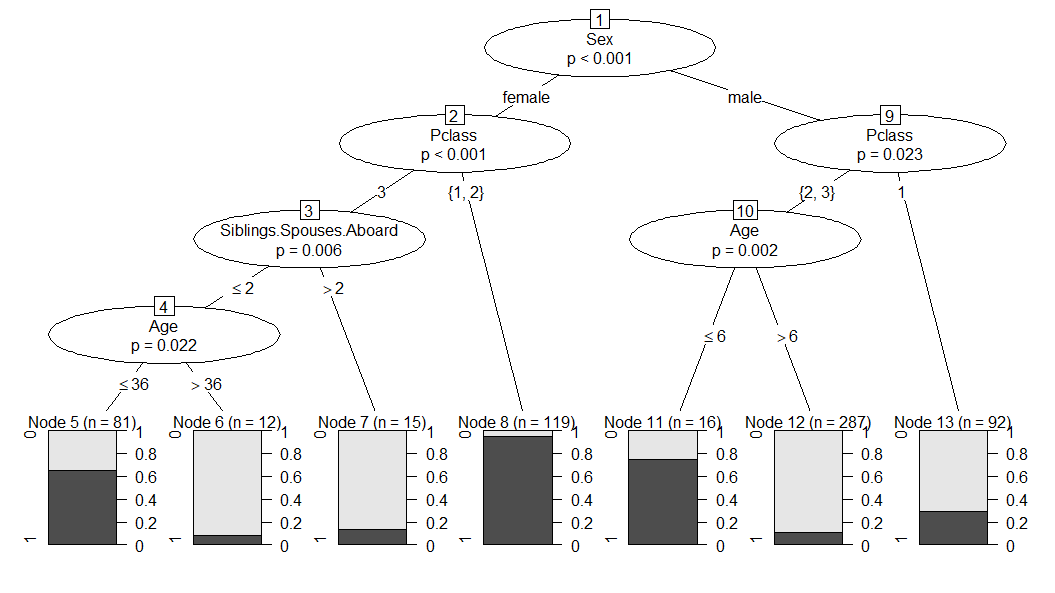
Install.packages(“caret”) # caret 명칭은 "**C**lassification **A**nd **RE**gression **T**raining" 의 앞글자

library(caret)

install.packages("party")

library(party)

1. **[10점] 아래와 같은 의사결정나무를 생성하십시오.**



1. **[10점] Confusion matrix를 생성하고, 만들어진 모형의 정확도를 제시하시오.**

titanic <- read.csv("/Data/rd\_titanic.csv")

set.seed(1234)

training.samples <- pima.titanic$Survived %>% createDataPartition(p = 0.7, list = FALSE)

(training.samples)

train.data <- pima.titanic[training.samples, ]

test.data <- pima.titanic[-training.samples, ]

train.data

test.data

train.data$Sex = as.numeric(train.data$Sex)

test.data$Sex = as.numeric(test.data$Sex)

train.data$Sex <- train.data$Sex %>% as.factor()

#의사결정나무

titanic.survival.train.ctree = ctree(as.factor(Survived ) ~ Pclass + Sex + Age + Siblings.Spouses.Aboard + Parents.Children.Aboard, data=train.data)

titanic.survival.train.ctree

plot(titanic.survival.train.ctree)

titanpred<-predict(titanic.survival.train.ctree, train.data)

confusionMatrix(titanpred, test.data$Survived)

#b. [10점] Confusion matrix를 생성하고, 만들어진 모형의 정확도를 제시하시오.

table(test.cv$Survived, y\_predDT.CV) # Confusion matrix

# 정확도: 0.9290

**Q3. [10점] 의사결정나무와 Random Forest를 비교하기 위해 Random Forest 분석을 실시하고 정확도를 제시하십시오. 의사결정나무 대비 어느 것이 좋습니까?**

library(randomForest)

# Random Forest model

rf\_m <- randomForest(Survived ~ Sex + Age + Pclass , data = train.data)

summary(rf\_m)

# importance

rf\_info <- randomForest(Survived ~ Sex + Age + Pclass , data = train.data, importance = T)

summary(rf\_info)

importance(rf\_info)

varImpPlot(rf\_info)

# 정확도 : 0.8975

**Q4. [10점] titanic 데이터로 로짓회귀분석을 실시하고, 의사결정나무, 랜덤포레스트, 로짓회귀분석 중 어는 것이 가장 좋은 결과를 산출하였는지 설명하시오.**

TitanicLog1 = glm(Survived~., data = train.data, family = binomial)

summary(TitanicLog1)

# 성능 순위 : 의사결정나무 > 로지스틱 회귀분석 > 랜덤포레스트