```
##########################################
### TextMining Lecture 06              #####
### Subject: Text Preprocessing II     #####
### Developed by. KKIM                 #####
##########################################

library(dplyr)
library(magrittr)
library(stringr)

######## Tokenization
son.wiki.doc <-
  "Son Heung-min is a South Korean professional footballer who plays as a forward for
Premier League club Tottenham Hotspur and captains the South Korea national team.
Considered one of the best forwards in the world and one of the greatest Asian
footballers of all time, he is known for his explosive speed, finishing, two-footedness
and ability to link play.
Born in Chuncheon, Gangwon Province, Son relocated to Germany to join Hamburger SV at
16, making his debut in the Bundesliga in 2010. In 2013, he moved to Bayer Leverkusen
for a club record €10 million before signing for Tottenham for £22 million two years
later, becoming the most expensive Asian player in history. While at Tottenham, Son
became the top Asian goalscorer in both Premier League and Champions League history,
and surpassed Cha Bum-kun's record for most goals scored by a Korean player in European
competition. In 2019, he became the second Asian in history to reach and start a UEFA
Champions League final after compatriot Park Ji-sung. In the 2021–22 season, he won,
shared alongside Mohamed Salah, the Premier League Golden Boot award with 23 goals,
becoming the first Asian player to win it."
son.wiki <- "Son Heung-min is a South Korean professional footballer who plays as a
forward for Premier League club Tottenham Hotspur and captains the South Korea national
team."

son.wiki.par <-
  son.wiki.doc %>%
  str_split("\n")
son.wiki.par

son.wiki.sen <-
  son.wiki.par[[1]] %>%
  str_split("\\.")
son.wiki.sen

son.wiki.word <-
  son.wiki.sen[[1]] %>%
  str_split(" ")
son.wiki.word[[1]]

### ngram
# install.packages("ngram")
library(ngram)
son.wiki.ng <-
  son.wiki %>%
  ngram(n=3)
son.wiki.ng
class(son.wiki.ng)
str(son.wiki.ng)

son.wiki.ng %>%
  print(output="truncated") #full

son.wiki %>%
  ngram(n=1, sep="-") %>%
  print(output="full")

son.wiki.ng %>%
  get.phrasetable %>% head

# son.wiki.ng %>%
#   get.phrasetable %>% group_by(ngrams) %>%
```

```
#   summarize(freq=sum(freq)) %>% arrange(desc(freq))

son.wiki.ng %>%
  get.ngrams %>% head

son.wiki.ng %>%
  get.string %>% head

library(RWeka)
son.wiki %>%
  NGramTokenizer(
    Weka_control(min=2, max=3))

######## Normalization
### whitespace
hgu <- "God  is good"
hgu
hgu %>%
  str_replace_all("[:space:]"," ")
hgu %>%
  str_replace_all("[:space:]{1,}"," ")
hgu %>%
  str_replace_all("[:space:]","")
hgu %>%
  str_squish

hgu <- "God             is       good"
hgu.word <-
  hgu %>% str_split(" ")
hgu.word

hgu.word[[1]] %>%
  str_replace_all("[:space:]"," ")
hgu.word[[1]] %>%
  str_squish()
hgu.word[[1]][hgu.word[[1]]!=""]

hgu <- "God             is       good"
hgu.word.not <-
  hgu %>%
  str_squish %>%
  str_split(" ")
hgu.word.not
hgu.word.not %>% unlist
# hgu.word.not[[1]] %>%
#   str_replace_all("[:space:]"," ")
# hgu.word.not[[1]] %>%
#   str_squish()

### numbers
numb.set <-
  c("7 players drank 7 bottles of wine on the 7th week")
numb.set %>%
  str_replace_all("[:digit:]","")

numb.set.word <-
  numb.set %>%
  str_split(" ")
numb.set.word
numb.set.word[[1]] %>%
  str_replace_all("[:digit:]","")

numb.set.word[[1]] %>%
  str_replace("[:alpha:]","")
numb.set %>%
  str_view("[:alpha:]","")

### punctuation
```

```
mark1.17 <-
  '"Come, follow me," Jesus said, "and I will send you out to fish for people."'
mark1.17
mark1.17 %>%
  str_view_all("[:punct:]",
               html=TRUE)
mark1.17 %>%
  str_replace_all("[:punct:]","")

### upper & lower cases
son <- "Son Heung Min's son plays football against Football Club Barcelona"
son %>%
  str_extract_all(boundary("word"))
son %>%
  str_extract_all(boundary("sentence"))
son %>%
  str_extract_all(boundary("line_break"))
son %>%
  str_extract_all(boundary("character"))

son %>%
  str_extract_all(boundary("word")) %>%
  table

son %>%
  tolower %>%
  str_extract_all(boundary("word"))
son %>%
  tolower %>%
  str_extract_all(boundary("word")) %>%
  table

### stopwords
# install.packages("stopwords")
library(stopwords)

stopwords::stopwords_getsources()
stopwords::stopwords_getlanguages("nltk")
stopwords::stopwords("en", source = "nltk")
stopwords::stopwords("en", source = "stopwords-iso")

steve.jobs <- "Your time is limited, so do not waste it living someone else's life."
# Have the courage to follow your heart and intuition. They somehow already know what
you truly want to become."

'%ni%' <- Negate('%in%')
steve.jobs.list <-
  steve.jobs %>%
  str_replace_all("[:punct:]"," ") %>%
  str_replace_all("[:space:]"," ") %>%
  tolower() %>%
  str_split(" ")
steve.jobs.list
steve.jobs.list.1 <-
  steve.jobs.list[[1]][
    steve.jobs.list[[1]] %ni% stopwords("en", source = "nltk")]
steve.jobs.list.1 <-
  steve.jobs.list.1[steve.jobs.list.1 %ni% ""]
steve.jobs.list.1

# finding possible stopwords
abs.1 <- "The present study aims to explore how processes of local knowledge bases have
been altered in this transformative environment and how these have impacted on local
employment growth."
abs.2 <- "This study explores the relationship between the knowledge recombination
types of exploitation and exploration and regional technical efficiency by using the
empirical data sets combining EPO PATSTAT, Eurostat, and Cambridge Econometrics
regional database."
```

```
abs.3 <- "For the empirical study, a structural equation modeling is employed by using
survey material gathered from South Korea in the early days of the COVID-19 outbreak."

abs.1.list <- abs.1 %>%
  str_replace_all("[:punct:]"," ") %>%
  str_replace_all("[:space:]"," ") %>%
  tolower() %>%
  str_split(" ")
abs.1.list <-
  abs.1.list[[1]][abs.1.list[[1]] %ni% ""]

abs.2.list <- abs.2 %>%
  str_replace_all("[:punct:]"," ") %>%
  str_replace_all("[:space:]"," ") %>%
  tolower() %>%
  str_split(" ")
abs.2.list <-
  abs.2.list[[1]][abs.2.list[[1]] %ni% ""]

abs.3.list <- abs.3 %>%
  str_replace_all("[:punct:]"," ") %>%
  str_replace_all("[:space:]"," ") %>%
  tolower() %>%
  str_split(" ")
abs.3.list <-
  abs.3.list[[1]][abs.3.list[[1]] %ni% ""]

abs.all <- append(abs.1.list, abs.2.list) %>%
  append(abs.3.list)
table(abs.all) %>% sort(decreasing=TRUE)

abs.all[abs.all %ni% stopwords("en", source = "nltk")] %>%
  table %>% sort(decreasing=TRUE)

# add stopwords
new.stopwords <- stopwords("en", source = "nltk") %>%
  append(c("study","empirical"))
new.stopwords
abs.all[abs.all %ni% new.stopwords] %>%
  table %>% sort(decreasing=TRUE)

# Stemming
# install.packages("textstem")
library(textstem)
library(stringr)

abs.1
str_split(abs.1," ")
abs.1 %>% str_split(" ") %>% .[[1]]
abs.1 %>% str_split(" ") %>%
  .[[1]] %>%
  stem_words()

# Lemmatization
str_split(abs.1," ")
abs.1 %>% str_split(" ") %>% .[[1]]
abs.1 %>% str_split(" ") %>% .[[1]] %>%
  lemmatize_words()


######## Text Mining with tm
library(dplyr)
library(tm)
# VCorpus: creates corpus

options(encoding = "UTF-8")
res.set <-
  VCorpus(DirSource("R data/awe_research"))
```

```
summary(res.set)
inspect(res.set)

as.VCorpus(list("a","b"))

res.set[[1]]$content
res.set[[1]] %>% as.character %>%
  writeLines
res.set[[1]]$meta

# meta(res.set[[2]], tag='author') <-'Kim'
# res.set[[2]]$meta

### text pre-processing with tm: tm_map
library(textstem)
res.set[[1]]$content
res.set.sed <- res.set %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument) %>%
  # tm_map(content_transformer(stem_strings)) %>%
  tm_map(content_transformer(tolower))
res.set.sed
res.set.sed[[1]]$content

res.set[[1]]$content
res.set.led <- res.set %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(lemmatize_strings)) %>%
  tm_map(content_transformer(tolower))
res.set.led[[1]]$content

### Tokenization
res.set[[2]]$content %>%
  Boost_tokenizer %>% head(30)

res.set[[2]]$content %>%
  MC_tokenizer %>% head(30)

res.set[[2]]$content %>%
  scan_tokenizer %>% head(30)

space_tokenizer <- function(x){
  unlist(strsplit(as.character(x), "[[:space:]]+"))
}
# space_transformer <- content_transformer(function(x, pattern){
#   return(gsub(pattern," ", x))
# })
res.set.ted <- res.set %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(lemmatize_strings)) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeSpecialChars)) %>%
  tm_map(content_transformer(removeWords),
         stopwords::stopwords('en', source='stopwords-iso')) %>%
  tm_map(content_transformer(str_squish)) %>%
  tm_map(content_transformer(space_tokenizer))
res.set.ted[[1]]$content

### Replace special characters
res.set[[16]]$content %>%
  str_extract_all("[[:alnum:]]{1,}[[:punct:]]{1}?[[:alnum:]]{1,}")
```

```
removeSpecialChars <-
  function(x) gsub("’","",x)
  # function(x) gsub("[^a-zA-Z0-9 ]","",x)

res.set.led <- res.set %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(lemmatize_strings)) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeSpecialChars))
res.set.led[[1]]$content


###### Text pre-processing checkpoints
# Find words with punctuation
res.set[[16]]$content
res.set.led[[16]]$content


res.set[[16]]$content %>%
  str_extract_all("[[:alnum:]]{1,}[[:punct:]]{1}[[:alnum:]]{1,}")

ExtractPuncWords <- function(x){
  str_extract_all(x$content,
                  "[[:alnum:]]{1,}[[:punct:]]{1}?[[:alnum:]]{1,}")
}
lapply(res.set, ExtractPuncWords)

# Find words with numbers
res.set[[4]]$content
res.set.led[[4]]$content

res.set[[4]]$content %>%
  str_extract_all("[[:graph:]]{0,}[[:digit:]]{1,}[[:graph:]]{0,}")
res.set[[4]]$content %>%
  str_extract_all("[[:graph:]]{1,}[[:digit:]]{1,}[[:graph:]]{1,}")

ExtractNumText <- function(x){
  str_extract_all(x$content,
                  "[[:graph:]]{0,}[[:digit:]]{1,}[[:graph:]]{0,}")
}
lapply(res.set, ExtractNumText)

# Find words with capital letters
res.set[[16]]$content
res.set.led[[16]]$content

res.set[[16]]$content %>%
  str_extract_all("[[:upper:]]{1}[[:alnum:]]{1,}")

ExtractCapText <- function(x){
  str_extract_all(x$content,
                  "[[:upper:]]{1}[[:alnum:]]{1,}")
}
lapply(res.set, ExtractCapText)

###
lapply(res.set.ted, unlist)
res.set.ted[[1]]$content

res.set.ted <- res.set %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(lemmatize_strings)) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeSpecialChars)) %>%
```

```
  tm_map(content_transformer(removeWords),
         stopwords::stopwords('en', source='stopwords-iso')) %>%
  tm_map(content_transformer(space_tokenizer))

res.set.ted.wo <- res.set %>%
  tm_map(content_transformer(space_tokenizer))

res.set.ted.wo[[1]]$content %>% length
res.set.ted[[1]]$content %>% length

res.set.ted.wo[[1]]$content %>% unique %>% length
res.set.ted[[1]]$content %>% unique %>% length

i<-1
res.set.ted.wo.corp <- NA
res.set.ted.corp <- NA
for(i in 1:length(res.set.ted)){
  res.set.ted.wo.corp <-
    c(res.set.ted.wo.corp, res.set.ted.wo[[i]]$content)
  res.set.ted.corp <-
    c(res.set.ted.corp, res.set.ted[[i]]$content)
}
res.set.ted.wo.corp <-
  res.set.ted.wo.corp[res.set.ted.wo.corp != 0]
res.set.ted.corp <-
  res.set.ted.corp[res.set.ted.corp != 0]

res.set.ted.wo.corp %>% length
res.set.ted.corp %>% length

res.set.ted.wo.corp %>%
  unique %>% length
res.set.ted.corp %>%
  unique %>% length

res.set.ted.wo.corp %>%
  table %>% data.frame %>%
  dplyr::arrange(desc(Freq)) %>%
  head(10)

res.set.ted.corp %>%
  table %>% data.frame %>%
  dplyr::arrange(desc(Freq)) %>%
  head(10)
```