```r
#############################################
### TextMining Lecture 10              #####
### Subject: Sentiment Analysis I      #####
### Developed by. KKIM                 #####
#############################################

# install.packages('tidytext')
# install.packages('textdata')

library(dplyr)
library(magrittr)
library(stringr)
library(tm)
library(tidytext)

# library(remotes)
# install_github("EmilHvitfeldt/textdata")
# install_github("juliasilge/tidytext")
# textdata::lexicon_nrc(delete = TRUE)
textdata::lexicon_nrc()

sentiments

get_sentiments('afinn')
get_sentiments('bing')
get_sentiments('nrc')
get_sentiments('loughran')

get_sentiments('afinn') %>% filter(word=='abandon')
get_sentiments('bing') %>% filter(word=='abandon')
get_sentiments('nrc') %>% filter(word=='abandon')

get_sentiments('afinn') %>% filter(value==5)
get_sentiments('afinn') %>% filter(value==0)
get_sentiments('afinn') %>% filter(value==-5)

get_sentiments('bing') %>% filter(sentiment=='negative')
get_sentiments('bing') %>% filter(sentiment=='positive')

get_sentiments('nrc') %>% filter(sentiment=="trust")
get_sentiments('nrc') %>% filter(sentiment=="fear")
get_sentiments('nrc') %>% filter(sentiment=="sadness")
get_sentiments('loughran') %>% filter(sentiment=="positive")
get_sentiments('loughran') %>% filter(sentiment=="negative")
get_sentiments('loughran') %>% filter(sentiment=="uncertainty")
get_sentiments('loughran') %>% filter(sentiment=="litigious")
get_sentiments('loughran') %>% filter(sentiment=="superfluous")

### Jane austen
# install.packges(janeaustenr)
library(janeaustenr)

austen_books() %>%
  select(book) %>% unique
austen_books() %>%
  select(text) %>% unique %>%
  data.frame %>%
  head(30)

pp <- austen_books() %>%
  filter(book == "Pride & Prejudice" & text !="")
pp %>% head

pp %<>% filter(text != "PRIDE AND PREJUDICE") %>%
  filter(text != "By Jane Austen")
pp %>% head

# Create Chapter column
pp %>% head
pp.ch.index <- which(substr(pp$text,1,7)=="Chapter")
pp.ch.index
pp$text[pp.ch.index]

pp$ch <- 0
for(i in 1:length(pp.ch.index)){
  pp$ch[pp.ch.index[i]:(pp.ch.index[(i+1)]-1)] <- i
}
pp %>%
  filter(ch==max(ch))
pp$ch <-
  ifelse(pp$ch==0, max(pp$ch)+1, pp$ch)
pp %<>%
  filter(substr(text,1,7)!="Chapter")
pp %>%
  filter(ch==max(ch))
```

```
pp %>% head
table(pp$ch)

# Create ch - token table
'%ni%' <- Negate('%in%')
library(textstem)

pp.clean <- 0
for (i in 1:length(unique(pp$ch))){
  temp <- pp %>% filter(ch==i)
  temp.text <- temp$text %>%
    str_split(" ") %>% unlist %>% tolower
  temp.text %<>%
    str_remove_all("[:punct:]") %>%
    lemmatize_words
  temp.text <- temp.text[
    temp.text %ni% stopwords::stopwords('en', source='smart')]
  pp.clean <- rbind(pp.clean,
                    data.frame(ch=i, token=temp.text))
}
pp.clean <- pp.clean[2:nrow(pp.clean),]

# remove "", mr, mrs
pp.clean %<>%
  filter(token!="") %>%
  filter(token!="mr") %>%
  filter(token!="mrs")
pp.clean %>% head

# Create pp.clean.sum
pp.clean.sum <-
  pp.clean %>% group_by(ch) %>%
  count(token) %>%
  arrange(ch,desc(n))
pp.clean.sum %>% head(10)

# unnest_tokens
austen_books() %>%
  filter(book == "Pride & Prejudice" & text !="") %>%
  select(text)

austen_books() %>%
  filter(book == "Pride & Prejudice" & text !="") %>%
  unnest_tokens(word, text)

pp %>%
  unnest_tokens(word, text) %>% data.frame %>% head(5)

austen_books() %>% filter(book == "Pride & Prejudice" & text
!="") %>%
  unnest_tokens(word, text) %>% data.frame %>% head(5)
austen_books() %>% filter(book == "Pride & Prejudice") %>%
  unnest_tokens(word, text) %>% count(word, sort=TRUE)

# check frequency of words
austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  unnest_tokens(word, text) %>%
  count(word) %>% arrange(desc(n))

# Adding linenumber and chapter
austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  mutate(line.number = row_number(),
         chapter = cumsum(str_detect(text, regex("^chapter
[₩₩divxlc]",

ignore_case=TRUE)))) %>%
  unnest_tokens(word, text)

# Sentiment analysis with 'afinn'
pp.word.affin <- austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments('afinn'))
pp.word.affin %>%
  left_join(pp.word.affin %>% count(word))

pp.word.affin %>% group_by(word) %>%
  dplyr::summarise(value=sum(value)) %>%
  arrange(desc(value))
pp.word.affin %>% group_by(word) %>%
  dplyr::summarise(value=sum(value)) %>%
  arrange(value)
```

```
# Sentiment analysis with 'bing'
pp.word.bing <- austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments('bing'))
pp.word.bing %>%
  left_join(pp.word.bing %>% count(word))

pp.word.bing %>% group_by(sentiment) %>%
  dplyr::summarise(count=length(word))
pp.word.bing %>% group_by(sentiment) %>%
  dplyr::summarise(count=length(unique(word)))

pp.word.bing %>% group_by(sentiment, word) %>%
  dplyr::summarise(count=length(word)) %>% ungroup %>%
  group_by(sentiment) %>% arrange(desc(count)) %>%
  slice(1:5)

# Sentiment analysis with 'nrc'
pp.word.nrc <- austen_books() %>%
  filter(book == "Pride & Prejudice") %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments('nrc'))
pp.word.nrc %>%
  left_join(pp.word.nrc %>% count(word))

pp.word.nrc %>%
  left_join(pp.word.nrc %>% count(word)) %>%
  group_by(sentiment) %>% dplyr::summarise(n=sum(n)) %>%
  arrange(desc(n))
pp.word.nrc %>%
  left_join(pp.word.nrc %>% count(word)) %>%
  group_by(sentiment) %>% dplyr::summarise(n=sum(n)) %>%
  arrange(n)

# Visualization

##### ML-based approach
load(file="R file/R file_LEC10/hs.df.RData")

hs.df %>% head(1)
hs.df %>% nrow
hs.df %>% dplyr::filter(type=="train") %>%
  dplyr::filter(label=="Happy") %>% nrow
hs.df %>% dplyr::filter(type=="train") %>%
  dplyr::filter(label=="Sad") %>% nrow
hs.df %>% dplyr::filter(type=="test") %>%
  dplyr::filter(label=="Happy") %>% nrow
hs.df %>% dplyr::filter(type=="test") %>%
  dplyr::filter(label=="Sad") %>% nrow

hs.df$label %>% table
hs.df$type %>% table
table(hs.df$label, hs.df$type)
hs.df$text

library(dplyr)
library(stringr)
library(magrittr)
library(textstem)
library(tidytext)
hs.df.clean <-
  hs.df %>%
  unnest_tokens(word, text, "words", to_lower=TRUE) %>%
  mutate(word = lemmatize_words(word)) %>%
  anti_join(stop_words, by="word")
hs.df.clean %>% head

hs.df.clean$word.ed <-
  hs.df.clean$word %>%
  str_remove_all("[:digit:]{1,}") %>%
  str_remove_all("[:lower:]{1,}₩₩.[:lower:]{1,}") %>%
  str_remove_all("[:punct:]{1,}") %>%
  str_remove_all("[^a-z0-9]")
hs.df.clean %>% head

hs.df.clean$word.ed %>% table %>%
  data.frame %>%
  arrange(desc(Freq)) %>% head(10)

hs.df.clean %<>%
  filter(word.ed!="")

dim(hs.df.clean)
```

```
# Label vector
hs.df.train.label <-
  hs.df$label[hs.df$type=="train"] %>%
  as.factor
hs.df.test.label <-
  hs.df$label[hs.df$type=="test"] %>%
  as.factor

hs.df$ID %>% unique %>% length
hs.df.train.label %>% length
hs.df.train.dtm %>% as.matrix %>% rownames %>% length

# DTM
hs.df.train.dtm <- hs.df.clean %>%
  filter(type=="train") %>%
  select(ID, word.ed) %>%
  count(ID, word.ed) %>%
  cast_dtm(document=ID, term=word.ed, value=n)
hs.df.train.dtm
row.names(hs.df.train.dtm)
hs.df.train.dtmatrix <-
  hs.df.train.dtm %>% as.matrix

ncol(hs.df.train.dtm)
ncol(hs.df.test.dtm)

# Wrong case
hs.df.test.dtm <- hs.df.clean %>%
  dplyr::filter(type=="test") %>%
  select(ID, word.ed) %>%
  count(ID, word.ed) %>%
  cast_dtm(document=ID, term=word.ed, value=n)
hs.df.test.dtm
hs.df.test.dtmatrix <-
  hs.df.test.dtm %>% as.matrix

# Find list of train set words that are not included in test
set
train.vector <- hs.df.clean %>% dplyr::filter(type=="train")
%>%
  select(word.ed) %>% unique
test.vector <- hs.df.clean %>% dplyr::filter(type=="test") %>%
  select(word.ed) %>% unique
```

```
train.test.missing <-
  train.vector$word.ed[train.vector$word.ed %ni%
test.vector$word.ed]
train.test.missing.df <- data.frame(ID=NA,
                                    word.ed=train.test.missing)

# Solution
# If inner_join is used, then
# you may exclude the text from the train set
hs.df.test.dtm <- hs.df.clean %>%
  dplyr::filter(type=="test") %>%
  select(ID, word.ed) %>%
  rbind(train.test.missing.df) %>%
# full_join(hs.df.clean %>% filter(type=="train") %>%
select(word.ed)) %>%
  count(ID, word.ed) %>%
  cast_dtm(document=ID, term=word.ed, value=n)
row.names(hs.df.test.dtm)

hs.df.test.dtmatrix <-
  hs.df.test.dtm %>% as.matrix
hs.df.test.dtmatrix <-
  hs.df.test.dtmatrix[1:20,]
row.names(hs.df.test.dtmatrix)

# dt <- train(x = hs.df.train.dtmatrix,
#       y = hs.df.train.label,
#       method = "rpart")
# predict(dt, newdata = hs.df.test.dtmatrix)

#### Random Forest
library('caret')
set.seed(1009)

dim(hs.df.train.dtmatrix)
length(hs.df.train.label)
dim(hs.df.test.dtmatrix)
length(hs.df.test.label)

rf.train <- train(x = hs.df.train.dtmatrix,
                  y = hs.df.train.label,
                  method = "ranger")
```

```
rf.train
rf.train.pred <- predict(rf.train)
table(rf.train.pred, hs.df.train.label)

rf.test.pred <-
  predict(rf.train,
          newdata = hs.df.test.dtmatrix)
table(rf.test.pred, hs.df.test.label)
table(rf.test.pred, hs.df.test.label) %>%
  confusionMatrix


#### Neural Network
library(caret)
set.seed(1009)

dim(hs.df.train.dtm)
length(hs.df.train.label)
dim(hs.df.test.dtm)
length(hs.df.test.label)

# neural network
nn.train <- train(x = hs.df.train.dtmatrix,
                  y = hs.df.train.label,
                  method = "nnet")
nn.train
nn.train.pred <- predict(nn.train)
table(nn.train.pred, hs.df.train.label)

nn.test.pred <-
  predict(nn.train,
          newdata = hs.df.test.dtmatrix)
table(nn.test.pred, hs.df.test.label)
table(nn.test.pred, hs.df.test.label) %>%
  confusionMatrix
```