```
##########################################
### TextMining Lecture 07              #####
### Subject: Text Quantification       #####
### Developed by. KKIM                 #####
##########################################

library(dplyr)
library(stringr)
library(tm)
library(ggplot2)

### DTM
ex.df <-
  data.frame(
    doc_id=c("Doc_1","Doc_2"),
    text=c("I love you",
           "I adore you you"))
ex.df
ex.df.dtm <-
  ex.df %>%
  DataframeSource %>%
  Corpus %>%
  DocumentTermMatrix(control =
                       list(wordLengths=c(1, Inf)))

ex.df.dtm %>%
  inspect

# without wordLengths
ex.df %>%
  DataframeSource %>%
  Corpus %>%
  DocumentTermMatrix() %>%
  inspect

ex.df.dtm %>%
  inspect

# weighted TF-IDF
ex.df.dtm.tfidf <-
  ex.df %>%
  DataframeSource %>%
  Corpus %>%
  DocumentTermMatrix(
    control = list(wordLengths=c(1, Inf),
                   weighting=function(x)
                     weightTfIdf(x, normalize = FALSE))) %>%
  inspect
ex.df.dtm.tfidf

ex.df.dtm.tfidf.w <-
  ex.df %>%
  DataframeSource %>%
  Corpus %>%
  DocumentTermMatrix(
    control = list(wordLengths=c(1, Inf),
                   weighting=function(x)
                     weightTfIdf(x, normalize = TRUE))) %>%
  inspect
ex.df.dtm.tfidf.w

ex.df %>%
  DataframeSource %>%
  Corpus %>%
  DocumentTermMatrix(
    control = list(wordLengths=c(1, Inf),
                   weighting=weightTfIdf)) %>%
  inspect
```

```
# Comparison of TF and TF-IDF
comp.table <- data.frame(
  doc = rep(rownames(ex.df.dtm), dim(ex.df.dtm)[2]),
  term = rep(colnames(ex.df.dtm) %>% sort(decreasing=FALSE),
             each=dim(ex.df.dtm)[1]),
  TF = c(0,1,1,1,1,0,1,2),
  TF_IDF = as.vector(ex.df.dtm.tfidf),
  W_TF_IDF = as.vector(ex.df.dtm.tfidf.w) %>% round(3))
comp.table

ex.df.dtm %>% inspect

### TDM
ex.df.tdm <-
  ex.df %>%
  DataframeSource %>%
  Corpus %>%
  TermDocumentMatrix(control =
                       list(wordLengths=c(1, Inf)))
ex.df.tdm %>%
  inspect

ex.df.tdm %>% as.matrix %>%
  data.frame

######## n-gram
library(tm)
library(RWeka)
library('textmineR')

mlk <-
  readLines("R file/R file_LEC07/mlk_speech.txt")
mlk
mlk <- mlk[mlk != " "]
mlk <- mlk[mlk != ""]

mlk.corpus <-
  mlk %>%
  VectorSource %>%
  VCorpus

bigramTokenizer <- function(x) {
  RWeka::NGramTokenizer(x,
                        RWeka::Weka_control(min=2,
                                            max=2))
  }

mlk.corpus %>%
  DocumentTermMatrix(
    control=
      list(tokenize=bigramTokenizer)) %>%
  inspect

mlk.corpus %>%
  TermDocumentMatrix(
    control=
      list(tokenize=bigramTokenizer)) %>%
  inspect

### Text Mining DTM
library(tm)
data("crude")
crude %>% length
crude %>% summary
crude[[2]]$content
```

```
library(textstem)
crude.cleaned <- crude %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords('en')) %>%
  tm_map(stripWhitespace) %>%
  tm_map(content_transformer(lemmatize_strings)) %>%
  tm_map(content_transformer(tolower))
crude.cleaned[[1]]$content

crude.dtm <-
  crude.cleaned %>%
  DocumentTermMatrix()
crude.dtm %>%
  inspect
crude.dtm$dimnames$Terms
crude.dtm$dimnames$Docs
crude.dtm %>%
  as.matrix

# removeSparseTerms
crude.dtm %>%
  removeSparseTerms(0.1) %>%
  inspect

crude.dtm %>%
  removeSparseTerms(0.9) %>%
  inspect

crude.dtm %>%
  removeSparseTerms(0.9) %>%
  as.matrix %>% sum

#
crude.dtm %>%
  removeSparseTerms(0.9) %>%
  findAssocs("oil", 0.8)

# BoW
crude.dtm.mat <-
  crude.dtm %>% as.matrix
crude.dtm.mat[1,1:8]
dim(crude.dtm.mat)
crude.dtm.mat %<>% colSums %>%
  sort(decreasing=TRUE)
crude.dtm.mat[1:10]
crude.dtm.mat.df <-
  data.frame(word=names(crude.dtm.mat),
             freq=crude.dtm.mat)
crude.dtm.mat.df %>%
  head()

library(ggplot2)
crude.dtm.mat.df %>%
  dplyr::filter(freq>10) %>%
  ggplot(aes(x=reorder(word,freq),y=freq)) +
  geom_bar(stat='identity') + coord_flip()

# TF mat
crude.dtm.mat <-
  crude.cleaned %>%
  DocumentTermMatrix %>%
  as.matrix

crude.dtm.mat.df <- 0
for(i in 1:nrow(crude.dtm.mat)){
  temp <- crude.dtm.mat[i,]
  temp <- data.frame(
```

```
    doc=rownames(crude.dtm.mat)[i],
    word=names(temp),
    tf=temp,
    row.names = NULL)
  crude.dtm.mat.df <-
    rbind(crude.dtm.mat.df, temp)
  rm(temp)
}
crude.dtm.mat.df <-
  crude.dtm.mat.df[2:nrow(crude.dtm.mat.df),]

crude.dtm.mat.df %>%
  dplyr::group_by(doc) %>%
  dplyr::arrange(desc(tf)) %>%
  dplyr::slice(1) # dplyr::slice(1:3)

# TF-IDF matrix
crude.dtm.tfidf <-
  crude.cleaned %>%
  DocumentTermMatrix(
    control=list(weighting=function(x)
      weightTfIdf(x, normalize=FALSE))) %>%
  as.matrix

# TF-IDF df
crude.dtm.tfidf.mat.df <- 0
for(i in 1:nrow(crude.dtm.tfidf.mat)){
  temp <- crude.dtm.tfidf.mat[i,]
  temp <- data.frame(
              doc=rownames(crude.dtm.tfidf.mat)[i],
              word=names(temp),
              tfidf=temp,
              row.names = NULL)
  crude.dtm.tfidf.mat.df <-
    rbind(crude.dtm.tfidf.mat.df, temp)
  rm(temp)
}
crude.dtm.tfidf.mat.df <-
  crude.dtm.tfidf.mat.df[2:nrow(crude.dtm.tfidf.mat.df),]

crude.dtm.tfidf.mat.df %>%
  dplyr::group_by(doc) %>%
  dplyr::arrange(desc(tfidf)) %>%
  dplyr::slice(1)

# Visualization
library("ggrepel")
crude.dtm.mat.df %>%
  group_by(doc) %>%
  arrange(desc(tf)) %>%
  slice(1) %>%
  ggplot(aes(x=word,y=tf, fill=doc))+
  geom_bar(stat='identity',
           position='dodge') +
  geom_text(aes(x=word, y=tf, group=doc, label=doc),
                position=position_dodge(0.9)) +
  coord_flip()
# geom_text_repel(aes(word, tf, label = doc)) +

crude.dtm.tfidf.mat.df %>%
  group_by(doc) %>%
  arrange(desc(tfidf)) %>%
  slice(1) %>%
  ggplot(aes(x=word,y=tfidf, fill=doc))+
  geom_bar(stat='identity') +
  geom_text(aes(label=doc)) +
  coord_flip()
```

```
###### wordcloud
crude.dtm
crude.dtm.mat <-
  crude.dtm %>% as.matrix
crude.dtm.mat %<>% colSums %>%
  sort(decreasing=TRUE)
crude.dtm.mat.df <-
  data.frame(word=names(crude.dtm.mat),
             freq=crude.dtm.mat)
head(crude.dtm.mat.df)

set.seed(1004)
library(wordcloud)
wordcloud(
  words=crude.dtm.mat.df$word,
  freq=crude.dtm.mat.df$freq,
  random.order=FALSE,
  colors = brewer.pal(8, "Dark2")
)
# rot.per=0.35, min.freq = 1, max.words=200,

set.seed(1004)
library('wordcloud2')
wordcloud2(
  data=crude.dtm.mat.df,
  size=2.0,
  color='random-dark'
)

# RColorBrewer
display.brewer.all(type="seq")
display.brewer.all(type="div")
display.brewer.all(type="qual")

# Wordcloud for multiple documents
i<-1
for(i in 1:nrow(crude.dtm)){
  temp <-
    crude.dtm[i,] %>%
    as.matrix
  temp %<>% colSums %>%
    sort(decreasing=TRUE)
  temp <-
    data.frame(word=names(temp),
               freq=temp)
  png(file=
        paste0("R file/R file_LEC07/wordcloud/wordcloud_",i,".png"),
      width=600, height=350)
  wordcloud(
      words=temp$word,
      freq=temp$freq,
      min.freq=1,
      random.order=FALSE,
      colors=brewer.pal(12, "Paired"))
  dev.off()
  rm(temp)
}

library(wordcloud2)
library(htmlwidgets)
library(webshot)
webshot::install_phantomjs()
set.seed(1004)
wordcloud2.ls <- list()
for(i in 1:nrow(crude.dtm)){
  temp <-
    crude.dtm[i,] %>%
    as.matrix
```

```
  temp %<>% colSums %>%
    sort(decreasing=TRUE)
  temp <-
    data.frame(word=names(temp),
               freq=temp)
  wordcloud2.ls[[i]] <-
    wordcloud2(
      data=temp,
      size=2.0,
      color='random-dark'
    )
  saveWidget(wordcloud2.ls[[i]],
             paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".html"),
             selfcontained = F)
  webshot(url=paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".html"),
          file=paste0("R file/R file_LEC07/wordcloud2/wordcloud2_",i,".png"),
          delay = 10, vwidth = 2000, vheight = 2000)
  rm(temp)
  print(i)
}

######## wordnet
# install.packages('wordnet') # If not working check bit of Java
library(wordnet)

# Window Users
setDict("C:/Program Files (x86)/WordNet/2.1/dict") # For mac, not necessary

# Mac Users
Sys.setenv(WNHOME = "/opt/homebrew/Cellar/wordnet/3.1_1")
setDict(Sys.getenv("WNHOME"))

getFilterTypes()

# Get a term filter
# ignoring lower and upper cases
word.filter <-
  getTermFilter("ExactMatchFilter",
                "worship",
                ignoreCase = TRUE)
word.filter

# Get index term from a wordnet
# specified by a filter
word.terms <-
  getIndexTerms("VERB", # POS
                maxLimit = -1,
                word.filter)
word.terms

word.terms[[1]] %>%
  getSynonyms()
# word.terms %>%
#   sapply(getSynonyms) %>%
#   unlist
synonyms("worship", "NOUN")
synonyms("worship", "VERB")

word.synsets <-
  getSynsets(word.terms[[1]])
word.synsets

sapply(
  getRelatedSynsets(word.synsets[[1]],
                    pointerSymbol="@"), getWord)
sapply(
  getRelatedSynsets(word.synsets[[2]],
                    pointerSymbol="@"), getWord)
```

```
sapply(
  getRelatedSynsets(word.synsets[[3]],
                    pointerSymbol="@"), getWord)


######## PoS Analysis
# https://ladal.edu.au/tagging.html
library('NLP')
library('openNLP')

sent.ant <-
  annotate('God loves you. You love God.',
           Maxent_Sent_Token_Annotator())
sent.ant

word.ant <-
  annotate('God loves you. You love God.',
           Maxent_Word_Token_Annotator(),
           sent.ant)
word.ant

pos.ant <-
  annotate('God loves you. You love God.',
           Maxent_POS_Tag_Annotator(),
           word.ant)
pos.ant

msg <- 'I love you. The love is all you need.'
sent.1.ant <-
  annotate(msg,
           Maxent_Sent_Token_Annotator())
sent.1.ant
word.1.ant <-
  annotate(msg,
           Maxent_Word_Token_Annotator(),
           sent.1.ant)
word.1.ant
pos.1.ant <-
  annotate(msg,
           Maxent_POS_Tag_Annotator(),
           word.1.ant)
pos.1.ant


pos.1.ant$features[length(sent.1.ant)+1:length(pos.1.ant)] %>%
  unlist %>% table

annotate(msg,
         Maxent_POS_Tag_Annotator(),
         word.1.ant)

library(udpipe)
msg <- 'I love you. The love is all you need.'
msg.pos <-
  udpipe(msg, object='english')
msg.pos
msg.pos %>% filter(token=="love") %>%
  select(doc_id, sentence, token, upos, xpos, feats)

kmsg <- "나는 행복합니다. 나는 햄볶습니다."
kmsg.pos <-
  udpipe(kmsg, object='korean')
kmsg.pos
kmsg.pos %>% filter(token=="나는") %>%
  dplyr::select(doc_id, sentence, token, upos, xpos, feats)
```