Assigment2

Primož Miheljak, Sara Veber

12/30/2021

Task 1 – Text preprocesing

For the first task, prepare a suitable dataset that can be used for clustering and classification.

1. Convert the file into a data frame.

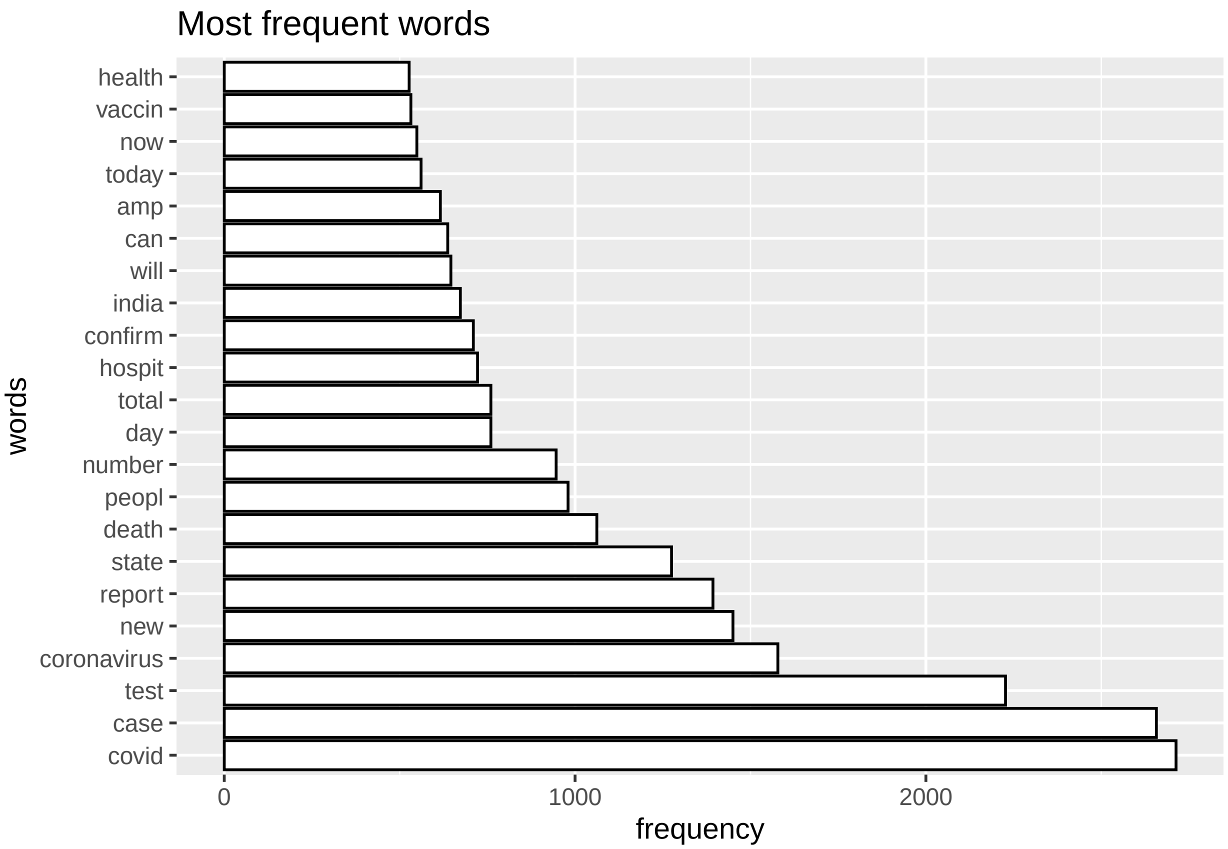
```
train = read_tsv("train_data.tsv", col_names = TRUE)
test = read_tsv("test_data.tsv", col_names = TRUE)

train_data_frame = data.frame(train["label"],train["text_a"])
names(train_data_frame) <-c("labels","text")
#train_data_frame = train_data_frame[1:100,]

test_data_frame = data.frame(test["label"],test["text_a"])
names(test_data_frame) <-c("labels","text")
#test_data_frame = test_data_frame[1:100,]
```

2. Preprocess the text.

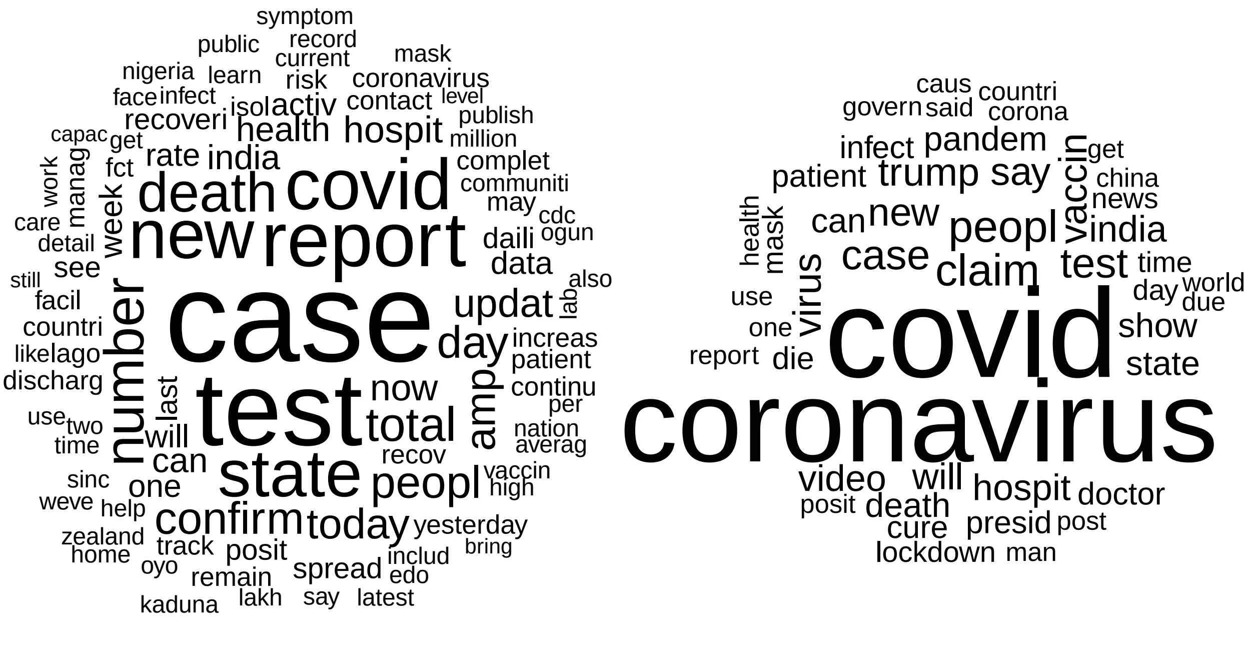
```
preprocess_text <- function(input_text){

    corpus <- Corpus(VectorSource(input_text))

    removeLinks <- function(x)gsub("http\\S*\\s+", '', x)
    corpus <- tm_map(corpus, content_transformer(removeLinks))

    removeLinksEndLine <- function(x)gsub("http\\S*", '', x)
    corpus <- tm_map(corpus, content_transformer(removeLinksEndLine))

    removeHashtag <- function(x)gsub("#\\S*\\s+", '', x)
    corpus <- tm_map(corpus, content_transformer(removeHashtag))

    removeHashtagEndLine <- function(x)gsub("#\\S*", '', x)
    corpus <- tm_map(corpus, content_transformer(removeHashtagEndLine))

    removeRT <- function(x)gsub("RT\\S*", '', x)
    corpus <- tm_map(corpus, content_transformer(removeRT))

    removeAt <- function(x)gsub("@\\S*\\s+", '', x)
    corpus <- tm_map(corpus, content_transformer(removeAt))

    removeAtEndLine <- function(x)gsub("@\\S*", '', x)
    corpus <- tm_map(corpus, content_transformer(removeAtEndLine))
    corpus <- tm_map(corpus, content_transformer(tolower))
    removeSpecialChars <- function(x) gsub("[^a-zA-Z ]", "", x)

    corpus <- tm_map(corpus, content_transformer(removeSpecialChars))
    corpus <- tm_map(corpus, removeNumbers)
    corpus <- tm_map(corpus, removePunctuation)
    corpus <- tm_map(corpus, removeWords, stopwords('english'))
    corpus <- tm_map(corpus, stemDocument)
    corpus <- tm_map(corpus, stripWhitespace)
}
all_text_data <- data.frame(rbind(as.matrix(train_data_frame$text), as.matrix(test_data_frame$text)))
names(all_text_data) <- c("text")
corpus <- preprocess_text(all_text_data$text)
tdm <- TermDocumentMatrix(corpus)
```

3. Visualize the data.

```
allRowNames <- rownames(tdm)
termFrequency <- rowSums(as.matrix(tdm))
termFrequency <- data.frame(sort(subset(termFrequency, termFrequency >= 500)))
termFrequency <-cbind(termFrequency, rownames(termFrequency))
colnames(termFrequency) <- c("val","words")
p <-  ggplot(data=termFrequency, aes(x=reorder(words,-val), y=val)) +
      geom_bar(stat="identity", color="black", fill="white") +
      ggtitle("Most frequent words") + xlab("words") + ylab("frequency")
p + coord_flip()
```



```
par(mar = c(4, 4, .1, .1))

idx = which(train_data_frame$labels == 1)
tdm_real = tdm[,idx]
mat <- as.matrix(tdm_real)
wordFreq <- sort(rowSums(mat), decreasing=TRUE)
grayLevels <- gray((wordFreq+10) / (max(wordFreq)+10))
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=100, random.order=F, colors=grayLevels)
mtext("Real news", side = 3, line = -21, outer = TRUE)

idx = which(train_data_frame$labels == 0)
tdm_fake = tdm[,idx]
mat <- as.matrix(tdm_fake)
wordFreq <- sort(rowSums(mat), decreasing=TRUE)
grayLevels <- gray((wordFreq+10) / (max(wordFreq)+10))
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=100, random.order=F, colors=grayLevels)
mtext("Fake news", side = 3, line = -21, outer = TRUE)
```



Real news Fake news

We observed that eventhough dataset is quite balanced, fake news has fewer words that have frequency higher than 100. We also noticed that as expected most common words are the ones retated to SARS-CoV-2 virus.

Task 2 – Feature construction

After preparing a suitable dataset, we extracted different features we thought could be useful for classification.

1. We added text length as an additional feature.

```
text_length <- nchar(as.character(all_text_data$text))
text_length[is.na(text_length)] <- 0
text_length <- text_length / max(text_length)
text_length <- as.matrix(text_length)
new_features <- text_length
```

```
colnames(new_features)[ncol(new_features)] <- "text_length"
```

2. **We added number of non-asci symbols as an additional feature.**

```
only_asci <- data.frame(matrix(NA, NROW(all_text_data),NCOL(all_text_data)))
names(only_asci) <- "text"

for (i in 1:NROW(all_text_data$text)){
  only_asci$text[i] <- gsub("[^a-zA-Z ]", "", all_text_data$text[i])
}
num_of_nonasci <- nchar(as.character(all_text_data$text)) - nchar(as.character(only_asci$text))
num_of_nonasci <- num_of_nonasci/max(num_of_nonasci)
num_of_nonasci <- as.matrix(num_of_nonasci)
new_features <- cbind(new_features,num_of_nonasci)
colnames(new_features)[ncol(new_features)] <- "num_od_nonasci"
colnames(new_features)
```

```
## [1] "text_length"    "num_od_nonasci"
```

3. **We added number of words that have frequency in specific interval as an additional features.**

```
intervals <- c(0,1,3,5,30,100,300,Inf)
for (i in 1:(length(intervals)-1)){
  termFrequency <- rowSums(as.matrix(tdm))
  termFrequency <- subset(termFrequency, termFrequency <= intervals[i+1])
  termFrequency <- subset(termFrequency, termFrequency > intervals[i])
  only_once_tdm <- as.matrix(tdm[names(termFrequency),])
  tren <- as.matrix(colSums(only_once_tdm))
  tren <- tren/max(tren)
  new_features <- cbind(new_features,tren)
  colnames(new_features)[ncol(new_features)] <- as.character(intervals[i])
}
```

4. **We only used terms that are not too sparse or too common.**

```
minDocFreq <- 10
maxDocFreq <- 800
tdm<- DocumentTermMatrix(corpus, control = list(bounds = list(global = c(minDocFreq, maxDocFreq))))
tdm2 <- as.matrix(tdm)
final <- (cbind(tdm2,new_features))
```

5. **Splitting data into initial train and test set.**

```
len_train = nrow(train_data_frame)
len_test = nrow(test_data_frame)
len_all = nrow(final)

train_features = final[0:len_train,]
test_features = final[(len_train+1):len_all,]

train_data <- cbind(train_features,train_data_frame$labels)
colnames(train_data)[ncol(train_data)] <- "LABELS"
train_data <- data.frame(train_data)
```

6. **Feature selection on train data set (minimum description length).**

```
feature_vals <- attrEval(LABELS ~ ., train_data, "MDL")
```

```
## Changing dependent variable to factor with levels: 0 1
```

```
## Warning in attrEval(LABELS ~ ., train_data, "MDL"): Possibly this is an error
## caused by regression formula and classification attribute estimator or vice
## versa.
```

```
num_features <- 100
threshold <- sort(feature_vals, decreasing=TRUE)[num_features]
selected_features <- c(feature_vals >= threshold, FALSE)
selected_features <- as.matrix(as.matrix(selected_features)[0:(NROW(selected_features)-1)])

X_train <- as.matrix(train_features[, selected_features])
y_train <- as.factor(train_data_frame$labels)

X_test <- as.matrix(test_features[, selected_features])
y_test <- as.factor(test_data_frame$labels)
colnames(X_train)
```

```
##   [1] "data"          "number"        "detail"        "doctor"
##   [5] "itali"         "die"           "man"           "auckland"
##   [9] "communiti"     "contact"       "facil"         "isol"
##  [13] "total"         "indian"        "complet"       "now"
##  [17] "publish"       "see"           "track"         "updat"
##  [21] "yesterday"     "rate"          "recoveri"      "septemb"
##  [25] "confirm"       "presid"        "virus"         "capac"
##  [29] "current"       "manag"         "week"          "continu"
##  [33] "bring"         "activ"         "date"          "daili"
##  [37] "last"          "imag"          "sampl"         "today"
##  [41] "increas"       "learn"         "level"         "cure"
##  [45] "amp"           "china"         "remain"        "day"
##  [49] "video"         "cumul"         "high"          "kill"
##  [53] "drink"         "novel"         "weve"          "bauchi"
##  [57] "borno"         "delta"         "discharg"      "edo"
##  [61] "enugu"         "fct"           "gomb"          "kano"
##  [65] "lago"          "nigeria"       "ogun"          "river"
##  [69] "news"          "photo"         "latest"        "zealand"
##  [73] "chines"        "claim"         "recov"         "kaduna"
##  [77] "donald"        "trump"         "bill"          "gate"
##  [81] "katsina"       "ondo"          "osun"          "oyo"
##  [85] "plateau"       "facebook"      "corona"        "ekiti"
##  [89] "kwara"         "muslim"        "averag"        "text_length"
##  [93] "num_od_nonasci" "0"            "1"             "3"
##  [97] "5"             "30"            "100"           "300"
```

Selected feature set is not surprising. We have expected that among the selected features will be: text_length, num_of_nonasci, and interval features we added before. Other selected features represent words that are important in news.

Task 3 – Modeling

1. We implemented classification accuracy and F1-score.

```
CA <- function(observed, predicted){
  t <- table(observed, predicted)
  return(sum(diag(t))/sum(t))
}
```

```
F1 <- function(observed, predicted) {
  predicted <- factor(as.character(predicted), levels=sort(unique(as.character(observed))))
  cm = as.matrix(table(observed, predicted))
  precision <- cm[2,2] /(cm[1,2]+cm[2,2])
  recall <- cm[2,2] /(cm[2,1]+cm[2,2])
  f1 <-  ifelse(precision + recall == 0, 0, 2 * precision * recall / (precision + recall))
  return(f1)
}
```

2. Naive Bayes classifier.

```
X_train = data.frame(X_train)
X_test = data.frame(X_test)
cm.nb <- CoreModel(y_train ~ ., data = X_train, model="bayes")
predicted <- predict(cm.nb, X_test, type="class")
observed <- y_test
CA_ = CA(observed, predicted)
F1_ = F1(observed,predicted)
```

3. K-NN classifier.

```
X_train = as.matrix(X_train)
X_test = as.matrix(X_test)
predicted <- knn(X_train, X_test, y_train)
observed <- y_test
CA_ = CA(observed, predicted)
F1_ = F1(observed,predicted)
```

4. SVM with a radial basis kernel.

```
model.svm <- ksvm(X_train, y_train, kernel="anovadot", scaled=TRUE)
```

```
##  Setting default kernel parameters
```

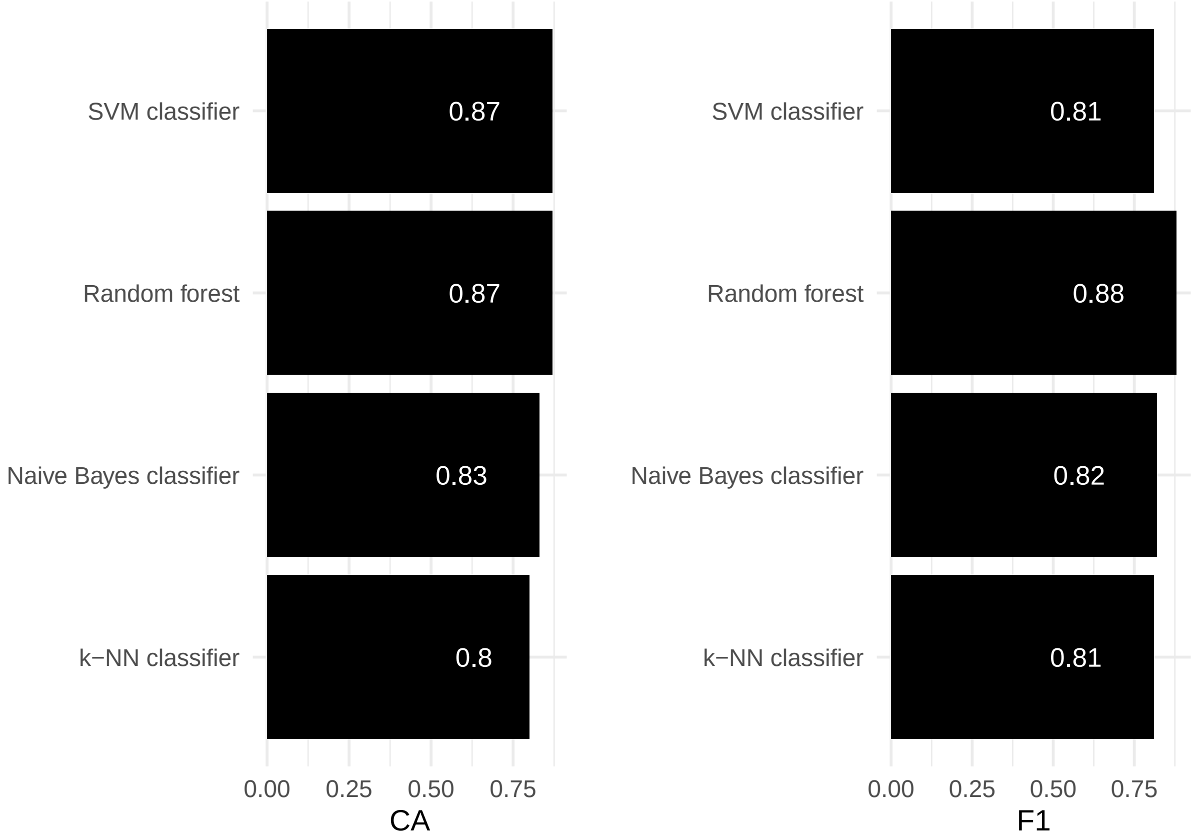
```
predicted <- predict(model.svm, X_test, type="response")
observed <- y_test
CA_ = CA(observed, predicted)
FA_ = F1(observed, predicted)
```

5. Random forest.

```
rf <- randomForest(X_train, y_train)
predicted <- predict(rf, X_test, type="response")
observed <- y_test
CA_ = CA(observed, predicted)
F1_ = F1(observed,predicted)
```

Task 3 – Evaluation

```
ca_scores = round(ca_scores,digits = 2)
f1_scores = round(f1_scores,digits = 2)
df = data.frame(ca_scores,f1_scores, NAMES)

p1<-ggplot(data=df, aes(x=NAMES, y=ca_scores,label = sprintf("%0.2f", round(ca_scores, digits = 2)))) +
  geom_bar(stat="identity", fill="black")+
  geom_text(aes(label=ca_scores), hjust=2, color="white", size=3.5)+
  theme_minimal()+xlab(" ")+ ylab("CA")+
  coord_flip()


p2<-ggplot(data=df, aes(x=NAMES, y=f1_scores,label = sprintf("%0.2f", round(f1_scores, digits = 2)))) +
  geom_bar(stat="identity", fill="black")+
  geom_text(aes(label=f1_scores), hjust=2, color="white", size=3.5)+
  theme_minimal()+xlab(" ")+ ylab("F1")+
  coord_flip()

plot_grid(p1, p2)
```



Used methods perform according to expectations. The worst performing is K-NN model because the feature space is high-dimensional and data points are sparse in it. A bit better is simple Naive Bayes classifier although its performance is still relatively poor. Significantly better are SVM and random forest since they have bigger expressive power and can model more complex feature relationships. Our best model is random forest and it is ranked between doc2vec+LR and word+LR. Yes, we tried different hyperparameter settings. It crucially influenced the results, for example, combining SVM with ANOVA RBF kernel boosted performance for enormous 5%.

```
library(knitr)
Accuracy = c(0.957,0.939,0.929,0.912,0.812,0.523,0.879)
F1 = c(0.956,0.939,0.929,0.911,0.812,0.344,0.886)
Baseline = c("autoML (1h)","MPNet + LR","char + LR","word + LR","doc2vec + LR","majority","our approch"
df = data.frame(Baseline,Accuracy,F1)
kable(df)
```

Baseline	Accuracy	F1
autoML (1h)	0.957	0.956
MPNet + LR	0.939	0.939
char + LR	0.929	0.929
word + LR	0.912	0.911
doc2vec + LR	0.812	0.812
majority	0.523	0.344
our approch	0.879	0.886