LAMPIRAN

Lampiran 1 *Scrapping* Data Menggunakan *Phyton*

Pada lampiran 1, digunakan untuk pengambilan data *tweet* pada twitter menggunakan *Phyton.*

```
import tweepy
from tweepy import Stream
from tweepy import StreamListener
from tweepy import OAuthHandler
import json

#Token
consumer_key = "Uqorv4b6QHvueAt9aJnkphgEF"
consumer_secret =
"uQKEaa2175CGntWuca4vHBwBbW4KslvsQsdSyOlT55sfUGb5O2"
access_token = "920175494-
5ZqjdjwwO9knqkeJ3YlZRjpR2b7h5AT61Jc6PgsV"
access_secret = "wfVN8rOrQV11h2s6EUWmBERxYX3SANFOcgjCpSdXcC1ah"

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
@classmethod
def parse(cls, api, raw):
    status = cls.first_parse(api, raw)
    setattr(status, 'json', json.dumps(raw))
    return status

# menyimpan data dari sebuah model tweet
tweepy.models.Status.first_parse = tweepy.models.Status.parse
tweepy.models.Status.parse = parse
class MyListener(StreamListener):

    def on_data(self, data):
        try:
            with open('datakugabung.json', 'a') as f:
                f.write(data)
                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
        return True

#Mencari kata kunci dalam scrapping
twitter_stream = Stream(auth, MyListener())
twitter_stream.filter(track=['Jokowi','#HidupMahasiswa','#SayaBers
amaJokowi','#TrisaktiTurunLagi'])
```

Lampiran 2 *Syntax Convert Json* ke dalam *Csv*

Mengubah format data yang diperoleh dari ".json" menjadi ".csv" agar lebih mudah menganalisisnya.

```
#########################CONVERT                JSON                TO
CSV###############################
library(jsonlite)
main_sample                                                          =
jsonlite::stream_in(file("D:/SKRIPSI/datakugabung.json"),pagesize
= 100000)
data123= jsonlite::flatten(main_sample)
i <- sapply(data123, is.list)
data123[i] <- lapply(data123[i], as.character)

#########################MEMFILTER     KOLOM     YANG     MAU     DI
AMBIL###############################
t<-
data.frame(data123$text,data123$extended_tweet.full_text,data123$r
etweeted_status.extended_tweet.full_text,data123$user.location,dat
a123$user.screen_name,data123$user.created_at)
#menghapus kolom NA di file location
t=t[!is.na(t$data123.user.location),]
#menghapus kolom NA
library(dplyr)
p1=t[!is.na(t$data123.retweeted_status.extended_tweet.full_text),]
z=t[!is.na(t$data123.extended_tweet.full_text),]
l1=t[with(t,is.na(data123.extended_tweet.full_text)&
is.na(data123.retweeted_status.extended_tweet.full_text)),]
#merubah ke bentuk karakter
k1=as.character(p1$data123.retweeted_status.extended_tweet.full_te
xt)
k2=as.character(z$data123.extended_tweet.full_text)
k3=as.character(l1$data123.text)
k4=as.character(p1$data123.user.location)
k5=as.character(z$data123.user.location)
k6=as.character(l1$data123.user.location)
k7=as.character(p1$data123.user.screen_name)
k8=as.character(z$data123.user.screen_name)
k9=as.character(l1$data123.user.screen_name)
k10=as.character(p1$data123.user.created_at)
k11=as.character(z$data123.user.created_at)
k12=as.character(l1$data123.user.created_at)
mrtgab=data.frame(text=paste(c(k1,k2,k3)),lokasi=paste(c(k4,k5,k6)
),username=paste(c(k7,k8,k9)),waktu=paste(c(k10,k11,k12)))
View(mrtgab)
#save data#
write.csv(mrtgab,file = "D:\\SKRIPSI\\data full gabungan.csv")


## save data
dataframe<-data.frame(text=unlist(sapply(twitclean,          `[`)),
stringsAsFactors=F)
View(dataframe)
datadibagab<-data.frame(dataframe,mrtgab$lokasi,mrtgab$username)
```

```
View(datadibagab)

##menghapus data kosong##
df1=dataframe %>%
  filter(text != ' ')
View(df1)

##menyimpang data yang sudah pre-processing
write.csv(datadibagab,file = "D:/SKRIPSI/data diba sebelum
labeling gab fix.csv")

#######################LABELLING#################################
library(tm)
kalimat2<-df1
kalimat2
#ambil kata kata untuk skoring
positif                  <-                  scan("D:/SKRIPSI/s-
pos.txt",what="character",comment.char=";")
negatif                  <-                  scan("D:/SKRIPSI/s-
neg.txt",what="character",comment.char=";")
kata.positif = c(positif)
kata.negatif = c(negatif)
score.sentiment = function(kalimat2, kata.positif, kata.negatif,
.progress='none')
{
  require(plyr)
  require(stringr)
  scores = laply(kalimat2, function(kalimat, kata.positif,
kata.negatif) {
    kalimat = gsub('[[:punct:]]', '', kalimat)
    kalimat = gsub('[[:cntrl:]]', '', kalimat)
    kalimat = gsub('\\d+', '', kalimat)
    kalimat = tolower(kalimat)

    list.kata = str_split(kalimat, '\\s+')
    kata2 = unlist(list.kata)
    positif.matches = match(kata2, kata.positif)
    negatif.matches = match(kata2, kata.negatif)
    positif.matches = !is.na(positif.matches)
    negatif.matches = !is.na(negatif.matches)
    score = sum(positif.matches) - (sum(negatif.matches))
    return(score)
  }, kata.positif, kata.negatif, .progress=.progress )
  scores.df = data.frame(score=scores, text=kalimat2)
  return(scores.df)
}

#melakukan skoring text
hasil = score.sentiment(kalimat2$text, kata.positif, kata.negatif)
head(hasil)

#CONVERT SCORE TO SENTIMENT
hasil$klasifikasi<-    ifelse(hasil$score>0,"Positif",    ifelse
(hasil$score<0,"Negatif"))
hasil$klasifikasi
View(hasil)
```

```
#CONVERT SCORE TO SENTIMENT
hasil$klasifikasi<- ifelse(hasil$score<0, "Negatif","Positif")
hasil$klasifikasi
View(hasil)

#Tukar Row
data <- hasil[c(3,1,2)]
View(data)
write.csv(data, file = "D:/SKRIPSI/data ter labeli fix.csv")

#Memisahkan twit
data.pos <- hasil[hasil$score>0,]
View(data.pos)
write.csv(data.pos, file = "D:/SKRIPSI/data-pos.csv")

data.neg <- hasil[hasil$score<0,]
View(data.neg)
write.csv(data.neg, file = "D:/SKRIPSI/data-neg.csv")

#negatif
negatif=read.csv("D:\\SKRIPSI\\data-neg.csv",header=TRUE,sep=";")
View(negatif)

#wpositif
negatif=read.csv("D:\\SKRIPSI\\data-pos.csv",header=TRUE,sep=";")
View(negatif)

#r.6
library(tm)
library(RTextTools)
library(e1071)
library(dplyr)
library(caret)
install.packages("maxent")
install.packages("pbkrtest")

df<- read.csv("D:/SKRIPSI/data sudah dilabeli.csv", header = TRUE,
sep = ";")
glimpse(df)


set.seed(1)
df <- df[sample(nrow(df)), ]
df <- df[sample(nrow(df)), ]
glimpse(df)

df$klasifikasi <- as.factor(df$klasifikasi)

corpus <- Corpus(VectorSource(df$text))

corpus

inspect(corpus[1:3])
```

```
dtm <- DocumentTermMatrix(corpus)
inspect(dtm[40:50, 10:15])

df.train <- df[1:2848,]
df.test <- df[2849:3561,]

dtm.train <- dtm[1:2848,]
dtm.test <- dtm[2849:3561,]

corpus.train <- corpus[1:2848]
corpus.test <- corpus[2849:3561]

dim(dtm.train)

fivefreq <- findFreqTerms(dtm.train, 10)
length((fivefreq))
fivefreq
## [1] 12144

# Use only 5 most frequent words (fivefreq) to build the DTM

dtm.train.nb          <-          DocumentTermMatrix(corpus.train,
control=list(dictionary = fivefreq))

dim(dtm.train.nb)
## [1]  1500 12144

dtm.test.nb           <-           DocumentTermMatrix(corpus.test,
control=list(dictionary = fivefreq))

dim(dtm.train.nb)

# Function to convert the word frequencies to yes (presence) and
no (absence) labels
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

# Apply the convert_count function to get final training and
testing DTMs
trainNB <- apply(dtm.train.nb, 2, convert_count)
testNB <- apply(dtm.test.nb, 2, convert_count)

library(naivebayes)
naive=classifier   <-   naiveBayes(trainNB,   df.train$klasifikasi,
laplace = 0)
print(classifier)
View(naive)
# Train the classifier
naive=classifier   <-   naiveBayes(trainNB,   df.train$klasifikasi,
laplace = 0)
print(naive)
View(naive)
# Use the NB classifier we built to make predictions on the test
```

```
set.
system.time( pred <- predict(classifier, newdata=testNB) )
# Create a truth table by tabulating the predicted class labels
with the actual class labels
table("Predictions"= pred,  "Actual" = df.test$klasifikasi )
# Prepare the confusion matrix
conf.mat     <-     confusionMatrix(pred,     df.test$klasifikasi,
positive="Positif")
conf.mat
conf.mat$byClass
conf.mat$overall
conf.mat$overall['Accuracy']
```

Lampiran 3 *Syntax R Analisis Wordcloud*

```
#negatif
negatif=read.csv("D:\\SKRIPSI\\data-neg.csv",header=TRUE,sep=";")
View(negatif)

#wpositif
positif=read.csv("D:\\SKRIPSI\\data-pos.csv",header=TRUE,sep=";")
View(positif)

#wfull
datafull=read.csv("D:\\SKRIPSI\\data                              sudah
dilabeli.csv",header=TRUE,sep=";")
View(datafull)

library(corpus)
library(tm)
library(SnowballC)
library(wordcloud)
library(stringr)

#load the data as a corpus
docs<-Corpus(VectorSource(positif$text))
#Remove your own stop word
#Eliminate extra white spaces
docs<-tm_map(docs,stripWhitespace)
#Build a term-document matrix
dtm<-TermDocumentMatrix(docs)
m<-as.matrix(dtm)
v<-sort(rowSums(m),decreasing=TRUE)
d<-data.frame(word=names(v),freq=v)
head(d,20)

#Build a term-document matrix
memory.limit(size=100000)
{
  dtm <- TermDocumentMatrix(docs)
  m <- as.matrix(dtm)
  v <- sort(rowSums(m),decreasing=TRUE)
  d <- data.frame(word = names(v),freq=v)
}
head(d,n=350)

#generate the word cloud
library (wordcloud)
set.seed(1234)
wordcloud(words=d$word,freq       =       d$freq,min.freq       =
1,max.words=100,random.order=FALSE,rot.per   =   0.35,   colors   =
brewer.pal(8,"Dark2"))

## mencari asosiasi
v<-as.list(findAssocs(dtm,
                      terms= c('takut'),
                      corlimit= c(0.15)))
v
```

Lampiran 4 Hasil Output

```
> # Prepare the confusion matrix
> conf.mat <- confusionMatrix(pred, df.test$klasifikasi, positive="Positif")
> conf.mat
Confusion Matrix and Statistics

          Reference
Prediction Negatif Positif
   Negatif     494      45
   Positif       4     169

               Accuracy : 0.9312
                 95% CI : (0.91, 0.9487)
    No Information Rate : 0.6994
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8269

 Mcnemar's Test P-Value : 1.102e-08

            Sensitivity : 0.7897
            Specificity : 0.9920
         Pos Pred Value : 0.9769
         Neg Pred Value : 0.9165
             Prevalence : 0.3006
         Detection Rate : 0.2374
   Detection Prevalence : 0.2430
      Balanced Accuracy : 0.8908

       'Positive' Class : Positif
```

```
> conf.mat$byClass
         Sensitivity          Specificity       Pos Pred Value       Neg Pred Value
           0.7897196           0.9919679           0.9768786           0.9165121
           Precision              Recall                  F1          Prevalence
           0.9768786           0.7897196           0.8733850           0.3005618
      Detection Rate Detection Prevalence    Balanced Accuracy
           0.2373596           0.2429775           0.8908437
> conf.mat$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull AccuracyPValue
  9.311798e-01   8.268586e-01   9.100338e-01   9.486562e-01   6.994382e-01   6.640926e-53
 McnemarPValue
  1.101658e-08
> conf.mat$overall['Accuracy']
 Accuracy
0.9311798
>
```