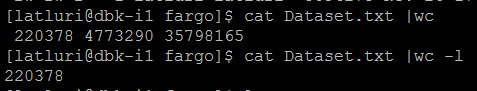
**Deliverable C: code with reference to analytic process flow-diagram from deliverable A**

The given Dataset has 220378 lines, each delivering a single comment.



The command used to remove the data not pertaining to any of the banks(A,B,C,D)

**grep -i 'bankC\|bankB\|bankA\|bankD' Dataset.txt >>Cleaned\_data.txt**

Cleaned Dataset contains 192180 comments

For example, comment no 8 is removed as it contains “8|8/15/2015|2015|8|twitter|i just invested in tsemah on twit\_hndl. listen and invest at INTERNET# tradiio” This could be related to any of the banks but because of the scrubbing, it is impossible for us to find out which bank this is related to or what comment it is referring to. Other junk messages such as 69 “- facebook wife wife? Ÿ ’ Ÿ˜:- \* Ÿ ’ – Ÿ ” ¥ Ÿ ’ ¦ Ÿ ’ ¦” are also removed.

Data containing the tweeter ids of banks are separated.

**grep -i 'twit\_hndl\_Bank' Cleaned\_data.txt >>Bank\_tweeter.txt**

**grep -i -v 'twit\_hndl\_Bank' Cleaned\_data.txt >>Cleaned\_data\_wo\_tweeterid.txt**

**grep -i bankB Cleaned\_data\_wo\_tweeterid.txt>>bankB.txt Note: This would contain other comments such as comments involving bankC with other banks.**

**grep -i 'bankA' Cleaned\_data\_wo\_tweeterid.txt |grep –I -v 'bankB'|grep -i -v 'bankC'|grep -i -v 'bankD' >>only\_bankA.txt**

**grep -i 'bankB' Cleaned\_data\_wo\_tweeterid.txt |grep –I -v 'bankA'|grep -i -v 'bankC'|grep -i -v 'bankD' >>only\_bankB.txt**

**grep -i 'bankC' Cleaned\_data\_wo\_tweeterid.txt |grep –I -v 'bankB'|grep -i -v 'bankA'|grep -i -v 'bankD' >>only\_bankC.txt**

**grep -i 'bankD' Cleaned\_data\_wo\_tweeterid.txt |grep –I -v 'bankB'|grep -i -v 'bankC'|grep -i -v 'bankA' >>only\_bankD.txt**

**grep -i 'bankB' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankC'|grep –I -v 'bankD'|grep -i -v 'bankA'>> bankB\_bankC.txt**

**grep -i 'bankB' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankA'|grep –I -v 'bankC'|grep -i -v 'bankD'>> bankA\_bankB.txt**

**grep -i 'bankC' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankA'|grep –I -v 'bankB'|grep -i -v 'bankD'>> bankA\_bankC.txt**

**grep -i 'bankD' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankA'|grep –I -v 'bankC'|grep -i -v 'bankB'>> bankA\_bankD.txt**

**grep -i 'bankB' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankD'|grep –I -v 'bankC'|grep -i -v 'bankA'>> bankB\_bankD.txt**

**grep -i 'bankC' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankD'|grep –I -v 'bankA’|grep -i -v 'bankB'>> bankC\_bankD.txt**

**grep -i 'bankB' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankC'|grep -i 'bankD'|grep -i -v 'bankA'>> bankB\_bankC\_bankD.txt**

**grep -i 'bankA' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankB'|grep -i 'bankC'|grep -i -v 'bankD'>> bankA\_bankB\_bankC.txt**

**grep -i 'bankA' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankB'|grep -i 'bankD'|grep -i -v 'bankC'>> bankA\_bankB\_bankD.txt**

**grep -i 'bankA' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankC'|grep -i 'bankD'|grep -i -v 'bankB'>> bankA\_bankC\_bankD.txt**

**grep -i 'bankA' Cleaned\_data\_wo\_tweeterid.txt |grep -i 'bankB'|grep -i 'bankC'|grep -i 'bankD'>> bankA\_bankB\_bankC\_bankD.txt**

**Similarly the bank\_tweeter can also be classified.**

To clean the data

Consider the file containing only bankA comments

The following R commands are used to clean the data.

**only\_bankA <- VCorpus(DirSource("<location>"))**

only\_bankA<- tm\_map(tweets, removeWords,'INTERNET')

only\_bankA<- tm\_map(tweets, removeWords,'Name)

**only\_bankA1 <-tm\_map(only\_bankA,content\_transformer(stripWhitespace))**

**only\_bankA2 <-tm\_map(only\_bankA1,content\_transformer(tolower))**

**only\_bankA3 <-tm\_map(only\_bankA2,removeWords,stopwords('English'))**

**only\_bankA4 <-tm\_map(only\_bankA3,content\_transformer(stemDocument))**

This command prepares the word cloud (1-gram)

**wordcloud(words = only\_bankA4,min.freq = 10,max.words = 500,random.order = 'false')**

To plot freq vs the word for top 100 high freq words

**TdmBi\_A <- TermDocumentMatrix(only\_bankA4, control = list(tokenize = BigramTokenizer))**

**TdmBi\_A\_RS <- removeSparseTerms(TdmBi\_A, 0.1)**

**freq.bi\_A <- rowSums(as.matrix(TdmBi\_A\_RS))**

**freq.bi\_A\_S <- sort(freq.bi\_A, decreasing = TRUE)**

**df.freq.bi\_A\_S <- data.frame("Term"=names(head(freq.bi\_A\_S,100)), "Frequency"=head(freq.bi\_A\_S,100))**

**df.freq.bi\_A\_S$Term1 <- reorder(df.freq.bi\_A\_S$Term, df.freq.bi\_A\_S$Frequency)**

**p2\_A <-**

**ggplot(df.freq.bi\_A\_S, aes(x = Term1, y = Frequency)) +**

**geom\_bar(stat = "identity", color="gray55", fill="steelblue2") +**

**geom\_text(data=df.freq.bi\_A\_S,aes(x=Term1,y=-250,label=Frequency),vjust=0, size=3) +**

**xlab("Terms") + ylab("Count") + ggtitle("Top 100 BiGram Tokenized Word Frequency") +**

**theme(plot.title = element\_text(lineheight=.8, face="bold")) +**

**coord\_flip()**

**plot(p2\_A)**

TO obtain bigram word cloud

**wordcloud(words = df.freq.bi\_A\_S$Term1,freq = df.freq.bi\_A\_S$Frequency,random.order=FALSE,rot.per=0.35,use.r.layout=FALSE,colors=brewer.pal(8, "Dark2"))text(x=0.5, y=1.1, "BiGram Word Cloud")**

Further data filtering to remove the words containing special character

gsub('\\S+/\\S+', '', only\_bankA4) to remove words containing special char”/” such as http:/

Furthermore, we can use the set of positive and negative words to compare and rate the comments as negative or positive.

score.sentiment <- function(sentences, pos.words, neg.words, .progress='none')

{

require(plyr)

require(stringr)

scores <- laply(sentences, function(sentence, pos.words, neg.words){

sentence <- gsub('[[:punct:]]', "", sentence)

sentence <- gsub('[[:cntrl:]]', "", sentence)

sentence <- gsub('\d+', "", sentence)

sentence <- tolower(sentence)

word.list <- str\_split(sentence, '\s+')

words <- unlist(word.list)

pos.matches <- match(words, pos.words)

neg.matches <- match(words, neg.words)

pos.matches <- !is.na(pos.matches)

neg.matches <- !is.na(neg.matches)

score <- sum(pos.matches) - sum(neg.matches)

return(score)

}, pos.words, neg.words, .progress=.progress)

scores.df <- data.frame(score=scores, text=sentences)

return(scores.df)

}

pos <- scan('C:/location/positive-words.txt', what='character', comment.char=';')

neg <- scan('C:/location/negative-words.txt', what='character', comment.char=';')

Dataset <- stack

Dataset$text <- as.factor(only\_bankA4$text)

scores <- score.sentiment(only\_bankA4$text, pos.words, neg.words, .progress='text')

write.csv(scores, file=paste(score, '\_scores.csv'), row.names=TRUE)

stat<-scores

stat$created<-stack$created

stat$created<-as.Date(stat$created)

stat <- mutate(stat, tweet=ifelse(stat$score > 0,'positive',ifelse(stat$score < 0, 'negative', 'neutral')))

data\_2<-group\_by(stat, data, created)

data\_2<-summarise(data\_2, number=n())

write.csv(data\_2, file=paste(only\_banA4$V1, '\_opin.csv'), row.names=TRUE)

The output would give positive or negative for each element. This when synchronized with the data would look something like this.

