Topic modeling and sentiment analysis using Tweets on Bangladesh

Arafath Hossain

Introduction

The objective of this analysis was to get an idea about what topic related to Bangladesh that people like to talk about and how they feel overall.

Objective

Breaking down the objective for clear analysis gives us three specific goals:

1. Around which topics twitter discussions usually circle around,
2. What is the overall sentiment about Bangladesh that is conveyed by the tweets,
3. As an extension of the previous two steps: A topic wise sentiment analysis to reveal what kind of sentiments(s) carried by the generally discussed topics.

Data collection

For that study I used public API that is provided from Twitter for twitter analysis. I fetched total 20,000 random twitter posts that were in English and had a mention of 'Bangladesh'. So I used 'Bangladesh' as the search term and collected total 20,000 twitters using R through the public API of Twitter.

Discussion of the methodology

To achieve this objective I applied Latent Dirichlet allocation (LDA) model from topicmodels package in R. LDA model is an unsupervised machine learning algorithm which was first presented as a topic discovery model by David Blei, Andrew Ng, and Michael I. Jordan in 2003. LDA considers a document as a collection of topics. So each word in the document is considered as part of one or more topics. LDA clusters the words under their respective topics. As a statistical model, LDA provides probability of each word to be belonging to a topic and again a probability of each topic to be belonging to each document. To run LDA, we have to pick number of topics. Since, based on this number LDA breaks down a document and words, in this study, I will try two different total numbers of topics. In LDA model, what could be the total number of topics to be looked for is a balance between granularity versus generalization. More number of topics can provide granularity but may become difficult to divide in clearly segregated topics. On the other hand, less number of topics can be overly generalized and may combine different topics into one. On the later part for sentiment analysis lexicon based sentiment analysis approach was followed. The lexicon used was NRC Emotion Lexicon (EmoLex) which is a crowd-sourced lexicon created by Dr. Saif Mohammad, senior research officer at the National Research

Council, Canada. NRC lexicon has a division of words based on 8 prototypical emotions namely: trust, surprise, sadness, joy, fear, disgust, anticipation, and anger and two sentiments: positive and negative. This NRC lexicon was used from the 'tidytext' package in R.

Data analysis:

Data processing and analysis will be discussed under this section.

Data processing

I have already harvested the tweets and fetched texts from the tweets into text file: 'bd_tweets_20k.Rds'. So I will skip the initial part of coding showing fetching tweets. Rather I will start by reading the already saved file and then will show the data cleaning and processing step by step.

Reading text file of the tweets:

```
tweets = readRDS('bd_tweets_20k.Rds')
```

Before going into the data cleaning step couple of things are to be cleared:

- It's very important to maintain logical order in executing the cleaning commands. Other wise some information can be missed unintentionally. For example, if we convert all the tweets and later on apply 'gsub' command to remove retweets with 'rt' pattern we may lose part of words that contain 'rt'. Retweets are marked as RT in the begining but since we converted everything into lower case using 'tolower' function, lateron our programs would not be able to detect difference of 'rt' for retweet and any other use of 'rt' as part of a word. Let's say there's a word 'Part', after the transormation we'll only see 'Pa' and 'rt' part will be replaced by blank.
- Throughout the cleaning step it's a good practice to randomly check the text file to make sure no unexpected transformation takes place. For example, I will view 500th tweet from my file as a benchmark. That tweet I picked arbitrarilly. I will check text of that tweet before starting the data cleaning process and also will view at different points during the cleaning steps.

Here's our sample tweets:

```
writeLines(as.character(tweets[[1500]]))

## Half a million Rohingya refugee children at risk in overcrowded camps in B
angladesh with cyclone and� https://t.co/jrp3yEvMJN #Bangladesh
```

We will revisit our sample tweet at different points during the next data cleaning process.

In the following section I start data cleaning process by converting the text to ASCII format to get rid of the funny characters usually used in Twitter messages. Here is one thing can be noted that these funny characters may contain significant subtle information about sentiment carried by the messages but since it will extend the area covered by this report, it has been skipped here. But it could definitely be a future research area! Before going any further

```
# Convert to basic ASCII text to avoid silly characters
tweets <- iconv(tweets, to = "ASCII", sub = " ")
```

On the following code section, I will apply bunch of codes to remove special characters, hyperlink, usernames, tabs, punctuations and unnecessary white spaces. Because all these are not don't have any relation to the topic modeling. I have mentioned specific use of each code along with the codes below.

```
tweets <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", tweets)  # Remove the "RT"
(retweet) and usernames
tweets = gsub("http.+ |http.+$", " ", tweets)  # Remove html links
tweets = gsub("http[[:alnum:]]*", "", tweets)
tweets = gsub("[[:punct:]]", " ", tweets)  # Remove punctuation
tweets = gsub("[ |\t]{2,}", " ", tweets)  # Remove tabs
tweets = gsub("^ ", "", tweets)  # Leading blanks
tweets = gsub(" $", "", tweets)  # Lagging blanks
tweets = gsub(" +", " ", tweets) # General spaces
```

In the above bunch of cleaning codes we have removed html, username and so on. We saw our sample tweet had a html link in it. Let's check if the transoformation worked properly or not:

```
writeLines(as.character(tweets[[1500]]))

## Half a million Rohingya refugee children at risk in overcrowded camps in B
angladesh with cyclone and
```

We see that the punchtuations (.) and website link have been removed from our sample tweet as intended.

I will convert all the tweets in lower case since in R words are case sensitive. For example: 'Tweets' and 'tweets' are considered as two different words. Moreover, I will remove the duplecate tweets. Among the tweets downloaded using twitter public API there duplicate tweets also exist. To make sure the tweets that are used here are not duplicated now I will remove the duplicated tweets.

```
tweets = tolower(tweets)
tweets = unique(tweets)
writeLines(as.character(tweets[[1500]]))

## best quality underground metal detector in bangladesh
```

To check I have extracted the 1500th tweet. But this time I have got a different tweet. Because after removing duplecate tweets I had left with 5,561 tweets out of 20,000 tweets which I started with. So the serial number of tweets have also changed.

As the next step of data processing I will convert this tweets file, which is a character vector, into a corpus. In general term, corpus in linguistic means a structured set of texts that can be used for statistical analysis, hypothesis testing, occurance checking and validating linguistic rules. To To achive this goal I will use ‘corpus’ and ‘VectorSource’ commands from ‘tm’ library in R. While ‘VectorSource’ will interpret each element of our character vector file ‘tweets’ as a document and feed that input into ‘corpus’ command. Which eventually will convert that into corpus suitable for statistical analysis.

```
library(tm)
corpus <- Corpus(VectorSource(tweets))
```

I will do some more cleaning on the corpus by removing stop words and numbers because both these have very little value, if there is any, towards our goal of sentiment analylsis and topic modeling. For clarity I will explain a bit more on stop words here before going into coding. Stop words are some extremely common words used in a language which may carry very little value for a particular analysis. In this case I will use the stop words list comes along with ‘tm’ package. To get an idea of the list here are some example of stop words: a, about, above and so on. The exhaustive list can be found in this Github link: https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords.

```
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, removeNumbers)
writeLines(as.character(corpus[[1500]]))

## best quality underground metal detector  bangladesh
```

We can see from our sample tweet that a bunch of stop words (in) is removed.

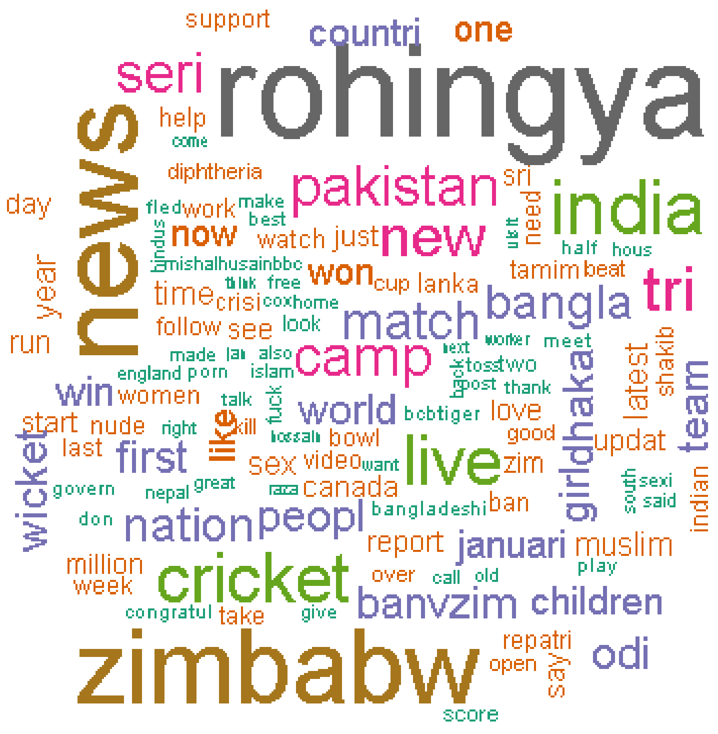
At this step I will convert the words in the corpus into stem words. In general terms, word stemming means the process of reducing a word to its base form which may or may not be the same as the morphological root of the word or may or may not bear meaning by the stem word itself. For example, all these words: ‘fishing’, ‘fisheries’ can be reduced to ‘fish’ by a stemming algorithm. Here ‘fish’ bears a meaning. But on the other hand this bunch of words: ‘argue’, ‘argued’ can be reduced to ‘argu’ in this case the stem doesn’t bear any particular meaning. Stemming a document makes it easier to cluster words and make analysis since. In addition to the stemming I will also delete my search key ‘Bangladesh’ from the tweets. Since I am analyzing tweets containing Bangaldesh and ‘amp’, it’s illogical to keep the term ‘bangladesh’ since that’s the search term and ‘amp’is abbrebiation of ’Accelerated Mobile Page’ which is a part of html link that improved web surfing experience from mobile devices.

```
corpus <- tm_map(corpus, stemDocument)
corpus = tm_map(corpus, removeWords, c("bangladesh","amp", "will", 'get', 'ca
n'))

writeLines(as.character(corpus[[1500]]))

## best qualiti underground metal detector
```

Again a recheck of our sample tweet we can see ‘quality’ has been tranformed into their stem form: ‘qualiti’ and ‘bangladesh’ has been removed.

I am finally done with our first step of data cleaning and pre-processing. On the next step I will start data processing to create our topic model. But before diving into model creation I decided to crate a word cloud to get a feel about the data.

```r
library(wordcloud)
set.seed(1234)
palet  = brewer.pal(8, 'Dark2')
wordcloud(corpus, min.freq = 50, scale = c(4, 0.2) , random.order = TRUE, col
= palet)
```



From the resulting word cloud we can see that the words are colored differently, which is based on the frequencies of the words appearing in the tweets. Looking at the most largest two fonts (black and green) we can find these words: rohingya, refuge, today, india, cricket, live. To interpret anything from such word cluster subject knowledge comes handy. Sinice I am from Bangladesh, I know that influx of Rohingya refugee from Myanmar is one of the most recent most discussed issue. Intuitively enough, Rohingya, refuge can be classified as related to the Rohingya crisis. On the other hand Cricket is the most popular game in Bangladesh along with other countries from south asian region. Cricket and Live can be thought to be related to Cricket. India and Today don’t have a general strong association with either of the two primary topics that we have sorted out. We will see how it goes in our further analysis in topic modeling.

As a next processing step now I will convert our corpus in a Document Term Matrix (DTM). DTM creates a matrix that consists all words or terms as an individual column and each document, in our case each tweet, as a row. Numeric value of 1 is assigned to the words that apprear in the document from the corresponding row and value of 0 is assigned to the rest of the words in that row. Thus the resulting DTM file is a sparse which is a large matrix containing a lot of 0.

```
dtm = DocumentTermMatrix(corpus)
dtm

## <<DocumentTermMatrix (documents: 5561, terms: 8561)>>
## Non-/sparse entries: 44969/47562752
## Sparsity           : 100%
## Maximal term length: 30
## Weighting          : term frequency (tf)
```

From the file summary of ‘dtm’ file we can see that it contains total 5,561 document, which is the total number of tweets that we have, and total 8,565 term, which shows we have total 8,565 unique words in our tweets. From the non/sparse entries ratio and the percentage of Sparsity we can see that the sparsity of the file, which is not exactly 100 but very close to 100, is very very high which is means lot of words appeard only in few tweets. Let’s inspect the ‘dtm’ file to have a feel about the data.

```
doc.length = apply(dtm, 1, sum)
dtm = dtm[doc.length > 0,]
dtm

## <<DocumentTermMatrix (documents: 5552, terms: 8561)>>
## Non-/sparse entries: 44969/47485703
## Sparsity           : 100%
## Maximal term length: 30
## Weighting          : term frequency (tf)

inspect(dtm[1:2,10:15])

## <<DocumentTermMatrix (documents: 2, terms: 6)>>
## Non-/sparse entries: 6/6
## Sparsity           : 50%
## Maximal term length: 6
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs dinesh india injur jan within year
##    1      0     0     0   0      1    1
##    2      1     1     1   1      0    0
```

We can see that out of the five terms first four terms were present in doc 2 and rest 2 terms were present in the doc 1. And accordingly the value of 1 and 0 have been distributed in the cells. Now let’s look at some of the most frequent words in our DTM.

```r
library(dplyr)
freq = colSums(as.matrix(dtm))
length(freq)

## [1] 8561

ord = order(freq, decreasing = TRUE)
freq[head(ord, n = 20)]

## rohingya     news  zimbabw    india   refuge     live  cricket    today
##      590      509      465      324      308      299      297      296
##      tri      new     camp     seri pakistan  myanmar    match   nation
##      266      248      245      239      238      236      221      203
##   bangla   wicket      odi    peopl
##      201      187      182      182
```

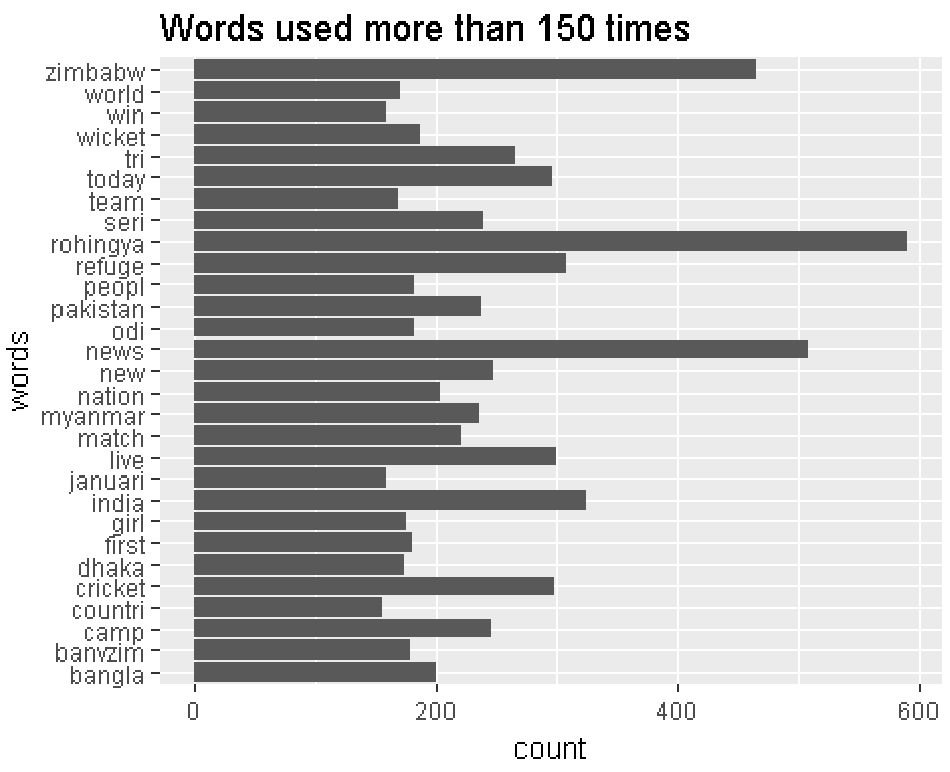
From the list of 20 most frequent words we can see that Rohingya crisis and Cricket related terms are the most frequntly used terms. Which shows resemblance with what we saw in our wordcloud. We can now see how different words are associated. Since we see that Cricket and Rohingy are two frequntly used topics, we can try to see which words associate with these two words. For this we will use 'findAssocs' command from 'tm' package. To run this command we need to provde the benchmark term and then a minimum value of correlation, which can range from 0 to 1.

```r
findAssocs(dtm, "cricket",0.2)

## $cricket
##     cup zimbabw   score    team
##    0.23    0.22    0.20    0.20

findAssocs(dtm, 'rohingya', 0.2)

## $rohingya
##   refuge     camp  myanmar  repatri    crisi     hous children
##     0.51     0.46     0.41     0.33     0.25     0.25     0.24

findAssocs(dtm, 'news', 0.2 )

## $news
##   today  latest  bangla   updat januari     atn  decemb ekattor  jamuna
##    0.77    0.77    0.76    0.68    0.56    0.53    0.34    0.33    0.28
## ekushey    post channel
##    0.26    0.21    0.21
```

From our resulting associations for both the words, we can see Cricket is associated with the words cup, zimbabwe, score and team. Which makes proper sense because every other words except Zimbabwe are related to game and Zimbabwe is one of the common team with whom Bangladesh have had quite a lot of cricket matches (such insights come from subject matter knowledge!). On the other hand, with the word Rohingya we can see assiciated words camp, refugee, myanmar, repatriation etc. evolve around the crisis created by the Rohingya refugees coming from Bangladesh's neighboring country Myanmar.

I will plot the most frequest 100 words now in a barplot to visually see how their frequencies are distributed. Checking the list of frequent words in list and graphically has two benefits: firstly, it gives a feeling about the analysis and secondly, it puts some sort of control on the quality of data cleaning done in previous steps. For example, after generating the most frequent words I found some of the words such as: amp, will, can, get are not removed. So I went back and added these words in the word remove step of data cleaning.

```
plot = data.frame(words = names(freq), count = freq)
library(ggplot2)
plot = subset(plot, plot$count > 150) #creating a subset of words having more
than 100 frequency
str(plot)
ggplot(data = plot, aes(words, count)) + geom_bar(stat = 'identity') + ggtitl
e('Words used more than 150 times')+coord_flip()
```



Data analysis:

Topic modeling using LDA

I have used ‘topicmodel’ package available in R for topic modeling. As discussed earlier, in LDA model number of topics are to be selected. Based on which LDA model creates the probability of each topic in each document and also distributes the words under each topic. Selecting more number of topics may result in more grannular segregation but at the same time the differences among different topics may get blurred. While on the other hand selecting very small number of topic can lead to losing possible topic. So to minimze this error I tried three different K or number of topics to create my LDA model. I used 2, 5, 10 as the number of topics and created three different LDA models based on these K values.

```r
library(topicmodels)
#LDA model with 5 topics selected
lda_5 = LDA(dtm, k = 5, method = 'Gibbs',
          control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TR
UE,
                         thin = 500, burnin = 4000, iter = 2000))

#LDA model with 2 topics selected
lda_2 = LDA(dtm, k = 2, method = 'Gibbs',
          control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TR
UE,
                         thin = 500, burnin = 4000, iter = 2000))

#LDA model with 10 topics selected
lda_10 = LDA(dtm, k = 10, method = 'Gibbs',
          control = list(nstart = 5, seed = list(1505,99,36,56,88), best = TR
UE,
                         thin = 500, burnin = 4000, iter = 2000))
```

LDA model produces a good bulk of information. But getting the most frequent words under each topic and document wise probability of each topic are the two most important pieces of information that I can use for my analysis purpose. First of all I will fetch top 10 terms in each topic:

```r
#Top 10 terms or words under each topic
top10terms_5 = as.matrix(terms(lda_5,10))
top10terms_2 = as.matrix(terms(lda_2,10))
top10terms_10 = as.matrix(terms(lda_10,10))

top10terms_5

##       Topic 1   Topic 2    Topic 3   Topic 4    Topic 5 
##  [1,] "news"    "rohingya" "zimbabw" "india"    "new"   
##  [2,] "live"    "refuge"   "cricket" "pakistan" "one"   
##  [3,] "today"   "camp"     "tri"     "peopl"    "year"  
##  [4,] "bangla"  "myanmar"  "seri"    "countri"  "day"   
##  [5,] "dhaka"   "girl"     "match"   "like"     "see"   
##  [6,] "januari" "children" "nation"  "time"     "just"  
##  [7,] "now"     "muslim"   "wicket"  "indian"   "work"  
##  [8,] "latest"  "say"      "odi"     "take"     "week"  
##  [9,] "updat"   "sex"      "banvzim" "don"      "follow"
## [10,] "love"    "million"  "world"   "nepal"    "last"  

top10terms_2

##       Topic 1    Topic 2 
##  [1,] "rohingya" "news"
##  [2,] "refuge"   "zimbabw"
##  [3,] "camp"     "india"
##  [4,] "myanmar"  "live"
##  [5,] "peopl"    "cricket"
##  [6,] "girl"     "today"
##  [7,] "countri"  "tri"
##  [8,] "children" "new"
##  [9,] "like"     "seri"
## [10,] "one"      "pakistan"
```

```
top10terms_10
```

```
##       Topic 1      Topic 2           Topic 3    Topic 4   Topic 5
##  [1,] "time"       "now"             "india"    "cricket" "zimbabw"
##  [2,] "dhaka"      "one"             "pakistan" "world"   "tri"
##  [3,] "bangladeshi" "report"         "like"     "team"    "seri"
##  [4,] "make"       "work"            "nepal"    "run"     "match"
##  [5,] "two"        "watch"           "hindus"   "canada"  "nation"
##  [6,] "islam"      "just"            "south"    "start"   "wicket"
##  [7,] "high"       "right"           "want"     "day"     "odi"
##  [8,] "state"      "indian"          "think"    "cup"     "first"
##  [9,] "also"       "mishalhusainbbc" "back"     "score"   "banvzim"
## [10,] "much"       "visit"           "take"     "play"    "win"
##       Topic 6 Topic 7   Topic 8    Topic 9 Topic 10
##  [1,] "girl"  "news"    "rohingya" "peopl" "new"
##  [2,] "sex"   "today"   "refuge"   "year"  "countri"
##  [3,] "love"  "live"    "camp"     "help"  "see"
##  [4,] "women" "bangla"  "myanmar"  "need"  "week"
##  [5,] "nude"  "januari" "children" "home"  "follow"
##  [6,] "video" "latest"  "muslim"   "give"  "last"
##  [7,] "kill"  "updat"   "say"      "look"  "england"
##  [8,] "fuck"  "sri"     "million"  "babi"  "australia"
##  [9,] "porn"  "lanka"   "support"  "sinc"  "bts"
## [10,] "dhaka" "post"    "repatri"  "even"  "set"
```

We can see that all three models picked the topics of Cricket and Rohingiya. But models with 5 and 10 topics also picked some other topics anlong with these two. From which we can see the application of the previous discussion about grannularity vs generalization. If we look at the top words from all the topics created from model with 10 topics, we can see that overall there is a lack of coherence among the words inside each topic. Similar observation can be made for the model with 5 topics. While the model with 2 topics provide two topics with a compact coherence among the topics. Another important thing to notice is that how the model with 10 topic picked some topic that were ignored by the model with 2 and 5 topics. Such as nudity (topic-6)!

Since we can clearly see that the topics of 'Rohingya Crisis' and 'Cricket' are two most common topics, I will move with these topic for further analysis.

<u>A little more discussion about LDA model output:</u>

To get a better idea about the outcome of LDA model we can get a summary of how many tweets were categorized as falling under which topic as follows

```
lda.topics_5 = as.matrix(topics(lda_5))
lda.topics_2 = as.matrix(topics(lda_2))
lda.topics_10 = as.matrix(topics(lda_10))

summary(as.factor(lda.topics_5[,1]))

##    1    2    3    4    5 
## 1208 1293 1230 1003  818

summary(as.factor(lda.topics_2[,1]))

##    1    2 
## 3151 2401

summary(as.factor(lda.topics_10[,1]))

##   1   2   3   4   5   6   7   8   9  10 
## 755 659 607 577 708 546 385 593 364 358
```

We can also get document wise probability of each topic against each document or here each tweet. I have created three files for each of my model and also saved the output for future use. Probability of each topic:

```
topicprob_5 = as.matrix(lda_5@gamma)
topicprob_2 = as.matrix(lda_2@gamma)
topicprob_10 = as.matrix(lda_10@gamma)

#write.csv(topicprob_5, file = paste('LDAGibbs', 5, 'DoctToTopicProb.csv'))
#write.csv(topicprob_2, file = paste('LDAGibbs', 2, 'DoctToTopicProb.csv'))
#write.csv(topicprob_10, file = paste('LDAGibbs', 10, 'DoctToTopicProb.csv'))

head(topicprob_2,1)

##           [,1]      [,2]
## [1,] 0.5409836 0.4590164
```

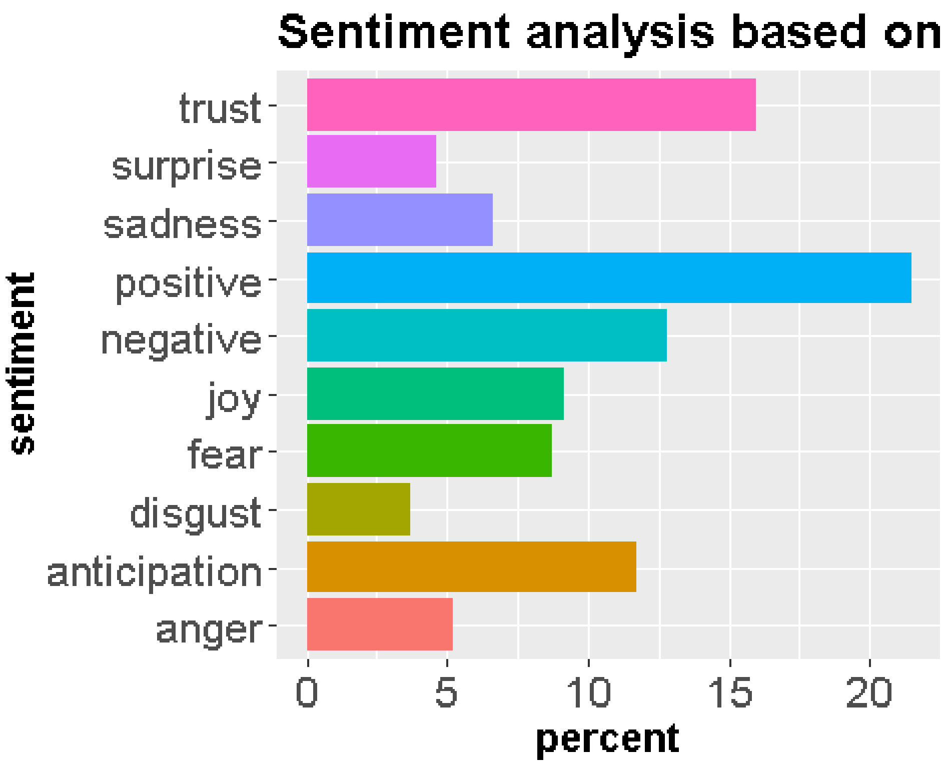
As a sample we can see that according to the model with 2 topics, for tweet 01 probability of topic 1 his higher compared of the probability of topic 2. Thus tweet 1 was categorized as falling under topic 1.

Sentiment analysis:

To conduct sentiment analysis, firstly tweets were converted into list of words. And then this list of words were matched with the word list of the NRC lexicon. To have a better comparison among different emotions, the result were converted into percentage. And thus the percentage of different emotions were plotted in the bar plot.

```
library(tidytext)
library(dplyr)
```

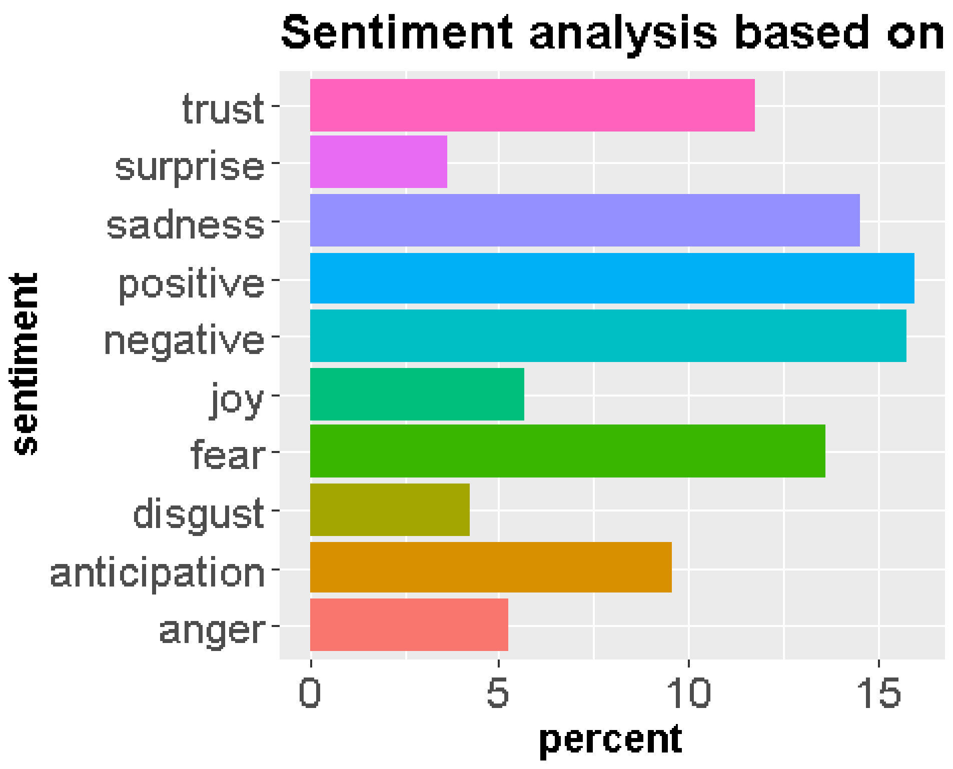
```r
library(ggplot2)

#Tokenizing character vector file 'tweets'.
token = data.frame(text=tweets, stringsAsFactors = FALSE) %>% unnest_tokens(w
ord, text)

#Matching sentiment words from the 'NRC' sentiment lexicon
senti = inner_join(token, get_sentiments("nrc")) %>%
  count(sentiment)
senti$percent = (senti$n/sum(senti$n))*100

#Plotting the sentiment summary
ggplot(senti, aes(sentiment, percent)) +
        geom_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity'
)+
        ggtitle("Sentiment analysis based on lexicon: 'NRC'")+
  coord_flip() +
        theme(legend.position = 'none', plot.title = element_text(size=18, fa
ce = 'bold'),
              axis.text=element_text(size=16),
              axis.title=element_text(size=14,face="bold"))
```



Additional analysis:

As an extension of the analysis so far, similar sentiment analysis was conducted over the tweets containing the key word of one of the two topics that we found: ‘Rohingya’. To conduct this, firstly tweets containing the word 'rohingya' were separated and then the same process was followed as described for sentiment analysis for overall tweets.

```
library(tm)
library(quanteda)

## quanteda version 1.0.0

## Using 3 of 4 threads for parallel computing

##
## Attaching package: 'quanteda'

## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, stopwords

## The following object is masked from 'package:utils':
##
##     View

corpus_roh = corpus(tweets)
corpus_roh = (corpus_roh = subset(corpus_roh, grepl('rohingya', texts(corpus_
roh))))
writeLines(as.character(corpus_roh[[150]]))

## the suffering in the rohingya refugee camp in bangladesh is indescribable

#Tokenizing character vector file 'tweets'.
library(tidytext)
token_roh = data.frame(text=corpus_roh, stringsAsFactors = FALSE) %>% unnest_
tokens(word, text)

#Matching sentiment words from the 'NRC' sentiment lexicon
library(dplyr)
senti_roh = inner_join(token_roh, get_sentiments("nrc")) %>%
  count(sentiment)

## Joining, by = "word"

senti_roh$percent = (senti_roh$n/sum(senti_roh$n))*100

#Plotting the sentiment summary
library(ggplot2)
ggplot(senti_roh, aes(sentiment, percent)) +
        geom_bar(aes(fill = sentiment), position = 'dodge', stat = 'identity'
)+
        ggtitle("Sentiment analysis based on lexicon: 'NRC'")+
  coord_flip() +
        theme(legend.position = 'none', plot.title = element_text(size=18, fa
ce = 'bold'),
              axis.text=element_text(size=16),
              axis.title=element_text(size=14,face="bold"))
```



Overall finding and discussion:

From our random walk on the tweets related to Bangladesh, we have seen ‘cricket’ and ‘Rohingya’ were the two areas that people cared most. Overall, people exuded a positive sentiment along with emotions of trust and anticipation. But in case of Rohingya crisis, we showed a mixed sentiment. About Rohingya issue both the positive and negative sentiments were high. Moreover, the emotions also seemed to be mixed. People felt sorry for the Rohingya people but they also expressed fear. So, what could that mean? Were people sympathetic to the plight of Rohingya but also had some share of fear related to the issue? This study doesn’t allow us to conclude on any such conclusion. But it gives us an idea of what we may achieve by having such a walk. Maybe we need to go for a walk with Bangladesh on the online social media sites more often to get a clearer image on what people talk about Bangladesh and what may needs to be improved to leave a better digital footprint for the country.